A System for Heuristic Modifications on PLA-Specifications

Reinhard Rauscher
University of Hamburg
Department of Computer Science
22527 Hamburg, Germany
rauscher@informatik.uni-hamburg.de

Andreas Krause
Philips Semiconductors
Multimedia-PC, Product Development
22529 Hamburg, Germany
krause@semicon.philips.de

Abstract

When specifying a PLA for the implementation of a numerical function, usually an ad hoc procedure is followed: Generating real values, some type of scaling, rounding to integers, and finally binary coding.

Examining the numerical exactness of the performed steps in detail, some kind of blurring can always be recognized. In fact, designers have to accept such unavoidable effects.

In many practical applications a good compromise can be achieved between a reasonable decrease of PLA costs and an acceptable amount of implicated deviation.

We introduce in this paper the optimization tool HOPE *which includes the described heuristics. It is applicable in any design procedure which needs PLAs and uses* ESPRESSO *as an integral part.*

1. Introduction

In EDA *(Electronic Design Automation)* the implementation of two-level logic networks with the regular structure of the PLA (*Programmable Logic Array*) offers many advantages; some of them are:

- regular layout structure
- small layout size dimensions
- less redundancy as in ROM-matrices
- efficient in common technologies

These advantages lead to lot of research, especially focussing on the optimization of the topological configuration of the PLA to minimize the layout size. In our local situation another aspect became more relevant: In various designs, the problem of an increasing power consumption [6] of PLA structures arose substantially, so that we had to spent extra effort on power supply [1]. However, this simple solution could be not acceptable [11] for a low-power circuit. In some cases, the *number of product terms*[1] of a PLA exceeded the limit of the PLA generator tool we used, so that we had to withdraw our approach of a PLA implementation [12]. When replacing the intended PLA by a ROM, we ended up in an increasing demand of size and delay. As an alternative to this unsatisfactory solution we had to implement an random logic network.

Generally, in the context of constantly rising complexity of the desired systems and the increasing demand on low-power circuits, the issue of PLA power consumption becomes a critical part of the design task. When asking our customers how exact the given specification (mostly a table of real values) has to be processed, we discovered that in most cases the customers response showed no really significant reason to interpret the values very exact. That is, there is always an application specific amount of deviation that the customer can accept. Therefore, our motivation was to search for interesting modifications [9] which are used to reduce PLA costs.

2. PLA-Specifications

Implementing a truth table for the sine function is a good example for a logic function suitable for PLA realization. As an *ad hoc* specification of the PLA we may choose a table that holds 100 entries in the range of $[0,\pi/2]$ with 1000.0 as a maximum value.

In Table 1 such a typical specification [9] of a PLA table is shown: According to the various states which have to be passed intermediate results are also shown. After defining the function a set of *discrete* values are scanned. For a suitable range of these values, the entries of the table have to be *scaled* e. g. by multiplying with factor 1E3. Rounding to integer values followed by coding results in the desired PLA specification.

Figure 1 shows the resulting graph of the *Sine* function as it is represented after rounding. All resulting discrete values

[1] The major criterion for PLA costs which is independent from design environment and technology.

i	$\sin(\frac{i\pi}{99 \cdot 2})$	$\cdot 1\text{E}3$	rnd()	truth table
0	0.000000	0.000	0	00 0000 0000
1	0.015866	15.866	16	00 0001 0000
2	0.031728	31.728	32	00 0010 0000
3	0.047582	47.582	48	00 0011 0000
4	0.063424	63.424	63	00 0011 1111
5	0.079250	79.250	79	00 0100 1111
6	0.095056	95.056	95	00 0101 1111
7	0.110838	110.838	111	00 0110 1111
⋮	⋮	⋮	⋮	⋮
97	0.999497	999.497	999	11 1110 0111
98	0.999874	999.874	1000	11 1110 1000
99	1.000000	1000.000	1000	11 1110 1000
100	-	-	-	** **** ****
⋮	⋮	⋮	⋮	⋮
127	-	-	-	** **** ****

Table 1. A specification of the Sine function

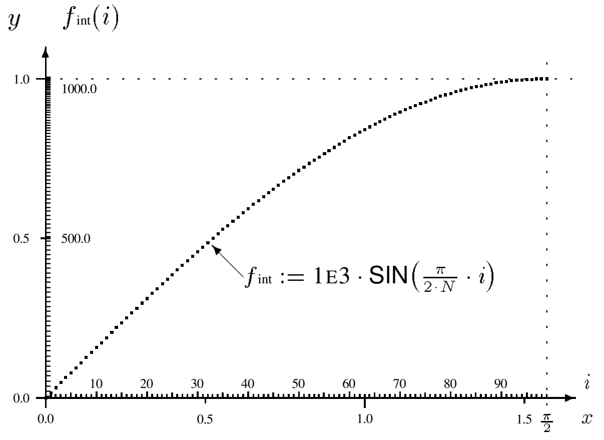

Figure 1. Graph of the Sine function

of domain and range have been marked. Note the increasing density of discrete values on the upper part of the y-axis.

The process of specification can be viewed as a top-down sequence of five steps. These steps are shown in Figure 2. According to each step the function is represented with changing sets for domain and range.

After *defining* the function and a real domain the function be represented as $f_{\text{real}} : I\!R \rightarrow I\!R$. The continuity of domain and range will be transformed by scanning discrete values, so that they could be indexed in a table; this state is represented by $f_{\text{ind}} : I\!N \rightarrow I\!R$. For a fixpoint range of suitable positive values, the scaled $f_{\text{fix}} : I\!N \rightarrow I\!R^+$ is given. The next step converts the entries by rounding into integers which result in $f_{\text{int}} : I\!N \rightarrow I\!N$. Finally a binary representation of f_{int} delivers the truth table for the PLA as $f_{\text{tab}} : I\!B^n \rightarrow I\!B^m$, where n and m denote the number of the PLA inputs and outputs.

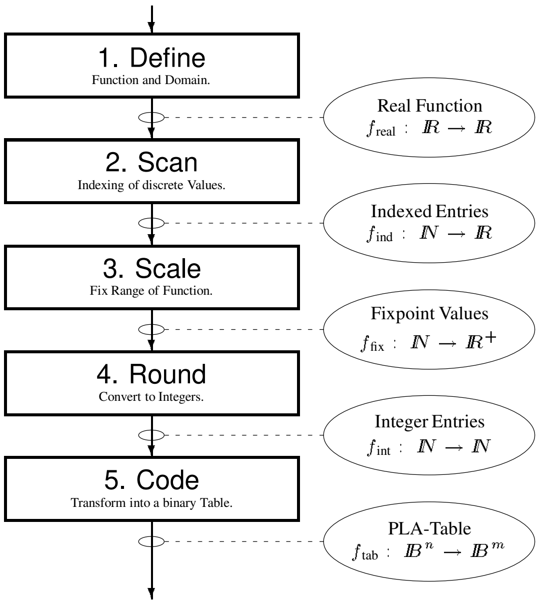

Figure 2. Top-Down Sequence of ad hoc Specification

3. Functional Overview

After defining a function and its domain in the first step of the sequence described above, certain transformations take place.

Step 2: Discrete domain values have to be scanned; let $N \in I\!N$ be the number of the resulting values:

$$\text{Dis}_N : I\!R \rightarrow \{0, \ldots, N-1\} \tag{1}$$

Step 3: Scaling for a positive fixpoint range; let $M \in I\!N$ be the maximum value:

$$\text{Scl}_M : I\!R \rightarrow \{q \in I\!R^+ \mid q \leq M\} \tag{2}$$

Step 4: Conversion to an integer range; e. g. standard rounding as in Table 1:

$$\text{Con} : I\!R^+ \rightarrow I\!N \tag{3}$$

Step 5: Binary representation to form a truth table; let $k \in I\!N$ the number of bits:

$$\text{Bin}_k : \{0, \ldots, 2^k - 1\} \rightarrow I\!B^k \tag{4}$$

All functional relationships are illustrated in a scheme which is presented in Figure 3. The acyclic graph shows sets as circles and transfer functions as arrows.

Starting at the set $I\!R$ all transformations finally end at the set $I\!B^m$. The top-down sequence of a specification can be found in the vertical center part of the scheme: The various versions of f are indexed, starting with f_{real} at the top and ending with f_{tab} at the bottom. On the left and right

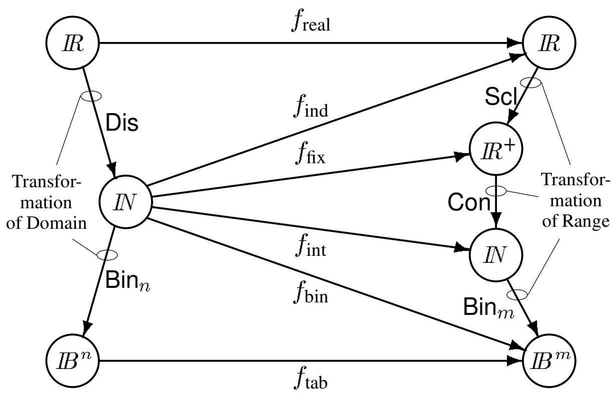

Figure 3. Scheme of Transformational Relations

edges of the scheme, the domain and range related transformations are indicated. When doing a specification the passing of transfer functions through the scheme may be viewed as a progress. This progress we call the *transfer path*.

4. Specifications and Exactness

Before discussing any modifications on specifications, two aspects of inaccuracy have to be considered: *Blur* and *Options* which are closely related to the topic of *digitizing* of analog signals.

4.1. Blur is Inherent

The basic idea comes from the circumstance that on each step of a specification a certain degree of inaccuracy exists in the performed transformation. When passing the transfer path, typical effects of *Blur* occur:

B-1 **Scan Errors** are evident because single values represent intervals.

B-2 **Random Matching** for scanned domain values is unsufficient for special-interest points, e. g. *Zero Passages* or *Extreme Values*.

B-3 **Conversion Errors** happen, e. g. when rounding real values to integers.

To a certain degree, these errors are unvoidable. Therefore, *Blur* is inherent.

4.2. Options are Hidden

During an *ad hoc* specification as described above at some points certain decisions have to be made. Up-coming questions are mostly answered in a quick and standard way, sometimes without considering alternatives. There are several occasions where *Options* provide decision points, e. g. the following are considerable:

O-1 Variable scan gap may be helpful when storage gets wasted because of an over-extreme high resolution.

O-2 The maximum value in a range is determined by scaling. It should be a power of 2.

O-3 Non-linear scaling helps to sparse values in range and to decrease data-storage amount without loosing exactness.

O-4 Coding *don't cares* is adequate in cases where rounding of a real value results in an interval of at least two integers.

Sometimes, when discussing this kind of decisions with customers, they do not seem to be aware of them.

5. Heuristic Modifications

When realizing the occurences of *inherent blur* and *hidden options* special interest lays in methods to modify specifications. The general goal is to minimize costs of the implementing PLA block.

5.1. Hypotheses for Heuristics

As common in VLSI design the methods presented in this paper are heuristics. As a consequence, the desired effect cannot be granted for any case. Some possible hypotheses are the following:

H-1 Alternative conversions of real values to integers can be achieved aside form just *rounding-to-next* integer which generally causes different PLA costs.

H-2 Single `1`-entries inside a truth table for a PLA lead to less PLA costs.

H-3 Interval entries model the tolerance allowed for each value and are given by blurring effects.

H-4 Selections out of intervals can be done to reduce PLA costs.

H-5 Minimal numeric modifications on entries may reduce the number of `1`s.

H-6 Identical entries on specific positions in the truth table reduce PLA costs.

H-7 Additional application of the *Consensus method* on the set of terms reduces the number of terms and therefore also PLA costs.

5.2. Heuristic Methods and Modes

According to the hypotheses in the previous section some heuristic methods are developed in detail in the following:

5.2.1 Conversion via Rounding

Refering to Hypothesis H-1 (Section 5.1), the method $\mathsf{Rnd} : \mathbb{R}^+ \rightarrow \mathbb{N}$ of rounding positive real values to positive integers can be done in at least three alternative ways. They may be derived from the well-known *truncation*, which just separates those digits ahead of the comma position of a real number. *Rounding* as $\mathsf{Rnd}(q), q \in \mathbb{R}^+$ is a method of conversion and may be split into the following three modes:

$$\begin{array}{lclcl} \mathsf{Floor}(q) & := & \lfloor q \rfloor & := & \mathsf{Trunc}(q) \\ \mathsf{Ceil}(q) & := & \lceil q \rceil & := & \mathsf{Trunc}(q) + 1 \\ \mathsf{Round}(q) & := & \langle q \rangle & := & \mathsf{Trunc}(q + \frac{1}{2}) \end{array} \qquad (5)$$

5.2.2 Intervals inside Conversion-Layer

As pointed out by Hypothesis H-3 intervals provide the opportunity of choosing between more than one value for an entry of the table. Before that choice may be done on each entry $i \in \mathbb{N}$, the assignment of an interval $[v_i, w_i]$ with $v_i, w_i \in \mathbb{N},\ v_i < w_i$ causes a non-unique relation of values in $[v_i, w_i]$ to an entry i. Such relation is defined as follows:

$$r_{\text{Int}} \quad \subseteq \quad \mathbb{N} \times \mathbb{N} \qquad (6)$$

$$\text{where: } (i, u_i) \in r_{\text{Int}} \quad \Longleftrightarrow \quad u_i \in [v_i, w_i] \qquad (7)$$

Instead of functions we do face here relations and in logical context they appear as *Boolean Relations*, i. e. single input assignments relate to multiple output assignments [4], [10] and [5]. Algorithms and heuristics for minimizing such relations have been presented recently ([3], see [7]).

For a real argument $q \in \mathbb{R}^+$ there are three modes to generate an interval $[v, w],\ v, w \in \mathbb{N}$. The modes are defined as follows:

- $\mathsf{Int\_UpDn}$ assigns the interval which is spanned by the two integers closest to q:

$$\mathsf{Int\_UpDn}(q) := \Big[\mathsf{Floor}(q), \mathsf{Ceil}(q)\Big] \qquad (8)$$

In contrast to this mode, large intervals can be selected because the *inherent blur* for each entry may be of a greater effect than modeled through the first mode. Occassionally, the actual dimension of tolerance may be extracted from the context of the problem [11].

The more integers are in an interval, the greater is the possibility of a suitable selection of an integer in the interval which results in less PLA costs. However, tolerance modeled as an interval $[v, w]$ with $v \leq q \leq w$ and $v, w \in \mathbb{N}$ does not necessarily have to be symmetric with $q \in \mathbb{R}^+$ in the center. Therefore, it makes sense to specify the distances of $q - v$ (lower) and $w - q$ (higher) separately.

- $\mathsf{Int\_Abs}$ assigns *absolutely* specified intervals; the lower and higher distances are given with $a, b \in \mathbb{N}$:

$$\mathsf{Int\_Abs}\Big|_a^b(q) := [v, w] \qquad (9)$$

$$\text{with } v := \begin{cases} \lfloor q \rfloor - a : & \lfloor q \rfloor \geq a \\ 0 \qquad : & \text{otherwise} \end{cases}$$
$$\text{and } w := \begin{cases} \lceil q \rceil + b : & \lceil q \rceil < M - b \\ M - 1 : & \text{otherwise} \end{cases}$$

- $\mathsf{Int\_Rel}$ assigns *relatively* specified intervals; the two distances to q are given as percentages $a, b \in \mathbb{N}$:

$$\mathsf{Int\_Rel}\Big|_a^b(q) := [v, w] \text{ where } a, b \leq 100 \qquad (10)$$

$$\text{with } v := \Big\lfloor q \cdot (1 - \tfrac{a}{100}) \Big\rfloor \geq 0$$
$$\text{and } w := \Big\lceil q \cdot (1 + \tfrac{b}{100}) \Big\rceil$$

i	q_i	$\mathsf{Int\_UpDn}(q_i)$	$\mathsf{Int\_Abs}\Big\|_3^2(q_i)$	$\mathsf{Int\_Rel}\Big\|_{5\%}^{5\%}(q_i)$
0	0.0	0	[0,2]	0
1	2.5	[2,3]	[0,5]	[2,3]
2	5.7	[5,6]	[2,8]	[5,6]
3	13.1	[13,14]	[10,16]	[12,14]
4	22.4	[22,23]	[19,25]	[21,24]
5	30.9	[30,31]	[27,33]	[29,33]
6	47.8	[47,48]	[44,50]	[45,51]
7	69.2	[69,70]	[66,72]	[65,73]

Table 2. Demonstration of the Interval Method

The three presented modes of the interval generating process are demonstrated in Table 2 for a 7-bit representation: There are eight entries for which these modes have been evaluated with distances (a, b) of absolutely $(-3/+2)$ for $\mathsf{Int\_Abs}$ and relatively ($\pm 5\%$) for $\mathsf{Int\_Rel}$. For $\mathsf{Int\_UpDn}$ all generated intervals have a span of 1, i. e.. the coding of the LSB for [2,3] with `*` is possible, but not for [5,6]. The binary representation of interval [47,48] is [`111111`,`110000`] and according to hypothesis H-2 (see Section 5.1) the selection of the higher border 48 is a more suitable choice for less PLA costs.

5.2.3 Conversion by Selection

In combination with the generation of intervals, selection is the process of selecting a single particular integer for each entry out of its interval. It is based on hypothesis H-4.

$$\mathsf{Sel} : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N},\ u := \mathsf{Sel}(v, w),\ v < w \qquad (11)$$

Relating to an interval $[v, w]$ a single integer u has to be selected. Besides the two modes $\mathsf{Sel\_Low}(v, w) := v$ and $\mathsf{Sel\_High}(v, w) := w$, which trivially select either the lower or the higher border of an interval, there are at least two additional modes:

- Mean Value: $\mathsf{Sel\_Mean}(v, w) := \langle \frac{v+w}{2} \rangle$. This mode selects the rounded mean value of the interval.

- Minimal-1 Value: $\mathsf{Sel\_Min1}(v, w)$ selects the integer inside the interval $[v, w]$ which has the minimal number of `1`s. For this purpose v, w have to be examined in their m-bit representation: $v_{m-1} \dots v_0 := \mathsf{Bin}_m(v)$ and $w_{m-1} \dots w_0 := \mathsf{Bin}_m(w)$.

In Table 3 all selection modes are demonstrated on intervals generated with Int_Abs which are taken from Table 2.

Int_Abs		Selection Modes			
i	[v,w]	Sel_Low	Sel_High	Sel_Mean	Sel_Min1
0	$[0, 2]$	0	2	1	2
1	$[0, 5]$	0	5	2	4
2	$[2, 8]$	2	8	5	8
3	$[10, 16]$	10	16	13	16
4	$[19, 25]$	19	25	22	24
5	$[27, 33]$	27	33	30	32
6	$[44, 50]$	44	50	47	48
7	$[66, 72]$	66	72	69	72

Table 3. Demonstration of Selection Modes

5.2.4 Similarity of Entries

Comparing two entries of a truth table may result in an identical match which is a good condition for a later logic minimization step. In a truth table identical entries in pairs of certain indices support the covering combination of minterms [2]. This is exploited by the *Consensus* method (see [8]) which is part of usual logic minimizers. However, two unequal entries will fail that goal even if they compare quite similar to each other. In this context *similarity* can be interpreted either in binary or numeric consideration. In order to achieve reduced PLA costs according to hypothesis H-6 an assimilation of such *similar* entries proves useful.

Aside from measuring numeric similarity by looking at the difference of two entries, the *Hamming distance* may be used to measure binary similarity.

This part introduces the method Sim which splits into three steps:

- **Classify:** In order to support the *Consensus* method of the logic minimizer, only those entries are classified to be comparable, which have binary indices with a *Hamming distance* of 1. In a table this counts as well for consequtive as for widely separated pairs of entries; an assimilation of entries in the later case could result in discontinuity of the represented numeric function. Therefore, an additional restriction to only consequtive pairs of entries seems reasonable. Both aspects result in the condition that those pairs of entries are comparable which differ only in their LSBs.
- **Examine:** Only for comparable pairs of entries (named Y_i and Y_j) the present similarity has to be examined either by calculating the numerical difference $|Y_j - Y_i| \in I\!N$ or by evaluating the *Hamming distance* of the binary entries $\mathsf{Ham}(a, b) \in I\!N$ where $a := \mathsf{Bin}_m(Y_i)$ and $b := \mathsf{Bin}_m(Y_j)$.
- **Assimilate:** If the measure of similarity of a pair of entries reaches a given limit $z_{\text{num}}, z_{\text{bin}} \in I\!N$ this pair is qualified to get assimilated. The resulting change in the truth table should be that the one with more `1`s gets substituted by the other.

Let the numeric mode of this method be called Sim_Num and the binary Sim_Bin, respectively. Additionally, two limit values are necesary: z_{num} and z_{bin}. For z_{bin} is specifying the maximum number of deviating bits which will be tolerated to assimilate two entries. The numeric similarity has a limit in the maximum difference of all classified entries:

$$z_{\text{num}} \leq \max\left(\left\{|Y_i - Y_j| \;\middle|\; i \neq j \text{ classified}\right\}\right) \tag{12}$$

Table 4 shows a demonstration of both Sim modes Sim_Num and Sim_Bin: With help of the left-hand part the entries can be classified refering to the different LSB. The resulting pairs of entries are grouped through horizontal lines. In the middle part the entries and their *Hamming distance*, and their numerical difference are listed. With this information the examination of the similarity can be done. On the right part of Table 4 the performed assimilation of either mode is shown: The acceptable limits are given for Sim_Bin with a maximum *Hamming distance* of $z_{\text{bin}} := 3$ and a maximal numerical difference of $z_{\text{num}} := 13$ for Sim_Num.

6. The Optimization Tool HOPE

6.1. Solutions and Strategies

In this Section we introduce HOPE as a design tool. With HOPE designers are able to optimize a given specification for a PLA by slightly modifying it and yielding less costs. The optimization idea of HOPE is to generate a set of specification objects, i. e. many possible solutions which are in fact truth tables, each specifying an individual PLA. One of them is declared as a reference to be compared to the others. Mostly the designer wants the initial *ad hoc* specification of the customer to be the reference object because that one has to be improved.

Possible solution objects are generated with the heuristic methods of Section 5 which have been implemented in HOPE [9]. With HOPE the design procedure follows one of two basic strategies which are focussing either on *deviation* or *costs*:

comparable		similar					Assimilation	
i	$\mathsf{Bin}_7(i)$	Y_i	$\mathsf{Bin}_{10}(Y_i)$	#1	Ham()	Diff.	Sim_Bin $z_{\text{bin}} := 3$	Sim_Num $z_{\text{num}} := 13$
32	001 0000	486	01 1110 0110	6	2	14	-	-
33	001 0001	500	01 1111 0100	6			equal #1	$Y_{33} - Y_{32} > z_{\text{num}}$
34	001 0010	514	10 0000 0010	2	3	13	$Y_{35} := Y_{34}$	$Y_{35} := Y_{34}$
35	001 0011	527	10 0000 1111	5				
36	001 0100	541	10 0001 1101	5	5	13	-	$Y_{36} := Y_{37}$
37	001 0101	554	10 0010 1010	4			Ham $> z_{\text{bin}}$	
38	001 0110	567	10 0011 0111	6	5	13	-	$Y_{38} := Y_{39}$
39	001 0111	580	10 0100 0100	3			Ham $> z_{\text{bin}}$	

Table 4. Demonstration of Similarity Modes

- **Deviation Strategy:** For all possible solution objects the PLA costs are determined. Only those are considered as *candidates* for the final solution which result in less costs compared to the reference object. The final selection is done by examining the *deviation* between the current and the reference object. That object with minimal deviation is the final solution.
- **Cost Strategy:** In most cases, designer and customer together are able to estimate the tolerance for the table's entries by reflecting the context of the problem. After generating possible solution objects, the set of *candidates* for the final solution is separated by examining the individual deviation between the current and the reference object. Finally the costs of those *candidates* are determined and the one with minimal costs is selected for final solution.

Note that both strategies do not result in an unique solution. Therefore, designer and customer together have to discuss the results and have to decide which solution is finally selected.

So, the process of optimization which consists of HOPE and a *Manual Step* performed in cooperation of designer and customer, is in general executed interactively. As a consequence this optimization approach performed with HOPE does - optionally - not work automatically.

6.2. System Overview

HOPE is implemented in the *C* programming language. It has been developed for several operating systems simultaneously, like UNIX and MSDOS.

According to the transformation sequence a specification is a track of steps which may be manipulated by HOPE. When executing a heuristic method HOPE performs certain modifications on a steps of the sequence. As a result another (pre-)solution object will be created; in the terminology of HOPE these will be called `objects`. They may be of type *numeric* for results after re-scaling a table of real values or *binary* for results that could be in fact a truth table specifying a PLA. In HOPE the totality of all `objects` is called `set`.

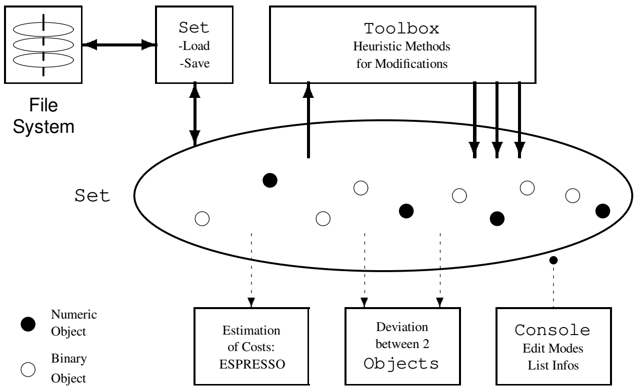

Figure 4. System components of HOPE

The system components of HOPE are shown in Figure 4: Besides storage facilities (`Set`-Load/-Save) for the whole `set` of `objects` and a component to view and control the data structure (`Console`), HOPE offers a collection of heuristic methods (`Toolbox`) to perform object-manipulating functions. For statistics there are components which estimate PLA costs (a code-integrated version of ESPRESSO [2]) and determine the deviation between two `objects`.

7. Evaluation of Examples

7.1. Illustration with Sine

In order to evaluate the heuristic methods introduced the *ad hoc* specified *Sine* function described above serves as a first illustration; as an equivalent of PLA costs, it caused initial 93 terms.

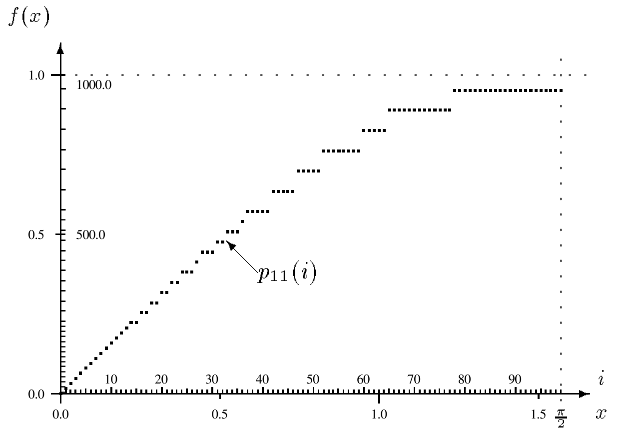

Figure 5. Graph of p_{11}, reduction: 67%

One solution hit an extremly high reduction of costs by greatly deviating from the referenced *ad hoc* specification. Let that solution be called p_{11} for which we executed method Int_Rel with distances of $\pm 5\%$ to the initial real values. Secondly method Sel_Min1 has been invoked and resulted in 31 terms which is a reduction of 67%. The graph in Figure 5 shows significant changes to the original graph (see Figure 1) which may not be acceptable by the customer.

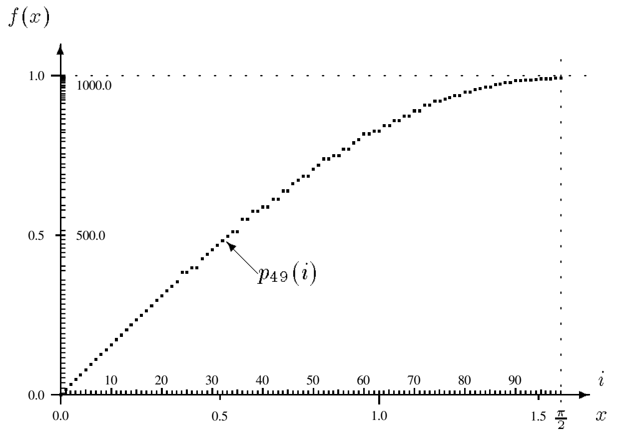

Figure 6. Graph of p_{49}, reduction: 24%

The graph of another solution p_{49} looks much closer to the desired *Sine* function as shown in Figure 6 and still reduces PLA costs about 24% (down to 71 terms): After the *ad hoc* specification only method Sim_Num with $z_{\text{num}} := 14$ of Section 5.2.4 has been executed.

7.2. Set of Examples

In this Section five examples are presented. For each example an *ad hoc* specifiedPLA-table has been generated initially for the purpose of a reference solution.

`sine` In this example the *Sine* function has been defined; the PLA-table has $N := 2^7$ entries ($n := 7$), of $m := 10$ bits each, i. e. a $n \times m$ matrix of 7×10 bits. The function is defined by:

$$f(i) := \sin\left(\frac{\pi}{2} \cdot \frac{i}{N-1}\right),\ x \in [0, \tfrac{\pi}{2}] \qquad (13)$$

A minimization via ESPRESSO resulted in 118 terms.

`root` This example has its origin in a software distribution of ESPRESSO [2, Table 7.1 (p. 165)]. It is *ad hoc* specified PLA-table with an 8×8 bits format. The values of the function are:

$$f(x) := \sqrt{x},\ x \in I\!N,\ 0 \leq x < 2^8 \qquad (14)$$

A minimization via ESPRESSO[2] resulted in 58 terms.

`expo` With constants $N := 2^6$ and $a := 2^{16} - 1$ another PLA-table of 6×16 bits is defined:

$$f(x) := \exp\left(\frac{x}{N} \cdot \log(a)\right),\ x \geq 0,\ a > 0 \qquad (15)$$

Minimizing via ESPRESSO results in 58 terms.

`loga` The invers function to `expo` is defined as `loga` with $N := 2^7$ entries and a constant $a := 13$:

$$f(x) := a \cdot \log_e(x),\ a, x > 0 \qquad (16)$$

After running ESPRESSO 59 terms result.

`hbol` The hyperbolic function is calculated as follows:

$$f(x) := a \cdot \frac{1}{x},\ 0 < x < N,\ a > 0 \qquad (17)$$

A minimization via ESPRESSO obtains 24 terms.

The presented heuristic methods of Section 5.2 have been evaluated in all their modes to modify the *ad hoc* specification and generate deviating solutions. These were examined according to the suggested *Cost strategy* of Section 6.1: Actually, the selection of *candidates* was done by allowing a defined tolerance for all entries of the reference solution. To estimate realistic *blur* on the real values we defined three ranges of tolerance related to a real $q \in I\!R$:

- *Round-Up/-Down* spans the interval just by down- and up-rounding, like Int_UpDn in (8).
- *Absolut* ± 2 spans the interval, like Int_Abs in (9).
- *Relativ* $\pm 5\%$ spans the interval, like Int_Rel in (10).

In Table 5 the examples are listed, beginning on the left part with their *name* and $n \times m$ bit-format (PLA-I/O). The right part is splitted into three sections, one for each case of allowed tolerance as described above. Inside the table those

[2] In [2, Table 7.1, p. 165] this example is listed with 57 terms. This result has been obtained in the *strong* mode of ESPRESSO which is most effectiv but in turn most time-consuming.

Example		Cost-Reduction depending on allowed Tolerance					
Name	$n \times m$	Round: Up/Dn		Absolut: ± 2		Relativ: $\pm 5\%$	
sine	7×10	Sel_Mean	8%	Sel_Min1	12%	Sel_Min1	53%
		Sel_Min1	7%	Sel_Mean	8%	Sim_Num	25%
		Rnd_Ceil	6%	Rnd_Ceil	3%	Rnd_Ceil	3%
root	8×5	Sel_Min1	40%	Sel_Min1	67%	Sel_Min1	43%
		Rnd_Floor	34%	Rnd_Floor	34%	Rnd_Floor	34%
		Sim_Bin	28%	Sim_Bin	28%	Sim_Bin	28%
expo	6×16	Sel_Mean	3%	Sel_Mean	3%	Sel_Min1	12%
		Rnd_Floor	2%	Sim_Num	3%	Rnd_Floor	2%
		Sim_Num	2%	Rnd_Floor	2%	Sim_Num	2%
loga	7×6	Sel_Min1	24%	Sel_Min1	56%	Sel_Min1	56%
		Rnd_Ceil	5%	Sim_Num	29%	Sim_Num	27%
		–	–	Rnd_Ceil	5%	Rnd_Ceil	5%
hbol	6×7	Sel_Min1	25%	Sel_Min1	46%	Sel_Min1	38%
		Rnd_Ceil	8%	Sim_Num	13%	Rnd_Ceil	8%
		Sim_Num	8%	Rnd_Ceil	8%	Sim_Num	8%

Table 5. Achievable Reduction of PLA-Costs depending on allowed Tolerance

top three methods are shown which obtained the highest reduction of PLA costs under the condition of meeting the allowed tolerance. Obviously, the two modes Sel_Mean and Sel_Min1 of Section 5.2.3 dominate the results. The two upright entries Sel_Min1 with 53% and Sim_Num with 25% reduction corresponds to the p_{11}- and p_{49}-graph in Figure 5 and Figure 6, respectively. Reduction is smaller because `sine` has more entries (128 instead of 100) and higher initial PLA costs (118 instead of 93 terms).

8. Conclusions

The analysis of results achieved with the system HOPE shows, that the system could produce remarkable improvements of the PLA implementation while using only slight modifications of the desired behaviour. As an example see Figures 1 and 6. The functionality between the the original specification and the implementation differs only slightly, but the reduction (24%) in power consumption and area is impressive.

References

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

[2] R. K. Brayton and G. D. Hachtel and C. T. McMullen and A. L. Sangiovanni-Vincentelli: Logic Minimization Algorithms for VLSI Synthesis, Kluwer Academic Publishers, Boston, MA, 1984

[3] R. K. Brayton and F. Somenzi: An Exact Minimizer for Boolean Relations, in IEEE Int. Conf. on CAD, 316-319, Nov. 1989

[4] R. K. Brayton and F. Somenzi: Minimization of Boolean Relations, in ISCAS 89, Portland, OR, May 1989

[5] R. K. Brayton, Y. Watanabe: Heuristic Minimization of Multiple-Valued Relations, IEEE Transactions on CAD of Integrated Circuits and Systems, CAD-12(10: 1458-1472, Oct. 1993

[6] S. Charoun, M. Dana, E. Melcher, W.Röthig: Power Consumption Estimation Using Statistical Signal Properties, In Proc. EUROMICRO 92, 691-696, Paris, 1992

[7] A. Ghosh, S. Devadas, A. R. Newton: Heuristic Minimization of Boolean Relations Using Testing Techniques, in IEEE Int. Conf. on Computer and Design, 227-281, Cambridge, Sept. 1990

[8] D. Givone: Introduction to Switching Circuit Theory, McGraw-Hill, New York, 1970

[9] A. Krause: Heuristische Optimierungen an Spezifikationen von PLA-Tabellen, Diploma thesis, University of Hamburg, Sept. 1995

[10] J. A. Larkin: The Application of Iterated Consensus to Multiple - Valued Logic Design,

Technical report GE/EE/75-6, Air Force Institute of Technique, Wright - Patterson AFB, Ohio, June 1975

[11] R. Rauscher: Entwicklung und Implementierung eines ICs zur krümmungsadaptiven Deltamodulation, University of Hamburg, Internal Report, FBI-HH-M-210/92, Aug. 1992

[12] R. Rauscher: A System for PLA Minimization based on Slight Behaviour Modifications, Proc. EUROMICRO 95, 8-15, Como, Sept. 1995