A New Approach to Logic Synthesis of Multi-Output Boolean Functions on PAL-based CPLDs

Dariusz Kania

Silesian University of Technology, Institute of Electronics

ul. Akademicka 16, 44-100 Gliwice, Poland

tel. (48) 32 237 22 86

Dariusz.Kania@polsl.pl

ABSTRACT

A PAL-based logic block is the core of great majority of contemporary CPLD devices. The purpose of the paper is to present a new approach to multi-level synthesis for PAL-based CPLDs. The presented approach is an alternative to the classical method based on two-level minimization of separate single-output functions. The essence of the presented method is to search for multi-output implicants that can be shared by several functions. This approach presents a original form for illustrating a minimized form of a multi-output Boolean function. Graph node represents groups of multiple-output implicants with the common output part. The graph analyses allow to effective implementation of multi-output function in PAL-based devices. Results of experiments, which are also presented, prove that the proposed algorithm leads to significant reduction of chip area in relation to the classical method, especially for CPLD structures consisting of PAL-based logic blocks containing low number of product terms.

Categories and Subject Descriptors:

B.6.3 [**Design Aids**]

General Terms: Algorithms

Keywords: logic synthesis, technology mapping, CPLDs

1. INTRODUCTION

The most of CPLDs consist of logic blocks with internal structures that resemble architectures of simple PALs (Programmable Array Logic) – see Fig1. Distinct from the other, less popular group of PLA-based (PLA – Programmable Logic Array) CPLD architectures, further on in this paper such devices will be referred to as PAL-based CPLDs.

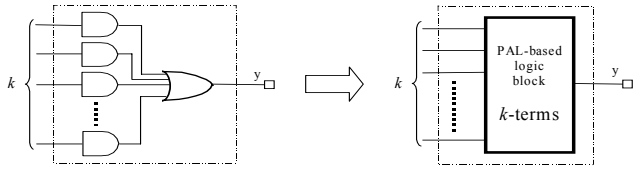


Figure 1. The structure and block - scheme of PAL-based logic block with *k*-terms

GLSVLSI'07, March 11–13, 2007, Stresa-Lago Maggiore, Italy.

The classical method of logic synthesis, dedicated for PAL-based CPLDs, consists of two steps. First a two-level minimisation is applied separately to every single-output function, next implementation of the minimised functions in PAL-based blocks containing a predefined number of product terms is performed. If the number of implicants p, representing a function after minimisation, is greater than the number of product terms k, available in a logic block (Fig. 1), a greater number of logic blocks have to be used to implement the function. The classical product term expansion consists in introducing cascaded feedback connections, increasing propagation delays between inputs and outputs. Some other concepts of product term expansion are also presented in literature, e. g. in [3]. Multi-level strategy is used in logic synthesis dedicated for different types of programmable devices. Synthesis algorithms dedicated for PLA structures are known. In some other methods algorithms developed for LUT-based FPGAs were directly adapted to another PLD architectures [1, 2]. A characteristic feature of algorithms of this kind is a process of appropriate coding of inputs and outputs, which significantly influences minimisation of product term numbers in blocks obtained as the result of decomposition. The main limitation of PAL-based logic blocks is the number of multi-input product terms available in one block. This results in observation, that the essence of synthesis dedicated for PAL-based structures can be reduced to two major tasks: minimising the number of PAL-based logic blocks used and adjusting the designed circuit to fit the structures of PAL-based blocks best. The proposed multi-level synthesis concerns directly above issue. The objective of this paper is to present multi-level synthesis for PAL-based CPLDs. The method consists in searching for the common multi-output implicants [5]. The process of searching for the common implicants is carried out after having completed the two-level minimization of the multi-output function by means of the Espresso algorithm.

2. THEORETICAL BACKGROUNDS

Let f be a multi-output logic function $f:B^n \rightarrow B^m$, where $B=\{0,1\}$. The classical implementation of the function $f:B^n \rightarrow B^m$ within the PAL-based structures is related to implementation of the minimized functions $f_o:B^n \rightarrow B^1$ (o =1,2,…,m) by means of the PAL-based logic blocks. Let the discriminant Δ_{fo} be the decimal number equal to the number of those implicants, provided function $f_o:B^n \rightarrow B^1$ constitutes true values. Let δ_{fo} denote the number of logic blocks necessary for implementation of the o^{th} function. In the case when $\Delta_{fo}>k$, implementation of the f_o function by means of the PAL-based logic blocks consisting of k terms needs the realization of feedback loops. The number of $\delta_{fo}=\lceil(\Delta_{fo}-k)/(k-1)\rceil+1$

PAL-based logic blocks consisting of *k*-terms will be used, where $\lceil x \rceil$ denotes the lowest integer number not less than *x*. For implementation of *m*-functions (every function has been minimized separately), implementation of δ^1_f PAL-based logic blocks is necessary, where $\delta^1_f = \sum_{o=1}^{m} \left(\left\lceil \frac{\Delta f_o - k}{k-1} \right\rceil + 1 \right)$.

The minimized form of multi-output functions $f:B^n \rightarrow B^m$ can be described by a set of multi-output implicants, including an input part consisting of components {0,1,-} and an output part consisting of {0,1} components [5]. Let *y* be an *m*-component output vector that is associated with the output part of the multi-output implicant. The decimal number equal to the number of the same y_i vectors that constitute the set of multi-output implicants defining the $f:B^n \rightarrow B^m$ function will be called the discriminant Δ_y. Let $\mu(\Delta_y)$ (range of Δ_y discriminant) be a decimal number equal to the number of {1} components included in the *y* vector. Let's assume, that $G<Y,\bar{U}>$ is the directed graph, where Y is the set of all the graph nodes Δ_y, while $\bar{U}$ is a set of graph edges connecting the such nodes of the graph Δ_{ys}, Δ_{yr}, that the code distance of the y_s, y_r vectors is 1, and $\mu(\Delta_{ys})+1=\mu(\Delta_{yr})$. By means of elimination from the primary graph such nodes, where $\Delta_y=0$, we obtain the reduced graph presented in the Fig. 2. For simplification, the nodes of the graph contain decimal value of descriminants only. Every node of the first range that is related to the implicants of the o^{th} output of the *m*-output function can be associated with the decimal value of Δ^m_o equal to the sum of discriminants included in nodes covered by all the paths starting from this node and ending in nodes of the upper ranges, can be associated with every node of the first range (Fig. 2). Based on values of discriminants Δ^m_o, the number of PAL-based blocks, which are necessary for implementation of the multi-output function, can be calculated. This number is equal to $\delta_f = \sum_{o=1}^{m} \left(\left\lceil \frac{\Delta^m_o - k}{k-1} \right\rceil + 1 \right)$ and, for most cases, is greater than δ^1_f.

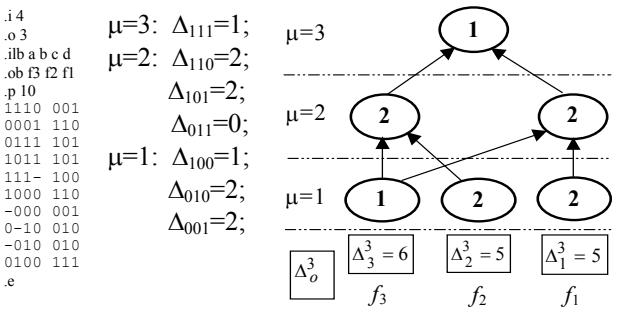


Fig. 2 Representation of the minimised function $f:B^4 \rightarrow B^3$ by means of the graph of outputs

Nodes of the graph correspond to the number of multi-output implicants and the last ones are, in their turn, associated with one of the multi-output vectors. For example, when a node of the μ^{th} range belongs to the graph and for that node $\Delta_y=k$, implementation of *k* implicants constituting common resources of the μ functions, is possible within the one block. Selection of the node leads to transformation of the graph and corresponding reduction of the Δ^m_o coefficients (Fig. 3). As a result, the selection of a certain node introduces feedback loops, which is shown on the reduced graph by means of implementation of the fourth-range nodes marked on the graph by means of the dashed line.

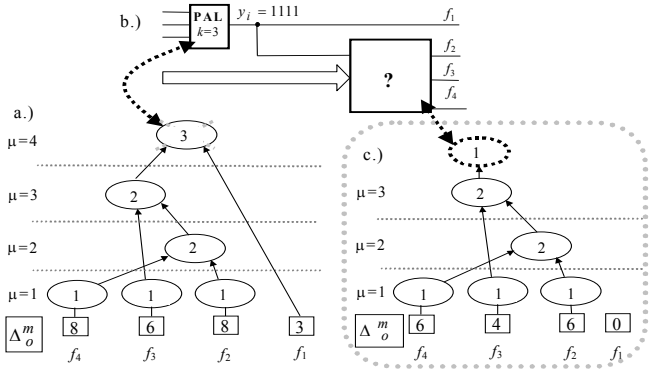


Fig. 3. a.) The graph of an exemplary function $f:B^n \rightarrow B^4$; b) Implementation of the implicants defined by the fourth-range node; c) The graph after reduction

3. SELECTION OF A NODE OF THE GRAPH

Let ${}^i\Delta_y$ be the discriminants that correspond to the node, which is chosen during i^{th} step of the algorithm of implementation of the multi-output function. Implementation of the group of implicants which correspond to the selected node ${}^i\Delta_y$, may lead to minimization of the number of used PAL-based blocks consisting of *k* terms, if the requirement ${}^i\delta_f - {}^{i+1}\delta_f > \lceil ({}^i\Delta_y - k)/(k-1) \rceil + 1$ is met. Since selection of the node ${}^i\Delta_y$ affects $\mu({}^i\Delta_y)$ discriminants ${}^i\Delta^m_o$, the condition for minimization of the PAL-based logic blocks (after having the discriminants re-ordered in such a way, that the selected node affects the consecutive ${}^i\Delta_j^{\mu({}^i\Delta_y)}$ discriminants), can be shown in the following form:

$$\sum_{j=1}^{\mu({}^i\Delta_y)} \left(\left\lceil \frac{{}^i\Delta_j^{\mu({}^i\Delta_y)} - k}{k-1} \right\rceil + 1 \right) - \sum_{j=1}^{\mu({}^i\Delta_y)} \left(\left\lceil \frac{{}^{i+1}\Delta_j^{\mu({}^i\Delta_y)} - k}{k-1} \right\rceil + 1 \right) > \left\lceil \frac{{}^i\Delta_y - k}{k-1} \right\rceil + 1$$

Let $r_j^{\mu({}^i\Delta_y)}$ be numbers calculated from the congruence:

$${}^i\Delta_j^{\mu({}^i\Delta_y)} - 1 \equiv r_j^{\mu({}^i\Delta_y)} \pmod{(k-1)} \quad \text{where } j=1,2,\ldots, \mu({}^i\Delta_y).$$

The essence of the theorem presented in [4] consists in selection of the nodes (multi-output implicants) that can be shared by several single-output functions. The theorem serves as a background to draw up an algorithm for multi-level implementation of multi-output functions by means of PAL-based logic blocks. As number of logic blocks δ_f is generally greater than the value of δ^1_f the essence the algorithm proposed consists in analysis of the graph nodes and searching for the nodes that are associated with possible large groups of common implicants. Let us assume that the graph of outputs describes multi-output implicants that are accomplished by means of ${}^i\delta_f$ PAL-based logic blocks with *k* terms. Selection, at the i^{th} step of iteration, the ${}^i\Delta_y$ node of the graph implies utilization of ${}^i\gamma = \lceil ({}^i\Delta_y - k)/(k-1) \rceil + 1$ PAL-based logic blocks. This leads to reduction of the graph of outputs, and the graph, after reduction, describes the implicants that are covered by ${}^{i+1}\delta_f$ PAL-based logic blocks. The higher is the value of the expression ${}^1\delta_f - ({}^{i+1}\delta_f + {}^i\gamma)$ the better is selection of the node in question. The rules for selection of the node can be deducted directly from the theorem [4] and can be listed in the following way:

Firstly, at the very beginning, one has to choose the ${}^{i}\Delta_{y}$ node, for which $\mu({}^{i}\Delta_{y})$=max

1. From the nodes of the same range, further selection must be carried out depending on values of discriminants:

2a. *if there exist nodes, for which* ${}^{i}\Delta_{y} \geq k$

- the node, for which the discriminant ${}^{i}\Delta_{y}$=max

2b. *if values of all the discriminants are lower that k*

- the node, for which within the set of remainders $\mathbf{R} = \{r_j^{\mu({}^{i}\Delta_y)};\ j \in <1, \mu({}^{i}\Delta_y)>\}$ there exists the maximum number of remainders $r_x^{\mu({}^{i}\Delta_y)}$ that meet the condition $0 < r_x^{\mu({}^{i}\Delta_y)} < {}^{i}\Delta_y < k$

The above rules serve as the basis for the algorithm of multi-level synthesis implemented in PALDec system.

Example:

Let us consider the function $f:B^5 \rightarrow B^4$ which, after minimization (Espresso) can be depicted in the file *f.pla* (Fig. 4.a).The reduced graph of outputs, associated to thereof, is shown in Figure 4.b.

Direct realization of implicants with PAL-based logic blocks that contain 3 terms each (by means of the classical method, after minimization the multi-output function) needs of ${}^{0}\delta_f$=12 PAL-based logic blocks. During the first step of the proposed algorithm, the implicants associated with the node Δ_{1111}=3 are realized. This step involves only one PAL-based logic block (${}^{1}\gamma$=1) and leads to significant lowering of the number of blocks that are necessary for direct realization of implicants depicted by the reduced graph. That graph is obtained after removing the node Δ_{1111}=3 and introducing an additional node, which is connected with the node Δ_{1110} and represents the feedback loop (Fig. 4.c). (${}^{1}\delta_f + {}^{1}\gamma = 3+2+3+0+1=9 \leq {}^{0}\delta_f$) The implicants that correspond to the nodes Δ_{1110}, Δ_{1010} (Fig. 4. c,d) are realized during the following steps of circuit synthesis. Realization of those implicants leads to reduction of the required PAL-based blocks number (${}^{2}\delta_f + {}^{2}\gamma = 2+1+2+0+1=6 \leq {}^{1}\delta_f$; ${}^{3}\delta_f + {}^{3}\gamma = 1+1+1+0+1=4 \leq {}^{2}\delta_f$). The nodes that correspond to the 1st range (Fig. 4.e) are implemented at the last stage of the synthesis process. The final circuit representation that uses $\delta_f = 6 < \delta^1{}_f = 12$ PAL-based logic blocks is shown in Figure 4.f.

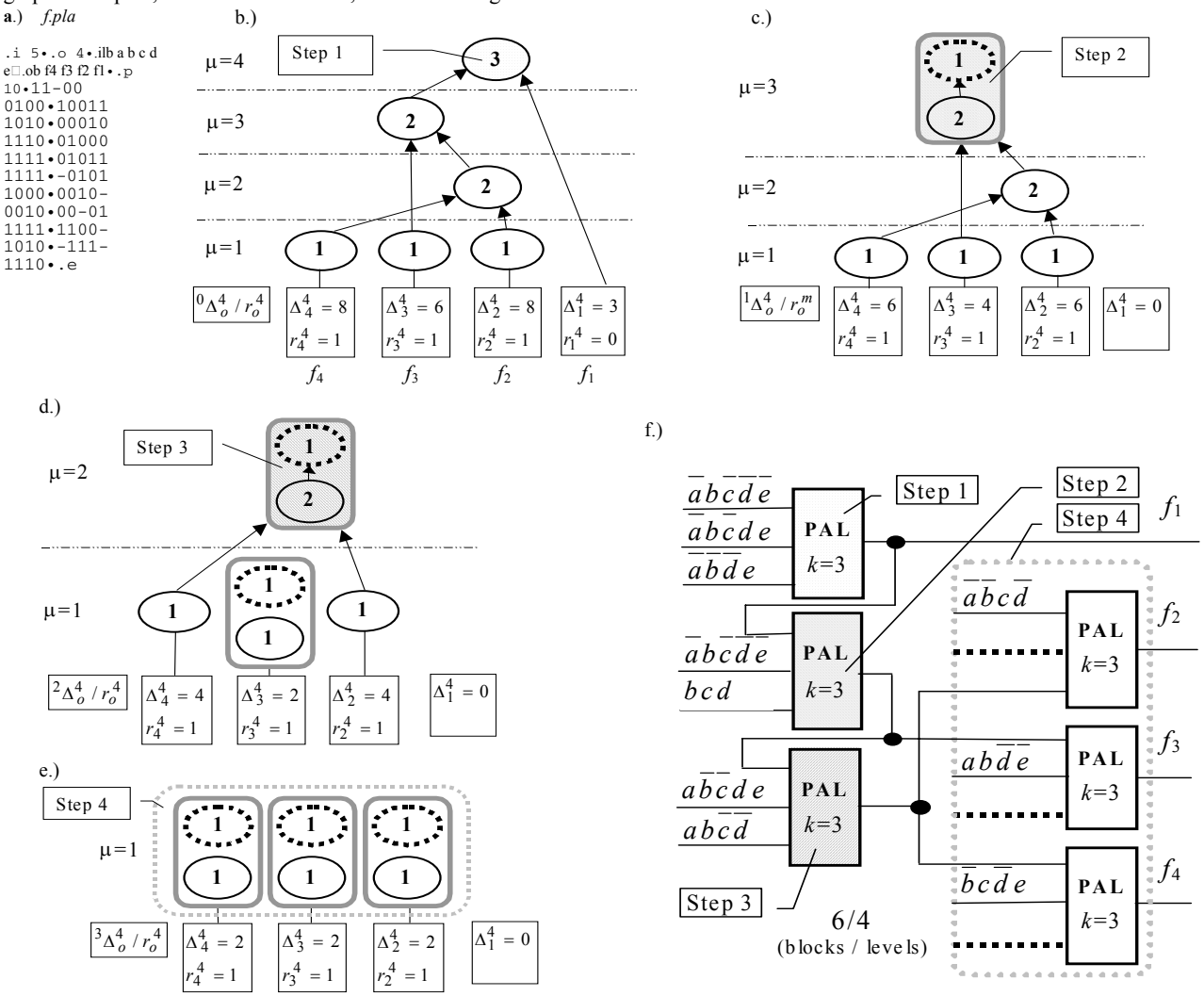


Fig. 4 Realization of the exemplary function $f:B^5 \rightarrow B^4$ a) description of the function by the *f.pla* file; b) reduced graph of outputs, c,d,e) graphs of outputs that correspond to successive steps of the synthesis process, f) final structure of the circuit

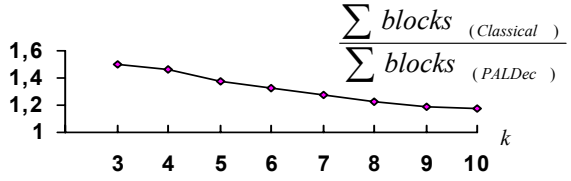
Table 1. Results of benchmark synthesis for the PAL-based logic blocks with *k*-terms

Classical – **C**lassical approach, PALDec – proposed **M**ulti-level synthesis based on analysis of nodes of the graph of outputs

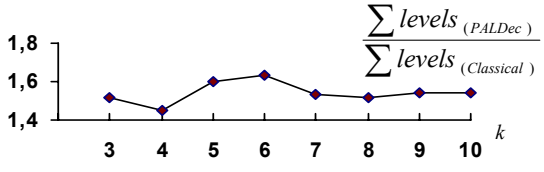
	Classical																PALDec													
	k= 3		k= 4		k= 5		k= 6		k= 7		K= 8		k= 9		k= 10		k= 3		K= 4		k= 5		k= 6		k= 7		k= 8		k= 9		k= 10	
	B	L	B	L	B	L	B	L	B	L	B	L	B	L	B	L	B	L	B	L	B	L	B	L	B	L	B	L	B	L	B	L
alu4	315	5	211	4	159	4	128	3	107	3	91	3	82	3	72	3	290	8	193	6	148	6	119	5	98	5	85	4	77	4	67	4
clip	73	4	49	3	38	3	30	3	26	2	22	2	21	2	20	2	66	5	45	4	35	4	29	4	26	3	23	2	21	2	20	2
duke2	99	3	72	3	61	2	49	2	44	2	41	2	39	2	36	2	100	4	76	4	64	3	55	3	49	2	46	2	43	2	40	2
misex3	612	5	410	4	309	4	249	3	208	3	178	3	157	3	143	3	450	8	316	7	257	7	219	6	189	6	170	6	153	4	136	4
rd73	70	4	47	3	36	3	29	3	24	3	20	2	19	2	16	2	63	6	44	5	32	4	26	4	23	3	19	3	18	3	16	3
rd84	142	5	95	4	72	4	58	3	49	3	42	3	28	2	21	2	130	8	87	6	67	5	54	4	45	4	40	4	27	3	20	3
sao2	36	3	24	3	19	2	15	2	14	2	11	2	11	2	10	2	29	5	20	4	16	4	13	4	11	3	10	3	9	3	9	3
seq	691	5	469	4	355	3	287	3	244	3	212	3	188	3	172	3	295	8	208	7	165	7	142	6	129	5	114	6	109	6	105	6
spla	425	6	293	5	227	4	188	4	163	3	144	3	128	3	120	3	285	6	190	4	158	4	137	4	124	4	115	4	105	4	97	3
table3	264	4	181	4	135	3	110	3	94	3	80	3	71	2	65	2	154	8	110	7	95	6	80	6	73	6	70	5	63	4	63	5
table5	272	4	183	4	141	3	112	3	95	3	82	3	74	2	65	2	143	7	104	6	87	6	73	6	69	5	64	5	61	5	56	5
Σ	2999	48	2034	41	1552	35	1255	32	1068	30	923	29	818	26	740	26	2005	73	1393	60	1124	56	947	52	836	46	756	44	686	40	629	40

4. EXPERIMENTAL RESULTS

Table 1 shows synthesis results for the benchmarks using two methods. Individual columns contain number of PAL-based logic blocks that have *k* terms, which are necessary for implementation of the appropriate benchmark (columns marked B) as well as numbers of logic levels (columns marked L). Within the set of 88 experiments for which results of minimization had been compared for the both methods, the proposed approach (PALDec) led to 75 (85%) solutions that used less number of logic blocks than the classical technique (Classical). Significant discrepancies can be observed if the logic blocks with low number of terms have been exploited. Having compared the overall number of logic blocks that are used by all the benchmarks, one can observe that the PALDec is more efficient if the PAL-based logic blocks with less number of terms are used (k=3,4,5). The above observation seems to be intuitively obvious, as smaller groups of multi–output implicants that have the same output part, more frequently are obtained as a result of minimization. The common coverage of such implicants brings more benefits if the structures that contain small PAL-based logic blocks are in use. Disadvantageously, all the circuits that were examined by means of the classical method proved to contain less or equal number of logic levels as compared to the PALDec. The results of experiments are presented in a synthetic way on Graphs 1 and 2. The values represented on the axis of ordinates in Graph 1 were calculated from the rational formula shown on the graph. $\Sigma blocks_{(\text{classical})}$ and $\Sigma blocks_{(\text{PALDec})}$ denote the relevant total sums of block counts obtained using the corresponding synthesis methods and presented in Table 1.



Graph 1. A comparison of the PALDec with the classical method with respect to logic blocks



Graph 2. A comparison of the PALDec with the classical method with respect to logic levels

The values represented on Graph 2 were calculated in a similar manner. Analysis of the benchmarks allows us to state, that in most cases reduction of block counts by using the new algorithm is obtained at the expense of certain expansion of levels. The proposed method is especially efficient, if $k = 3$ or 4. In this case a significant reduction of block counts, while preserving a comparable expansion number of logic levels, was observed. The rate of total synthesis time for the circuits under experiments, calculated for the PALDec and for the classical approach is about 1.25. The obtained results also have been compared to firmware tools [4].

5. CONCLUSION

The presented method is an alternative to the classical approach based on two-level minimization of individual single-output functions. The essence of the proposed method is to search for implicants that can be shared by several functions. Subsequent steps of the synthesis process are adapted to logical resources of PAL-based CPLDs. Adjusting elements of multi-level synthesis to logical resources characteristic for a PAL-based logic block allows for significant improvement of synthesis effectiveness in relation to the classical approach. If compared with another multi-level synthesis methods, the proposed algorithm seems to be especially advantageous, because it leads to minimization of the number of logic blocks in the synthesised structures. Simplicity of the algorithms that are based on graph analyzing methods and are useful for multi-level synthesis of multi-output logic functions within the PAL-based structures results in their applicability as an alternative approach instead of the other methods.

6. REFERENCES

[1] Anderson J. H., Brown S. D., Technology mapping for large complex PLDs, *Proceedings of Design Automation Conference, DAC'98, 15-19 January*, 1998, 698 -703

[2] Chen S., Hwang T., Liu C., A technology mapping algorithm for CPLD architectures, *IEEE International. Conference on Field - Programmable Technology, Hong Kong*, 2002, 204-210

[3] Kania D., Two-level logic synthesis on PALs, *Electronics Letters*, Vol.35, No. 11, 1999, 879-880

[4] Kania D., *The Logic Synthesis for the PAL-based CPLDs*, Wydawnictwo Politechniki Slaskiej, nr 14, Gliwice 2004 (*in polish*)

[5] Micheli G, *Synthesis and optimization of digital circuits*, McGraw-Hill International Editors, USA 1994