A New Technology Mapping for CPLD under the time constraint

Jae-Jin Kim*°, Hi-Seok Kim*, Chi-Ho Lin**

Dept. of Electronic Engineering, Chongju University*

Dept. Computer Science, Semyung University**

Dept. of Electronic Engineering, Chongju University, Naedok-dong

Sangdang-gu Chongju-shi, Korea 360-764

Tel : +82-43-229-8452 Fax : 82-43-229-8461

kimjj@chongju.ac.kr*°, khs8391@chongju.ac.kr*, ich410@venus.semyung.ac.kr**

Abstract

In this paper, we proposed a new technology mapping algorithm for CPLD under the time constraint(TMCPLD-II). In our technology mapping algorithm, we generate the feasible clusters from a given Boolean. The generated feasible clusters create clusters with minimum area under the time constraint. A covered Boolean network is transformed to a Boolean equation. The transformed equations are reconstructed in order to fit to an architecture of selected target CPLD by using collapsing and bin-packing.

To demonstrate the efficiency of our approach, we applied our algorithm to MCNC benchmarks and compared the results with those of the existing algorithms. The experimental results show that our approach is better than any of the existing algorithms in the number of logic blocks.

Keywords: technology mapping for CPLD, time constraint, number of multi-level, feasible cluster

Ⅰ. Introduction

FPGAs and CPLDs are widely used for the implementation of digital circuit. Most FPGAs have logic blocks based on look-up-tables(LUTs), and some have multiplexer-based logic blocks. But CPLDs have a different style block like a PLA-style logic block.[1][2]

In recent years, many LUT-based technology mapping algorithms have been proposed. Some of these algorithms focus on minimizing either area or depth.[3][4] By contrast, very little has been published on technology mapping for CPLDs. The algorithms that published for CPLD technology mapping did not consider the time constraint[3][4] and used the logic blocks of CPLDs ineffectively.

We proposed a new technology mapping algorithm, called *TMCPLD-II* (TEchnology Mapping for CPLD), that can implement the resulting circuit in order to be fit for CPLDs under the time constraint and minimized the number of logic blocks in comparing to the results of other CPLD technology mapping algorithm.

* This work was supported by the RRC program of MOST and KOSEF.
* This work was supported by IDEC(IC Design Education Center).

II. Background and related work

CPLDs consist of several logic blocks because they typically have a large number of inputs, and hence can realize a large number of different logic functions. Since a PLA-style block contains a programmable AND-array followed by an OR-array, the traditional approach for synthesis of circuits for implementation in such blocks uses two-level minimization. Most CPLDs consist of a large number of PLA-style blocks, with available logic blocks up to about 100 blocks. Such a chips have not been implemented well to the large circuits only using two-level minimization method. Therefore, recent trend for synthesis using CPLDs is to make use of multi-level synthesis.

The combinational part of a digital circuit can be described by a directed acyclic graph (DAG). Each node in the DAG represents a single logic function in the circuit; the edges in the DAG represent dependencies between logic functions. The logic function associated with a node in DAG can be represented in sum-of-products (SOP) form.

In order to perform technology mapping for CPLDs, resulting circuit should be transformed to a forest of fanout free tree from the circuit's DAG. In the case of the nodes within the DAG that have an outdegree greater than one, existing technology mapping algorithm absolutely divided circuit's DAG into a forest of fanout free tree by partitioning the nodes that contained in DAG.[9]

However this method only considered the nodes with property of the feasible node size: input I and product P. However the number of OR terms within the logic blocks for CPLDs have a limited capacity to implement a combinational logic function. Without considering the number of OR terms, k , within a single logic block, in some case, the use of number of logic blocks have to be increased. Because, when the number of OR terms stored at each node within feasible subtree are considerably less than a capacity of the number of OR terms in a single logic block, a node implementing a combinational logic function can't be fit for a single logic block for CPLDs.

111. Problem Statement

In a combinational circuit, we can represent it by a Boolean network. A Boolean network, $G=(V, E)$, is a circuit's DAG . As each node, $v \in V$, represents a single-output logic function. An edge $(u, v) \in E$, is created and added to edge set E .A set of node in V with zero out-degree represents primary output(PO), and zero in-degree represents primary input(PI). A Boolean network, $G=(V, E)$, can be partitioned into the subnetwork(subgraph). A connected subgraph $C=(V', E')$, where $V' \subset V$ and $E' \subset E$. A subgraph, C, is a *cluster* A cluster, C, also represents the combinational logic function created from merging all of the nodes within each cluster C. A feasible cluster is a node implementing a single logic function with property that it can be mapped to one of a single logic block. A cluster C is trivial if it contains only a single node within V'. A node is kernel if out-degree of the node is two or more.

For example, in Fig. 1(a), C9...C17 are feasible clusters with trivial (C12), and C10 is kernel.

The remaining clusters C9, C10,.., C12 are also feasible clusters.

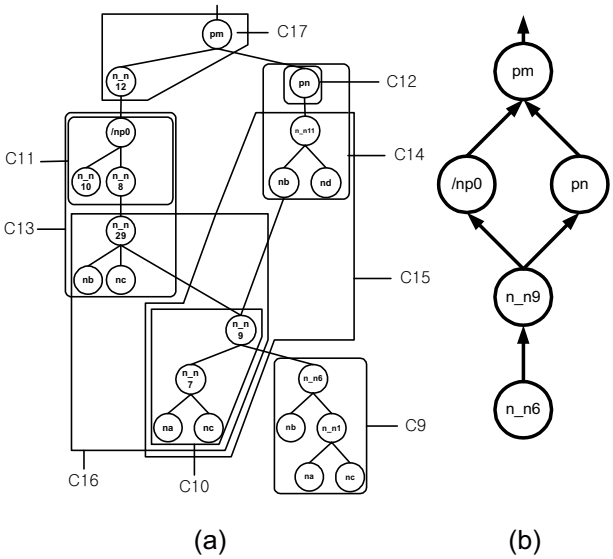

Fig. 1. An example of technology mapping

(a) all of feasible clusters (b) a minimum mapping result.

When we attempted to select the feasible clusters that have a minimum number of multi-level and mapped a least number of logic blocks among all the feasible clusters, C9, C10... C17, showed in figure 1(a), the minimum feasible clusters is implemented as a new

feasible cluster in mapping solution. This process continues until these feasible clusters have only primary input node. The results showed in figure1(b) that four feasible clusters are remained in a fanout-free tree.

To get the minimum mapping results, it is traversed in bottom up topological order. As each cluster is visited in turn, the number of logic blocks is needed to implement a logic function stored at each feasible cluster, and then the delay time of total logic blocks is computed. For example, when a cluster, C10, is visited, a cluster, C10, contained other feasible clusters i.e. C9, C10, C13, C14, and C17, are mapped to each logic block respectively. Therefore, five logic blocks are needed to implement a logic function. Hence, the number of multi-level of logic blocks is set to 4 and total delay time of the logic blocks set to 4T(delay time of logic block) in a critical path of a fanout free tree. The mapping results to each cluster shown in Table I.

The first column in the table lists shows the selected feasible clusters by removing the clusters that are trivial. The second column shows the mapping results implemented to the logic blocks and the mapping results of feasible clusters stored at each selected feasible cluster. The third column shows the number of logic blocks needed to implement for each cluster. The fourth column shows the number of logic block level. The fifth column shows the results of the delay time.

If one cluster, C10, is selected in Table 1, we can find the optimum solution; minimum number of logic blocks and delay time within a forest of fanout free tree required to implement a single logic function, shown in figure 1.

Table I. A mapping result

Selected Cluster	Total mapping results by logic block	Number of logic block	Number of level	Delay time T : delay time of logic block
C9	C9, C11, C12, C15, C16, C17	6	4	4T
C10	C9, C10, C13, C14, C17	5	4	4T
C11	C9, C10, C12, C15, C16, C17	6	4	4T
C12	C9, C11, C12, C14, C16, C17	6	4	4T
C13	C9, C10, C12, C13, C15, C17	6	4	4T
C14	C9, C10, C11, C14, C16, C17	6	4	4T
C15	C9, C11, C12, C15, C16, C17	6	4	4T
C16	C9, C11, C12, C14, C16, C17	6	4	4T

Therefore, it is necessary to consider the requirements of technology mapping in a Boolean network under the time constraint. We are defined some conditions as following.

Definition 1: A set of feasible clusters, *S={C1, ... Cn}* is a covering candidate set of a Boolean network *G=(V, E),* it is satisfied with the following two types of constraint.

number of multi-level constraint : The number of multi-level is defined by considering the time constraint and delay time of a given CPLD. The equation of multi-level is shown in Eq. 1.

$$Number\ of\ multi-level(mn) = \frac{Time\ constraint}{Delay\ time\ of\ a\ logic\ block} \quad \text{(Eq.1)}$$

$$Delay\ time\ of\ a\ logic\ block = T_{CO} + T_{GA} + T_{RO}$$

$T_{CO} = Combinatorial\ delay\ time$
$T_{GA} = Clock\ skew$
$T_{RO} = Register\ output\ delay\ time$

the number of OR terms : In order to map for a given CPLD efficiently, the number of OR terms, k, have to be considered to make a feasible cluster in a forest of fanout free tree.

In a Boolean network, our goal is to find a covering candidate set of clusters under the time constraint. This approach to synthesis circuits for logic blocks that represents several techniques. We use multi-level logic optimization and traditional technology mapping as a first step. Then, since it is not feasible to directly use traditional technology mapping for CPLD architectures, as discussed below, we collapse the resulting circuit to make efficient use of logic blocks. We describe our algorithm in detail in the next section.

Ⅳ. Technology Mapping algorithm for CPLD under the time constraint (TMCPLD-II)

4.1 Phase Ⅰ : Feasible Cluster Generation

In a Boolean network, the first phase of TMCPLD-II generates a FC (set of feasible clusters). This phase reduces the number of logic blocks by selecting a feasible cluster that is a node to be collapsed into its fanout nodes and covered all of the nodes within a feasible cluster. After performing the graph covering, a Boolean network is transformed to a forest of fanout free tree by replicating the nodes that out-degree more than one.

To select the highest priority of a feasible cluster in a circuit's DAG, it is traversed in top-down manner, the algorithm proceeds to find the set, S(n), of all nodes that contains the number of node replication rooted at node n, the highest priority of a feasible cluster is decided for each node that has a maximum number of node replication in a circuit's DAG.

The initial cost is defined the number of OR terms, k , within a node. And the total cost is defined the number of OR terms within a cluster. Initial and total costs are computed for each node of fanout free tree. An equation for total cost calculation is shown in Eq. 2.

$$Cost(node) = \begin{cases} \prod(Child\_Node\_Cost(node), & Node\_Cost = 1 \\ \sum(Child\_Node\_Cost(node), & Node\_Cost \geq 2 \end{cases} \quad \text{(Eq. 2)}$$

4.2 Phase Ⅱ : Graph Covering with Feasible Cluster

The second phase finds a set of the feasible clusters, S ,covering the Boolean network to meet the requirements of the time constraint in a set of the feasible clusters, FC, created from the previous phase.

Our proposed covering algorithm is basically a dynamic programming with a look-ahead feature, mapping proceeds by collapsing all of the nodes in a fanout free tree.

Procedure **BNTC_Cover**(NFC_MN, MN, N, K, FC)**begin**

subgraph_Level = the number of level for a set of all clusters covering the critical path in a fanout free tree

Subgraph_Level = 0;

foreach v∈ PI of N **do**
Re(v) = {}
End

foreach a set of all clusters, FC, in a critical path with a maximum depth **do**
foreach a cluster in a topological order **do**
while(MN ≥ Subgraph_Level){
while(k ≥ FC_cost){
Re(v) = Re(v)∪ FC(v);
}
Subgraph_Level ++;
}
Re(v)=***select_best_FC(v)***
End
End

end BNTC_Cover

Fig. 2. An overview of Boolean Network covering algorithm under the time constraint

In figure 2, *Re(v)* represents the minimum number of feasible clusters need to cover all of the nodes in a fanout free considering a requirement of the time constraint. When a node, n, is visited in a topological order from the primary input nodes, a feasible cluster contained at node n that have a certain value of the number of OR terms described as FC_cost. If FC_cost is less than or equal to the limited capacity of the number of OR terms contained in a basic logic block for CPLDs. Mapping continues by expanding a node, implementing a new feasible cluster at which point Re(v) is specified according to the union set operation of feasible clusters. Hence, Re(v) has a new FC_cost. This procedure repeated until k is greater or equal to FC_cost. The select_best_FC procedure selects the minimum number of feasible clusters in the set of feasible cluster, FC(v).Therefore, the results of selected FC are stored in *Re(v)*.

Subgraph_Level that is the number of level for a set of clusters, FC, in a critical path. This Subgraph_Level repeated until less than equal to the predefined the number of multi-level(mn).

In order to select the best result of feasible cluster, we should consider the delay time constraint and *the number of multi-level(mn)* of logic block in the fanout free tree. The number of block level

proceeds as much as *the number of multi-level(mn)*-2 defined by (Eq. 1). If the number of multi-level of clusters are greater than *the number of multi-level(mn)*, previous technology mapping methods [4][5][6][7][8][9] can not be performed since these methods can not consider the time constraint. But our proposed technology mapping algorithm successfully proceeds under the time constraint in a fanout free tree.

For example, we assume that *the number of multi-level(mn)* is set to 5 ,and the number of multi-level of feasible cluster set to 7, our technology mapping begins at each cluster that only contains primary input nodes. This process repeated until the *number of multi-level(mn)* become to 3.Hence, the remaining feasible clusters that is not mapping to the logic blocks in fanout free tree also repeated until *the number of multi-level(mn)* become to 2 through the procedure of collapsing and bin packing.

As a result, our proposed technology mapping method reduced the number of logic block in comparing with other technology mapping methods.

4.3 Phase Ⅲ : Collapsing

A feasible cluster containing the nodes that can be collapsed into their fanout nodes represents to a single Boolean logic equation. This is accomplished by exploiting the selected clusters. The selected clusters rooted at their fanout nodes. To identify nodes in the selected clusters that are inputs to newly created node and adding them to fanout node set. Therefore, A logic function stored at each node can be represented to AND-OR two level Boolean equation.

4.4 Phase Ⅳ : Bin-packing

The final phase of TMCPLD-II packs circuit nodes into the multi-output blocks available in the target architecture. This is accomplished by using a modified first-fit-decreasing bin packing algorithm that attempts to maximize the number of OR-term that are packed into the logic block.

Phase Ⅰ and Ⅱ, Ⅲ, Ⅳ of TMCPLD-II can be used to create nodes that can be fitted efficiently into MACH4 logic blocks by setting product term equal to 20.[3][9] The nodes can be packed into the MACH4 logic blocks using a slight modified version of phase Ⅳ of TMCPLD-II. Hence, we believe that relatively few changes to our algorithm would be required to it to efficiently target the MACH4 architecture.

Ⅳ. Experimental result

Our proposed algorithm has been implemented in the C language within the SIS[7] framework, allowing it to access the I/O routine and two-level minimization algorithms within SIS. To evaluate the quality of final mapping solutions produced by our algorithm, it was compared with the approach used in [8][9], called DDMAP and TEMPLA.

Table 2 shows the results when the technology mappers are used to map circuits into logic blocks with the number of OR-term.

The results for 12 circuit used in this study are shown in the table. Before applying TMCPLD-II, TMCPLD-I, TEMPLA or DDMAP, each of circuits was synthesized using the execution files in the SIS.

The first column in the table shows the name of each benchmark circuit. The second column shows the number of logic blocks needed to implement each circuit when DDMAP is used. The third column shows the number of logic blocks needed and running time to implement each circuit when TEMPLA is used. The fourth column shows the number of logic blocks needed and running time to implement each circuit when TMCPLD-I is used. The fifth column shows the number of logic blocks needed and running time to implement each circuit when TMCPLD-II is used.

On average, when the 12 circuits shown in Table 2 are mapped using DDMAP, they require 39.1% more blocks than when TMCPLD-II is used. In other case, mapped using TEMPLA, they require 86% more blocks than when TMCPLD-II is used.

Among 15 circuits used by experiment in [9], only 12 circuits are used in our experiments, because 3 benchmark circuits were not supported from SIS[7].

Table 2. Result comparison of the existing technology mapping and TMCPLD-II

	DDMAP	TEMPLA		TMCPLD-I		TMCPLD-II	
	block	block	running time (sec)	block	running time (sec)	block	running time (sec)
alu4	199	155	31.2	81	33	86	32.1
cps	159	120	20.4	119	22.1	119	20.9
apex4	193	193	38.2	139	50.6	142	40.3
misex3	214	154	29.3	147	32.8	149	30.2
ex5p	27	132	24.7	132	33.5	132	26.8
s38417	1208	603	498.8	479	502.7	498	500.1
seq	337	229	53.6	219	54.1	221	53.9
fir	1424	249	126.4	199	128.1	210	127
fsm8_8_13	58	49	5.6	49	5.8	49	5.8
pmac	911	237	129.4	232	131.3	233	130.1
psdes	301	151	37.8	119	39.4	120	38.2
sort	275	138	30.5	101	31.2	115	30.7

The complexity of our proposed algorithm is $O(n^2)$, and more greater than that of other technology mapping algorithms. Since the phase II in our algorithm is performed after creating FC. But our proposed technology mapping algorithm slightly more improved the execution time than that of TMCPLD-I[10].

V. Conclusion and future work

In this paper, we proposed a new CPLD technology mapping algorithm to minimize the number of logic blocks under the time constraint. In our algorithm, the technology mapping problem consist of four phases : feasible cluster generation, graph covering with feasible cluster, collapsing, bin-packing.

The experiments with MCNC benchmark circuits for logic synthesis show that the proposed technology mapping algorithm has reduced the logic blocks much more than that of the DDMAP and the TEMPLA.

In future work, we have to develop our algorithm for the purpose of a specific commercial CPLD and reduced the time complexity of the proposed algorithm. Our proposed algorithm will be loaded one of the routines in technology mapping for CPLDs called MyPLD(Seodu Logic Co.).

REFERENCES

[1] *The Altera Data Book*, Altera Corporation, 1996

[2] *ACT 1 Series FPGAs Data Sheet*, Actel Corporation, 1996

[3] *The MACH 4 Family Data Sheet*, Advanced Micro Devices, 1996

[4] J. Cong and Y. Ding, "FlowMap : An `Optimal Technology Mapping Algorithm for Delay Optimization in Lookup-Table Based FPGA Designs", *IEEE Transactions on Computer-Aided Design of Integrated Circuit and Systems*, Vol. 13, No. 1, January 1994, pp. 1-11

[5] R.J Francis, J. Rose and Z. Vranestic, "Chortle-crf : Fast Technology Mapping for Lookup Table-Based FPGAs", *28th ACM/IEEE Design Automation Conference*, June 1991, pp.227-233.

[6] R.J Francis, J. Rose and Z. Vranestic, "Technology Mapping of Lookup Table-Based FPGAs for Performance", *1991 IEEE Conference on Computer Aided Design*, pp. 568-571

[7] E. M. Sentovice et al., "SIS : A system for sequential Circuit Synthesis", *Technical Report UCM/ERL M92/41*, Electronics Research Laboratory, Department of Electrical Engineering and Computer Science, University of California, Berkeley, 1992

[8] J.L. Kouloheris, “Empirical Study of the Effect of cell Granularity on FPGA Density and Performance”, Ph.D. Thesis, Department of Electrical Engineering, Stanford University, 1993.

[9] Jason Helge Anderson, Stephen Dean Brown, "Technology Mapping for Large Complex PLDs", Design Automation Conference, 1998, pp. 698-703

[10] Hi-Seok. Kim, Jae-Jin. Kim, Chi-Ho. Lin, “ An Efficient CPLD Technology Mapping Under Time Constraint”, IEEE Conference on Microelectronics, 2000, pp. 265-268.