第 23 卷第 11 期　　半 导 体 学 报　　Vol. 23, No. 11
2002 年 11 月　　CHINESE JOURNAL OF SEMICONDUCTORS　　Nov., 2002

Graph Clustering Algorithm for RT Level ALU Technology Mapping*

Zhou Haifeng, Lin Zhenghui and Cao Wei

(*LSI Research Institute, Shanghai Jiao Tong University, Shanghai* 200030, *China*)

Abstract: Register-transfer level mapping (RTLM) algorithm for technology mapping at RT level is presented, which supports current design methodologies using high-level design and design reuse. The mapping rules implement a source ALU using target ALU. The source ALUs and the target ALUs are all represented by the general ALUs and the mapping rules are applied in the algorithm. The mapping rules are described in a table fashion. The graph-clustering algorithm is a branch and bound algorithm based on the graph formulation of the mapping algorithm. The mapping algorithm suits well mapping of regularly structured data-path. Comparisons are made between the experimental results generated by 1-greedy algorithm and graphclustering algorithm, showing the feasibility of presented algorithm.

Key words: high-level synthesis; technology mapping; register-transfer level; arithmetic logic units; graphclustering algorithm

EEACC: 1210

CLC number: TN47　　**Document Code**: A　　**Article ID**: 0253-4177(2002)11-1162-06

1 Introduction

Current VLSI designs have reached complexities of millions of gates; they are expected to be grown even higher in the coming years. Systems of such complexity are very difficult to design by handcrafting each transistor or by defining each signal for logic gates since human designers can not do an effective job for problems involving huge number of objects. For systems of such complexity, even the traditional design automation tools such as logic synthesis and physical design tools fail to provide good solutions in reasonable amount of time. There is a need to develop design methodologies that can handle systems of higher complexity. So, the idea of high-level design[1] and design reuse[2] came into being.

Technology mapping is a very important step in digital system design, which transforms the technology-independent structure description to the technology-dependent library. Based on the complexity of cells used in mapping, the library mapping could be categorized into different levels. At the logic level, library-mapping implements a logic level design using logic level cells from a library. IBM[3] used a heuristic algorithm in logic synthesis. Structural (tree-based), Boolean matching techniques are applied in DAGON[4] and MIS[5]: OBDD[6] is also proposed to solve this prob-

* Project supported by Natural Science Foundation, USA (No. 9602485) and Univercity Doctoral Point Foundation of Ministry of Education, PR China

Zhou Haifeng male, was born in 1976, PhD candidate. His research is on high-level synthesis in electronic design automation.

Lin Zhenghui male, was born in 1933, professor, director of PhD candidates. His research is in theory of circuit and system, electronic design automation.

Cao Wei male, was born in 1973, PhD. His research is on high-level synthesis and system.

Received 7 March 2002, revised manuscript received 4 June 2002

lem. In the system level, mapping can be performed between system-level components such as processors, memories and interface units. MICON[7] is a system that tries to reuse off-the-shelf system-level parts such as processors, memories and peripherals to build a single-board computer system. Jha[8] presented a rule-based algorithm for technology mapping in high-level synthesis. Similarly, library mapping could be applied at RT level through RT level design.

The output of high-level synthesis includes datapath netlist in RTL and state transition graph. However, module compiler and logic synthesis tools can not effectively reuse the data book libraries of higher-level components commonly used in design practices. In this paper, we present register-transfer level mapping (RTLM) to implement the RTL datapath using technology dependent RTL library components.

2 Problem description

According to Ref. [8], we defined the following RTLM terms: a source component (S), a target component (T) and a set of mapping rules (R). In order to describe source and target components, each component is described in terms of a set of RT-functions that are defined using a canonical representation of the component. A rule in R describes an alternative for implementing a function in S using a function in T; each source function can be potentially implemented by different target functions. The task of RTLM, then, reduces to selecting a set of rules, one for each function in source component S, that realizes the best mapping of S on T with respect to the cost function (e.g., area and delay).

2.1 Assumptions

(1) All data and operation use 2′s complement representation.

(2) A source component can only perform one function at a time. For example, a comparator can implement several RT functions (e.g., EQ, NEQ, GT, LT, etc.), but only one function is performed at a time.

(3) We restrict ourselves to arithmetic, logic and comparison functions.

(4) All RT functions are defined by using canonical functions or their simples.

(5) S and T should have the same bit-width.

2.2 General ALU representation and canonical ALU functions

To have a reference model for RTLM, we defined a universal ALU model (in Fig. 1) to perform a set of ALU canonical functions: five arithmetic functions (ADD, SUB, RSUB, INC, DEC), 16 logic functions (all Boolean functions of two variables), and 6 comparison functions (EQ, NEQ, GT, GEQ, LT, LEQ). So we can generate total 27 canonical ALU functions. Any other ALU function can be generated by using these canonical functions. Note that not every ALU should use all of the ALU ports. Figure 1 shows a universal ALU, which has the following ports:

In0: Primary left input.
In1: Primary right input.
CaIn: Carry input.
Out0: Primary output.
Out1: Secondary output.
CaOut: Carry output.
Con: Control input.

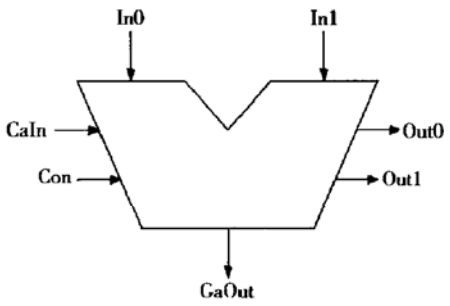

Fig. 1 General ALU representation

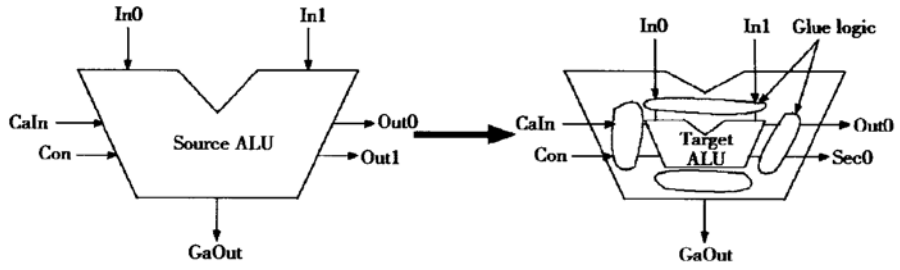

Fig. 2 RTLM of an ALU

2.3 Description of library components

The representation of S and T should not only grasp the function of ALU, but also facilitate the comparison of them. The method based on canonical function supports these demands. The function of the library components is composed of these 27 canonical ALU functions, so the component could be described through the Boolean relation between the ports of the library component and the ports of the universal component. The Boolean relations between the component ports are described in a tabular fashion.

3 Description of mapping rule

A mapping rule describes how to implement a canonical function by another canonical function so that the cost function is optimal. We use SF and TF to stand for the source canonical function and target canonical function, and use CS and CT to stand for source component and target component. The mapping rule is described through the mapping of the ports of CS and CT so that SF can be implemented using TF. Table 1 lists some of the rules. Each row in the table gives the name of the rule, source function, target function and port mapping information. The input and output entries indicate the connectivity and the additional logic required to implement the mapping rule. For example, the fourth rule SA in the table implements the source function SUB with the target function ADD by inverting the right input ($\overline{SIn1}$).

Table 1 Table representation of some mapping rules

Rule name	SF	TF	Input ports			Output ports		
			TIn0	TIn1	TCaIn	SOut0	SCaOut	SOut1
AA	ADD	ADD	SIn0	$\overline{SIn1}$	SCaIn	TOut0	TCaout	TOut1
AS	ADD	SUB	SIn0	$\overline{SIn1}$	SCaIn	TOut0	TCaout	TOut1
SS	SUB	SUB	SIn0	$\overline{SIn1}$	SCaIn	TOut0	TCaout	TOut1
SA	SUB	ADD	SIn0	$\overline{SIn1}$	SCaIn	TOut0	TCaout	TOut1
RA	RSUB	ADD	$\overline{SIn0}$	SIn1	SCaIn	TOut0	TCaout	TOut1
RS1	RSUB	SUB	$\overline{SIn0}$	$\overline{SIn1}$	SCaIn	TOut0	TCaout	TOut1
RS2	RSUB	SUB	$\overline{SIn1}$	SIn0	SCaIn	TOut0	TCaout	TOut1
IS	INC	SUB	SIn0	"1..1"	1'	TOut0	TCaout	TOut1
DA	DEC	ADD	SIn0	"0..1"	0'	TOut0	TCaout	TOut1
DS	DEC	SUB	SIn0	"1..1"	1'	$\overline{TOut0}$	TCaout	TOut1
ANAN	AND	AND	SIn0	SIn1	—	$\overline{Tout0}$	—	—
ANNA	AND	NAND	SIn0	SIn1	—	TOut0	—	—

From the table, we note that each source function could be implemented by several source functions using different rule. For example, ADD can be implemented using rule AA1 and AS1; SUB can be implemented using rule SS and SA. Note that not only can each source function have a variety of rule choices, but also the rule choices for each source function are interdependent. For example, if we

choose rule AS to implement source function ADD, rule SA to implement the source function SUB, we will generate a good design because two right inputs are all $\overline{SIn1}$ so that they can share the same input port. Thus a strategy is needed to select a mapping rule from multiple alternatives for each source function.

4 Mapping algorithm

The inputs of the mapping system are S, T, and R. The first step is to choose the target canonical component to realize the source component. Then the target canonical component is mapped to the feasible target component. The output of the system is an implementation of source component S with target component T plus some additional (glue) logic surrounding. We focus on the first step because the second step, such as matching of port names, is relatively more simple.

The graph clustering algorithm is a branch and bound algorithm based on the graph formulation of the mapping problem. We define a graph $G = (V, E)$, in which V denotes the set of vertices and E denotes the set of edges.

V: {v| v is a rule mapping an S to T}.

$p(v)$: represents the partial solution associated to node v.

E: {$e = (v_1, v_2)$ There is a possibility that the two sets of rules associated with v_1 and v_2 are used together}.

$w(e)$: Let $e = (v_1, v_2)$. Then $w(e)$ is defined to be the cost of the partial solution that is generated if v_1 and v_2 are combined.

Given the graph $G = (V, E)$, we define the following function to combine two connected v_1 and v_2, which generates a new vertex v and generates new edges which connect v and other vertices in the graph G. The algorithm is as follows.

Algorithm 1: Combination(v_1, v_2)
Create a new node v
$p(v) = p(v_1) \cup p(v_2)$
Add v to G.
Delete edge(v_1, v_2)
$\forall v_k | \exists (v_1, v_k)$ and (v_2, v_k)
Create an edge $e = (v, v_k)$
Calculate $w(e)$
Insert e in the edge list
Return v

Now we illustrate the clustering algorithm. We first build an initial graph by generating a set of nodes, one for each mapping rule. Then the algorithm connects each pair of nodes that do not map the same source function. The edges are sorted in the increasing order by their weight. From the initial solution, the algorithm selects the first edge, i.e. the one with the smallest edge weight, and merges the corresponding pair of vertices. This algorithm is repeated till we have a node representing a complete solution. The complete algorithm is shown in algorithm 2.

Algorithm 2: Graph clustering algorithm
INPUT: $f(S), f(T), r(ST)$
OUTPUT: A set of rules, one for each function in $f(S)$, with minimum cost
$\forall r \in f(ST)$ creates a node
$\forall (v_1, v_2) | f(v_1) \cap f(v_2) = \Phi$
Create an edge e
Calculate $w(e)$
Create edge-list by sorting all the edges
done= False
While not done
if (edge-list= Φ) then done= True; exit
Let $e = (v_1, v_2)$ be the first edge
v= Combine(v_1, v_2)
if v has one rule for each function in $f(S)$
done= True
Return $p(v)$
end While
Return Φ

The graph clustering algorithm is a united algorithm of branch-and-bound and dynamic programming algorithm. The algorithm guarantees to get the optimal result. In the worst case, we should consider all possible combination of source functions and mapping rules.

5 Experimental results

Considering the variety of ALUs of our RTLM, we chose a variety of source and target libraries for the experiment. They come from different libraries. The function and bit-width are varied. All the experiments are run on SUN Sparc-5 workstation. The experimental results are listed in Table 2.

Table 2 Experimental results of RTL ALU mapping

Experiment	Width	Source function	Target function	Cost	1-greedy algorithm[8]			Graph clustering algorithm		
					Area /GC	Delay ns	Run time /s	Area /GC	Delay /ns	Run time /s
1	16	ADD, SUB, RSUB	ADD, SUB	Area	432	42.1	1.36	421	44.7	10.9
				Delay	457	39.3	1.29	457	39.3	12.8
2	16	ADD, SUB, INC, DEC	ADD, SUB	Area	418	49.94	3.42	412	50.6	547.04
				Delay	436	42.4	3.56	436	42.4	607.26
3	32	ADD, INC, NEQ, GT	ADD, RSUB, AND, NOR	Area	638	71.06	19.88	628	73.7	1042.6
				Delay	687	61.7	19.56	673	63.4	1141.8
4	32	RSUB, OR, OR, AND	ADD, SUB	Area	704	58.68	32.14	694	59.3	3070.2
				Delay	708	51.9	33.02	692	55.6	3468.8

From table 2, we can find that the algorithm can get not only area optimal but also delay optimal design. Moreover, the run time of graphclustering algorithm is far longer than greedy algorithm, because the computation complexity of this algorithm is exponential with respect to numbers of source functions nS.

6 Conclusion

The methodology in this paper is applied in the VHB system developed by LSI research institute of Shanghai Jiaotong University. In this paper, we apply a graph clustering ALU RTLM approach. We can get both area optimal and delay optimal design by using this algorithm. With different library and different width, we design an experiment, the result of which proves the effectiveness of this algorithm.

References

[1] Gajski D, Dutt N, Wu A, et al. High-level synthesis: introduction to chip and system design. Kluwer Academic Publishers. 1992

[2] Jha Pradip K, Dutt N. Design reuse through high-level library mapping. ACM/IEEE International Conference on Computer-Aided Design, 1995: 345

[3] Darringer J, Joyner W, Berman C L, et al. Logic synthesis through local transformations. IBM J Res Develop, 1981, 25 (4): 272

[4] Keutzer K. DAGON: technology binding and local optimization by DAG matching. ACM/IEEE Design Automation Conference, 1987: 341

[5] Byayton R, Rudell R, Sangiovanni-Vincentelli A, et al. MIS: a multiple-level logic optimization system. IEEE Trans Comput-Aided Des, 1987, 6: 1062

[6] Mailbot F, De Micheli G. Algorithms for technology mapping based on binary decision diagrams and on Boolean operations. IEEE Trans Comput-Aided Des, 1993, 12: 599

[7] Birmingham W, Gupta A, Siewiorek D. The MICON system for computer design. ACM/IEEE Design Automation Conference, 1989: 135

[8] Jha P K, Dutt N. High-level library mapping for arithmetic components. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 1996, 4(2): 157

基于图聚集算法的寄存器传输级 ALU 工艺映射算法*

周海峰　林争辉　曹　炜

(上海交通大学大规模集成电路研究所，上海　200030)

摘要: 给出了寄存器传输级工艺映射(RTLM)算法, 该方法支持使用高层次综合和设计再利用的现代 VLSI 设计方法学, 允许复杂的 RT 级组件, 尤其是算术逻辑单元(ALU)在设计中重用. 首先提出了 ALU 的工艺映射问题, 给出了源组件和目标组件以及标准组件的定义, 在此基础上通过表格的方式给出映射规则的描述. 映射算法套用一定的映射规则用目标 ALU 组件来实现源 ALU 组件. 采用一种基于分支估界法的图聚集算法, 用该算法不仅可以产生面积最优的, 而且还可以产生延时最优的设计. 针对不同库的实验结果证明该算法对于规则结构的数据通路特别有效.

关键词: 高层次综合; 工艺映射; 寄存器传输级; 算术逻辑单元; 图聚集算法

EEACC: 1212

中图分类号: TN47　　**文献标识码**: A　　**文章编号**: 0253-4177(2002) 11-1162-06

* 美国国家科学基金(合同号: 9602485)和国家教育部博士点基金资助项目

周海峰　男, 1976 年出生, 博士研究生, 主要研究兴趣为电子设计自动化.

林争辉　男, 1933 年出生, 教授、博士生导师, 主要研究方向为电路与系统理论和电子设计自动化.

曹　炜　男, 1973 年出生, 博士, 主要研究方向为电子系统设计自动化.

2002-03-07 收到, 2002-06-04 定稿