

A Design Method for Irredundant Cascades

Tsutomu SASAO^{1,2}, Munehiro MATSUURA¹, and Yukihiro IGUCHI³

¹Department of Computer Science and Electronics, Kyushu Institute of Technology

²Center for Microelectronic Systems, Kyushu Institute of Technology

³Department of Computer Science, Meiji University

Abstract

A realization of multiple-output logic functions using an irredundant cascade is presented. First, a multiple-output function is represented by an encoded characteristic function for non-zeros (ECFN). Then, it is represented by a cascade of look-up tables (LUTs). Multiple-output functions for benchmark functions are realized by cascades of LUTs, and the number of LUTs and levels of cascades are shown.

Key words: Reconfigurable hardware, Multiple-output logic function, Functional decomposition, Cascade realization, FPGA synthesis, Time domain multiplexing (TDM)

1. Introduction

Two of the most crucial problems in system LSIs are their long design time and short life cycles [3]. A solution to these problems may be reconfigurable architecture. Reconfigurable LSIs will reduce the hardware development time drastically, since one LSI can be used for various applications.

In this paper, we consider a realization of combinational logic functions by reconfigurable architecture. Various methods exist to realize multiple-output logic functions by reconfigurable architecture. Among them, random access memories (RAMs) and programmable logic arrays (PLAs) directly implement logic functions. However, when the number of input variables n is large, the necessary hardware becomes too large. Thus, field programmable logic arrays (FPGAs) are often used. Unfortunately, FPGAs require layout and routing in addition to logic design. Also, the area for programming and interconnections are much larger than the logic area. Thus, FPGAs require large chip area.

When speed of the operation is not so important, a general-purpose microprocessor can be used to implement logic functions. However, the microprocessor implementation is often 100 to 1000 times slower than the direct circuit realizations. Also, the power dissipation is rather high.

In this paper, we survey our recent work on cascade synthesis [19, 6, 20, 12, 21]

Table 2.1. Decomposition chart.

		$X_1 = (x_1, x_2)$				
		0	0	1	1	
		0	1	0	1	
$X_2 = (x_3, x_4)$	0	0	0	1	1	0
	0	1	1	1	1	1
	1	0	0	1	1	0
	1	1	0	0	0	0

2. Functional Decomposition using BDDs

Definition 2.1 Let $X = (x_1, x_2, \dots, x_n)$ be input variables. A set of variables X is denoted by $\{X\}$. $X = (X_1, X_2)$ is a partition of X if $\{X_1\} \cup \{X_2\} = \{X\}$ and $\{X_1\} \cap \{X_2\} = \phi$. The number of variables in X is denoted by $|X|$.

Definition 2.2 For a logic function $f(X)$, let $X = (X_1, X_2)$ be a partition of X . The **decomposition chart** of f , denoted by $M(f : X_1, X_2)$, is the matrix having 2^{n_1} columns and 2^{n_2} rows. In $M(f : X_1, X_2)$, each row and column has label with binary number, and the corresponding element denotes the truth value of f , where $n_1 = |X_1|$ and $n_2 = |X_2|$. The columns and rows have all possible patterns of n_1 bits and n_2 bits, respectively.

Example 2.1 Let $f(X)$ be a 4-variable function, and $X = (X_1, X_2)$ be a partition of X , where $X_1 = (x_1, x_2)$ and $X_2 = (x_3, x_4)$. Table 2.1 is an example of a decomposition chart. (End of Example)

Definition 2.3 The number of different column patterns in a decomposition chart is the **column multiplicity**, and is denoted by μ .

The column multiplicity of a decomposition chart depends on the partition $X = (X_1, X_2)$ of the input variables.

Lemma 2.1 [2, 5] Let the partition of X be (X_1, X_2) . If the column multiplicity of the decomposition chart $M(f : X_1, X_2)$ for a function f is μ , then f can be represented as $f(X) = g(h_1(X_1), h_2(X_1), \dots, h_u(X_1), X_2)$, and f can be realizable with the network structure shown in Fig. 2.1, where $u = \lceil \log_2 \mu \rceil$. X_1 is the **bound set**.

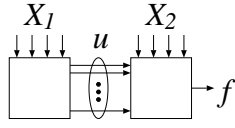


Figure 2.1. Functional Decomposition.

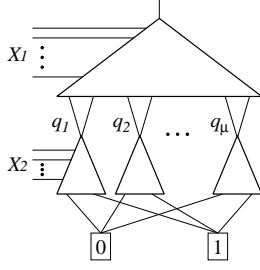


Figure 2.2. Functional decomposition using BDD.

Lemma 2.2 [8, 14] Let (X_1, X_2) be a partition of X , and let the BDD of the function f be partitioned as shown in Fig. 2.2. Suppose that k nodes in the lower block are adjacent to the upper block. Also, let the column multiplicity of the decomposition chart $M(f : X_1, X_2)$ for the function f be μ . Then, $\mu = k$.

Definition 2.4 [11] The width of the BDD at level k is the number of edges crossing the section of the graph between x_k and x_{k+1} , where the edges pointing to the same nodes are counted as one. The width of the BDD is the maximum width of the BDD among the levels.

Theorem 2.1 Consider a BDD for an n -variable logic function f . Let the width of the BDD be μ_{max} . If $u = \lceil \log_2 \mu_{max} \rceil \leq k - 1$, then f can be realized by a cascade of k -LUTs shown in Fig. 2.3. Let s be the numbers of levels of the cascade, and N be the number of LUTs, then we have

$$\left\lceil \frac{n+u-2}{k-1} \right\rceil \leq s \leq 1 + \left\lceil \frac{n-u-1}{k-u} \right\rceil$$

$$\left\lceil \frac{n+u-2}{k-1} \right\rceil + u - 1 \leq N \leq \left\lceil \frac{n-u-1}{k-u} \right\rceil u + 1$$

Theorem 2.2 Let s be the number of levels of the cascade in Fig. 2.3. Then, we have

$$\left\lceil \frac{n-\hat{u}}{k-\hat{u}} \right\rceil \leq s \leq 1 + \left\lceil \frac{n-\hat{u}-1}{k-\hat{u}} \right\rceil, \text{ where } \hat{u} = \frac{1}{s-1} \sum_{i=1}^{s-1} u_i.$$

$u_i = \lceil \log_2 \mu_i \rceil$ and μ_i is the column multiplicity for the decomposition of f , where $X_1 \cup X_2 \cup \dots \cup X_i$ is the bound set.

Theorem 2.2 gives tighter bounds on s than Theorem 2.1. Since \hat{u} is hard to obtain, we approximate it by the average value of the logarithm of the widths of all the levels in the BDD. From these relations, we can easily estimate the number of LUTs and the level of the cascade.

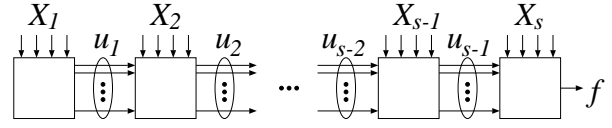


Figure 2.3. The network obtained by applying functional decompositions $s - 1$ times.

3. Representation of Multiple-output Function by ECFN

Although the method described in the previous section is useful for a single-output function, it is hard to apply to multiple-output functions. In the case of an m -output function, the number of terminal nodes of the MTBDD [15] can be as much as 2^m , which may be too large to construct. Also, the representations using characteristic function (CF) of multiple-output function have been developed [1]. However, in many cases, BDDs for CFs are too large to construct. From this, we use the following method to represent a multiple-output function.

Definition 3.1 [18] Let m functions be f_j ($j = 0, 1, \dots, m - 1$). The encoded characteristic function for non-zero outputs (ECFN) is

$$ECFN = \bigvee_{j=0}^{w-1} z_{w-1}^{b_{w-1}} z_{w-2}^{b_{w-2}} \dots z_0^{b_0} f_j,$$

where $\vec{b} = (b_{w-1}, b_{w-2}, \dots, b_0)$ is the binary representation of the integer j , and $w = \lceil \log_2 m \rceil$.

Note that z_0, z_1, \dots, z_{w-1} are auxiliary variables that represent the outputs.

Example 3.1 Consider the case of $m = 8$. Let z_0, z_1 , and z_2 be the auxiliary variables that represent output groups. In this case, the ECFN of the 8-output function (f_0, f_1, \dots, f_7) is represented by

$$ECFN = \bar{z}_2 \bar{z}_1 \bar{z}_0 f_0 \vee \bar{z}_2 \bar{z}_1 z_0 f_1 \vee \bar{z}_2 z_1 \bar{z}_0 f_2 \vee \bar{z}_2 z_1 z_0 f_3 \vee z_2 \bar{z}_1 \bar{z}_0 f_4 \vee z_2 \bar{z}_1 z_0 f_5 \vee z_2 z_1 \bar{z}_0 f_6 \vee z_2 z_1 z_0 f_7$$

(End of Example)

When constructing a BDD for an ECFN, we can reduce the width of the BDD by mixing the auxiliary variables and ordinary input variables. As will be shown in the experimental results, the widths of the BDDs obtained in this way are, in most cases, smaller than those of SBDDs, BDDs for CFs, and MTBDDs for the corresponding functions. When all the auxiliary variables are adjacent to the root node, the BDD is equivalent to an ordinary SBDD. On the other hand, when all the auxiliary variables are adjacent to the terminal nodes, the BDD is equivalent to the MTBDD. We can also reduce the sizes of BDDs by considering the encoding methods [18].

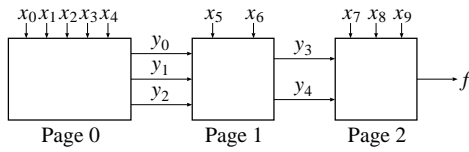


Figure 3.1. Cascade.

4. Experimental Results

4.1. Realization of Cascades

Table 4.1 shows the results for the cascade realization of benchmark functions. This table shows that the size of BDDs for ECFNs are, in most cases, smaller than corresponding MTBDDs and BDDs for CFs. Blank entries show that the BDDs were too large to construct.

We optimized the BDD for ECFN by mixing the input variables and auxiliary variables. We found the ordering of the variables by using a heuristics that reduces the total number of nodes in the QROBDD [15]. Note that this heuristic will reduce \hat{u} in Theorem 2.2. For some functions, the number of nodes in the BDDs increases, in spite of the reduction of \hat{u} . In the table, μ_{max1} denotes the width of the shared BDDs (SBDDs), and μ_{max2} denotes the width of the BDD for the ECFN. In most cases, our heuristics reduced the widths of BDDs: Exceptions are C3540 and rot. s_1 shows the lower and upper bounds on the number of levels obtained from Theorem 2.1. Similarly, s_2 shows the lower and upper bounds on the number of levels obtained from Theorem 2.2. We can see that s_2 is tighter than s_1 . Also, s denotes the number of levels in a cascade, and N denotes the number of LUTs. In this experiment, encodings of outputs [18] are not optimized. Currently, we do not have a good algorithm that works for large BDDs.

Murgai-Hirose-Fujita [13] have developed a logic simulation system which realizes given function by using k -LUT ($k = 15$). In their paper [13], no level of the networks are shown. So, we did similar experiment by using MIS-FPGA, and obtained N , the number of LUTs, and s , the number of levels. In this experiment, we used the following script:

```
> xl_imp -n 2
> xl_partition -n 15
> simplify
> xl_partition -n 15
```

The results are shown in the last two columns of Table 4.1. In most cases, MIS-FPGA produced networks with more LUTs, but fewer levels.

Since the evaluation time of the cascade is proportional to $s \cdot m$, when m is large we have to reduce the number of levels by partitioning the output set. Table 4.2 compares the numbers of LUTs and levels of cascades when the outputs are partitioned into four and eight groups. By partitioning the outputs into four groups, the number of levels can be reduced to half. In this case, the parallel evaluation is more than eight times faster than the original one.

Table 4.2. Results of Output Partition.

Name	Without partition		4 partition		8 partition	
	s	LUTs	s	LUTs	s	LUTs
C2670	28	170	12	178	11	183
C5315	23	142	13	257	10	295
C7552	25	156	17	216	17	203
des	34	235	15	319	9	293
rot	18	125	9	179	7	203

5. Conclusions

In this paper, we have shown a method to represent a multiple-output logic function by a cascade of k -LUTs.

The features of the method include:

1. The system uses a cascade of LUTs: The hardware is simple to implement. The design consists of iterative decompositions of BDDs for ECFNs.
2. The system uses multiple-output LUTs: It is faster than Murgai-Hirose-Fujita's simulator.
3. The system uses BDDs for ECFNs, which are smaller than the corresponding SBDDs: The input variables and the auxiliary variables are mixed to reduce the BDDs.

In this paper, we only considered the case where the values of k are the same for all the stages of a cascade. However, in general, the value of k can be different for different stages. By using this technique, we can implement larger function on a smaller memory.

Acknowledgments

This research is partly supported by the grant in Aid for Scientific Research of the Japan Society for the Promotion of Science (JSPS), and Takeda Foundation. Dr. R. Murgai of Fujitsu Laboratory of America showed us the method to use MIS-FPGA. Prof. Jon T. Butler's comments improved the English presentation. Prof. Qingjian Yu's comments improved Theorems 2.1 and 2.2.

References

- [1] P. Ashar and S. Malik, "Fast functional simulation using branching programs," *ICCAD'95*, pp. 408-412, Oct. 1995.
- [2] R. L. Ashenurst, "The decomposition of switching functions," *In Proceedings of an International Symposium on the Theory of Switching*, pp. 74-116, April 1957.
- [3] R. K. Brayton, "Future of logic synthesis and Verification," in S. Hassoun and T. Sasao (e.d.), *Logic Synthesis and Verification*, Kluwer Publishers, (2001-11).
- [4] R. E. Bryant, "Graph-based algorithms for Boolean function manipulation," *IEEE TC*, Vol. C-35, No. 8, pp. 677-691, Aug. 1986.

Table 4.1. Experimental Results ($k = 15$).

Name	In	Out	MTBDD	BDD for CF	SBDD	BDD for ECFN	μ_{max1}	μ_{max2}	Lower and Upper bounds				This method		MIS-FPGA		
									s_1		s_2		s	N	s	N	
C432	36	7	1198	885	1075	859	101	83	4	5	4	5	4	20	9	22	
C499	41	32			27876	24944	2176	2048	4	10	6	7	7	7	60	4	104
C880	60	26			4166	4567	466	464	6	11	8	9	8	5	51	9	66
C1908	33	25	9564117		7456	7548	620	620	4	7	5	6	5	35	7	104	
C2670	233	140			2847	3688	411	284	18	40	28	29	28	170	5	186	
C3540	50	22			34710	39320	5420	5482	5	22	11	11	11	107	13	154	
C5315	178	123	537	1786	2564	3256	258	238	14	27	22	23	23	142	7	188	
C7552	207	108			2945	3648	193	160	16	31	25	26	25	156	7	254	
apex3	54	50			986	1207	204	165	5	9	6	7	6	28	4	86	
apex7	49	37	32720	1582	300	437	55	49	5	7	5	6	5	21	2	38	
b9	41	21			177	219	42	40	4	6	4	5	4	14	2	22	
dalu	75	16			522749	9042	1178	1218	244	149	7	11	8	9	8	40	12
des	256	245	638	755	3975	3740	608	285	20	44	34	35	34	235	2	309	
duke2	22	29			366	452	78	48	3	4	3	4	3	10	3	38	
e64	65	65			131	2277	194	675	66	36	6	9	7	8	25	5	69
ex4*	128	28	4722244	2860	540	602	83	38	7	11	8	9	8	32	2	33	
k2	45	45			913	1321	1640	251	245	5	7	5	6	6	28	6	161
rot	135	107			8501	9658	1196	1204	11	34	18	19	18	125	7	141	
spla	16	46	11100	2121	628	626	101	69	2	3	2	3	2	8	6	144	

μ_{max1} : The width of SBDD.

μ_{max2} : The width of the BDD for the ECFN.

s_1 : Lower and upper bounds on the number of levels obtained by Theorem 2.1.

s_2 : Lower and upper bounds on the number of levels obtained by Theorem 2.2.

s : Number of levels.

N : Number of LUTs.

*: Contains redundant variables. We used the number of dependent variables to obtain the bounds.

- [5] H. A. Curtis, *A New Approach to The Design of Switching Circuits*, D. Van Nostrand Co., Princeton, NJ, 1962.
- [6] Y. Iguchi, T. Sasao, and M. Matsuura, "Realization of multiple-output functions by reconfigurable cascades," *International Conference on Computer Design: VLSI in Computers & Processors (ICCD-2001)*, Austin, TX, Sept. 23-26, 2001, pp. 388-393.
- [7] J.-H. R. Jian, J.-Y. Jou, and J.-D. Huang, "Compatible class encoding in hyper-function decomposition for FPGA synthesis," *Design Automation Conference*, pp. 712-717, June 1998.
- [8] Y.-T. Lai, M. Pedram, and S. B. K. Vrudhula, "EBDD-based algorithm for integer linear programming, spectral transformation, and functional decomposition," *IEEE Trans. CAD*, Vol. 13, No. 8, pp. 959-975, Aug. 1994.
- [9] C. Lee, "Representation of switching circuits by binary-decision programs," *Bell System Technical Journal*, Vol. 19, pp. 985-999, July 1959.
- [10] P. C. McGeer, K. L. McMillan, A. Saldanha, A. L. Sangiovanni-Vincentelli, and P. Scaglia, "Fast discrete function evaluation using decision diagrams," *ICCAD'95*, pp. 402-407, Nov. 1995.
- [11] S. Minato, "Minimum-width method of variable ordering for binary decision diagrams," *IEICE Trans. Fundamentals*, Vol. E75-A, No. 3, pp. 392-399, March 1992.
- [12] A. Mishchenko and T. Sasao, "Encoding of Boolean functions and its application to LUT cascade synthesis," *International Workshop on Logic and Synthesis (IWLS2002)*, New Orleans, Louisiana, June 4-7, 2002, pp. 115-120.
- [13] R. Murgai, F. Hirose, and M. Fujita, "Logic synthesis for a single large look-up table," *Proc. International Conference on Computer Design*, pp. 415-424, Oct. 1995.
- [14] T. Sasao, "FPGA design by generalized functional decomposition," (Sasao ed.) *Logic Synthesis and Optimization*, Kluwer Academic Publishers, 1993.
- [15] T. Sasao and M. Fujita (ed.), *Representations of Discrete Functions*, Kluwer Academic Publishers 1996.
- [16] T. Sasao and J. T. Butler, "A method to represent multiple-output switching functions by using multi-valued decision diagrams," *IEEE International Symposium on Multiple-Valued Logic*, pp. 248-254, Santiago de Compostela, Spain, May 29-31, 1996.
- [17] T. Sasao, *Switching Theory for Logic Synthesis*, Kluwer Academic Publishers, 1999.
- [18] T. Sasao, "Compact SOP representations for multiple-output functions: An encoding method using multiple-valued logic," *International Symposium on Multiple-Valued Logic*, Warsaw, Poland, 2001, pp. 207-212.
- [19] T. Sasao, M. Matsuura, and Y. Iguchi, "A cascade realization of multiple-output function for reconfigurable hardware," *International Workshop on Logic and Synthesis (IWLS01)*, Lake Tahoe, CA, June 12-15, 2001, pp. 225-230.
- [20] T. Sasao, M. Matsuura, Y. Iguchi, and S. Nagayama, "Compact BDD representations for multiple-output functions and their applications to embedded system," *IFIP VLSI-SOC'01*, Montpellier, France, December 3-5, 2001, pp. 406-411.
- [21] T. Sasao, "Design methods for multi-rail cascades," (invited paper), *International Workshop on Boolean Problems (IWBP2002)*, Freiberg, Germany, Sept. 19-20, 2002, pp. 123-132.