10

Balanced multilevel decomposition and its applications in FPGA-based synthesis

T. Łuba, H. Selvaraj, M. Nowicka and A. Kraśniewski
Warsaw University of Technology, Institute of Telecommunications
Nowowiejska 15/19, 00-665 Warsaw, Poland

Abstract

A general decomposition concept is presented in this paper. The main strategy behind the presented Multilevel Decomposition Method is striking a balance between serial decomposition and parallel decomposition. The method is applicable to a variety of Field Programmable Gate Arrays and allows trading-off area and delay of final implementation. The results prove that the method is efficient and does not suffer from its generality.

Keywords

Decomposition, FPGA, technology mapping, delay minimization

1 INTRODUCTION

Existing FPGA-oriented technology mapping algorithms can be roughly divided into two categories according to their optimization objectives. The algorithms in the first category emphasize minimizing the number of LUTs in the solution. These algorithms include Chortle-crf (Francis et al., 1991), MIS-pga (Murgai et al., 1991), Xmap (Karplus, 1991), Trade (Wan and Perkowski, 1992) and TOS (Schlichtman, 1993). The algorithms in the second category emphasize minimizing the delay in the solution. These are DAG-Map (Chen et al., 1992) and Flow-map (Cong and Ding, 1994). Several algorithms have been implemented in both the area- and delay-minimization versions, for example MIS-pga, Chortle, and ASYL (Abouzeid et al., 1993).

Although many of the above mentioned algorithms produce encouraging results, they have a common limitation that, for a given design, a simple solution optimized with respect to a certain criterion is produced, while other solutions, preferable under different optimization objectives are ignored. Usually the logic depth (length of critical path) and the number of LUTs for these solutions vary significantly. For example, the mapping results on the Alu2 benchmark circuit, obtained by area- and delay-minimization versions of the Flow-Map and Chortle are 125 cells/12 levels or 149 cells/8 levels and 128 cells/13 levels or 227 cells/9 levels, respectively (Cong and Ding, 1994).

G. Saucier et al. (eds.), *Logic and Architecture Synthesis*

In general, the area-minimized designs have much larger depth, while delay-minimized designs use much more LUTs.

It is very likely that in practice the best design does not come from either of these two extreme solutions. It is important to let the designer have the flexibility to choose from a set of mapping solutions with smooth trade-off between area and depth.

In this paper we propose a balanced multilevel decomposition to solve the problem of trading-off area and depth in LUT-based FPGA technology mapping. Specifically, we are interested in obtaining a set of mapping solutions for a given design, which can meet various area and depth requirements.

Unlike the traditional approach of first performing logic minimization and then mapping the design using decomposition and other methods, the proposed method starts the mapping process straight away using different decomposition strategies described in our previous papers (Selvaraj, 1993, Luba, 1994). An important feature of multilevel decomposition is an attempt to strike a balance between the wider known serial decomposition (Luba et al. 1992, Wan and Perkowski, 1992) and the parallel decomposition (Jozwiak and Volf, 1992, Luba, 1994). Such a strategy, on one hand, is capable of detecting and utilizing the fact that a group of outputs depend on the same set of input variables and, on the other hand, eliminates redundant variables.

2 BASIC CONCEPTS

Taking into consideration functional dependencies (Luba, 1994), two distinct decomposition strategies can be identified for decomposing Boolean functions (Fig. 1).

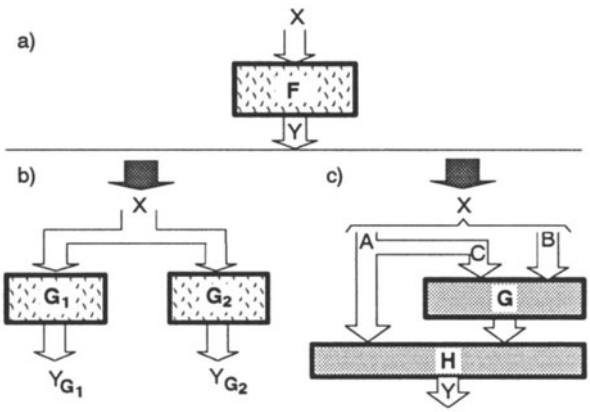

Figure 1 Parallel and serial decomposition of function *F*

The first one, called parallel decomposition (Fig 1b), relies on partitioning the outputs into two separate groups, so that each group of outputs depend on fewer or the same number of

variables as a given function F i.e. $Y = Y_{G1} \cup Y_{G2}$, where for X_{G1} and X_{G2}, the respective support sets of Y_{G1} and Y_{G2}, we have $X_{G1} \subseteq X$ and $X_{G2} \subseteq X$.

The second decomposition strategy, called serial decomposition (Fig. 1c), relies on partitioning the input variables so that to obtain a two-level functional dependency $F = H(A,G(B \cup C))$, where $A \cup B = X$ and $C \subset A$.

The essence of the serial decomposition is to find functions G and H, such that G depends on variables in $B \cup C$, whereas H depends on variables in A and variables in Z, where Z is the set of output variables of G.

These two strategies of decomposition play a crucial role in developing a general multilevel decomposition model.

The method is based on balancing serial and parallel decomposition, which in turn relies on certain features of minimal disjoint serial decomposition (Selvaraj, 1994).

Let $F(X)$ denote a Boolean function F with a set of input variables X. Let $X = A \cup B$ and $A \cap B = \emptyset$, where B is the set of bound variables (block G inputs) and A is the set of free variables (direct inputs to block H).

Theorem 1.

For every function F with $|X|$ inputs, $|Y|$ outputs, $|A|$ free variables and $|B|$ bound variables, if $|X| > 2|Y|$ and $|B| > |Y|2^{|A|}$, then there exists a minimal disjoint serial decomposition such that $F = H(G(B),A)$, where the number of block G outputs is $|Y|2^{|A|}$.

Theorem 1 leads to an observation that the existence of a disjoint serial decomposition $F = H(G(B),A)$, for a given number of inputs n and outputs m, is conditioned by the number of free variables (direct inputs to block H) and the properties of function F.

Using the properties of the theorem on r-admissibility (Luba, 1994) it is possible to calculate the set of free variables A, which determines the minimum number of block G outputs (and thereby the minimum number of inputs to block H or, in other words, a maximum decomposition). On the other hand, theorem 1 can be used to find whether or not a minimum decomposition exists. With the knowledge of the two extreme possibilities, it is relatively easy to find the actual maximum decomposition.

An intriguing situation arises when the process of iterative decomposition leads to a unitary function. This situation can be considered as a special case of theorem 1. By definition, a unitary function has one output and hence $|Y| = 1$. Therefore, the maximum number of block G outputs is $2^{|A|}$. If the number of inputs to block G, $|B| > 2^{|A|}$, a minimal disjoint serial decomposition exists.

The practical importance of the above conclusion is significant as it states that a disjoint serial decomposition always exists for a unitary function if $|B| > 2^{|A|}$. Now, if minimum value is assumed for the number of free variables ($|A| = 1$), the number of bound variables $|B|$ is at least 3. This means that any unitary function with a minimum of four input variables has a disjoint serial decomposition.

When a need arises to break a function F into subfunctions with two inputs, the iteration discussed so far may not lead to a complete solution. So, when a disjoint serial decomposition does not exist for a three-input unitary function, it is proposed to try non-disjoint serial (or conjunct) decomposition, $F = H(G(B),A)$, where $A \cup B = X$, $A \cap B \neq \emptyset$.

3 GENERAL AND COMPLETE DECOMPOSITION

On the basis of the presented concepts, a Multilevel Decomposition Method (MDM) has been developed. The proposed method is complete and general and can be applied to any type of FPGA Logic Block (LB) with an arbitrary number of inputs and an arbitrary number of outputs. The method looks at the Logic Block as a device capable of implementing any function with a fixed number of inputs and outputs. Unlike the traditional approach of first performing logic minimization and then mapping the design using decomposition and other methods, the proposed method starts the mapping process straight away using decomposition strategies discussed above.

Consider a function F: $\{0,1,-\}^n \rightarrow \{0,1,-\}^m$. Let us try to implement the function using, as a basic component, a hypothetical logic block LB with C_{in} inputs and C_{out} outputs, where $C_{in} > C_{out}$.

The method first tests "the sensibility" of performing disjoint serial decomposition. As LB has C_{in} inputs and C_{out} outputs, an ideal serial decomposition is the one where the number of block G inputs, G_{in}, is equal to the number of LB inputs, C_{in}, and the number of block G outputs, G_{out}, is equal to the number of LB outputs, C_{out}. If a serial decomposition is attempted with G_{in} and G_{out} as the number of block G inputs and outputs respectively, the resulting block H will have $n - (G_{in}-G_{out})$ inputs. If this number is not greater than m, then it is proposed to use parallel decomposition. In other words, if the number of inputs to the resulting block H is not expected to be greater than the number of its outputs, the truth table is split into two parts using parallel decomposition. Each part is then decomposed separately using the same method.

An interesting situation arises when the number of inputs to the truth table under consideration is greater than C_{in}, the "sensibility" test described earlier does not recommend parallel decomposition, and the iteration process is not able to produce any result.

In such a situation, the parameters G_{in} and G_{out} are adjusted to G'_{in} and G'_{out}, respectively, and serial decomposition is attempted to look for the existence of any possible function G', such that: $F' = H(A, G'(x_{i1},..x_{iG'in}))$, where F' is the function under consideration and $x_{i1},...,x_{iG'in} \in B$.

If a successful decomposition is found and the number of inputs to the truth table H continues to be larger than C_{in} (number of inputs to the given logic block), the iteration process is continued with the truth table H as the new starting point after resetting the number of G block inputs to C_{in} and the number of G block outputs to C_{out}.

The G' function is decomposed separately, using the same iterative method, so that

$$G' = H(A, G(x_{i1},...,x_{iGin})).$$

One important aspect of the method not discussed so far is the procedure for controlling the block G input/output parameters G_{in} and G_{out}. If C_{in} and C_{out} are the number of inputs and outputs of the Logic Block for which the synthesis is done, initially G_{in} and G_{out} assume the values C_{in} and C_{out}, respectively. Now, let us consider a situation when there is no further serial decomposition and parallel decomposition is not recommended. Under such conditions, it is proposed to increment the value of G_{in} by one, i.e., $G'_{in} = G_{in} + 1$. However, before increasing the number of block G inputs, it is made sure that for a given number of inputs G_{in}, there is no serial decomposition for $G_{out} = C_{out}, C_{out}+1,...,G_{in}-1$. For an LB with

$C_{in}-C_{out} > 1$, the existence of decomposition is verified for $G_{in} = C_{in}, C_{in}-1,...,G_{out}+1$.

For example, if the selected FPGA LB has two inputs and one output, then the consecutive block G parameters, G_{in} and G_{out}, will be 2,1; 3,1; 3,2; 4,1; 4,2; 4,3; 5,1 etc. On the other hand, if the selected FPGA LB has four inputs and one output, then the consecutive block G parameters G_{in} and G_{out} will be 4,1; 3,1; 2,1; 4,2; 3,2; 4,3; 5,1; etc.

By relating the disjoint serial decomposition to the number of outputs, theorem 1 in practice shows the need to balance between parallel and disjoint serial decomposition strategies. When there is no possibility of finding a disjoint serial decomposition, parallel decomposition can be performed to split the truth table into two separate parts, so that the search for disjoint serial decomposition can be continued. In the view of theorem 1, parallel decomposition is a tool with which it is possible to force the condition $n > (m + 1)$ for function F. In the exceptional case when the process has lead to a three-variable unitary function and no disjoint serial decomposition is found for that function, conjunct decomposition is performed. If even this fails, complex decomposition is applied. Such a situation may, however, arise only if there is a need to decompose and implement a function using FPGA cells having two inputs and one output.

4 EXPERIMENTAL RESULTS AND CONCLUSIONS

Based on the presented general decomposition algorithm, a logic synthesis system called DEMAIN has been developed for both HP-UX and MS-DOS operating systems.

An input to DEMAIN is a truth table in Berkeley's cube-based format and the output is a network of k-input l-output cells, each realizing an n-variable function with m outputs. The numbers k, l can be set arbitrarily.

DEMAIN can run in either interactive or automatic mode. The automatic mode minimizes the number of Logic Blocks. In the interactive mode, the user can steer the synthesis process, according to the requirements of the chosen type of FPGA. For example, this option can be used effectively for FPGAs with Logic Blocks capable of implementing functions with different number of variables and to optimize the number of Logic Blocks or to optimize the number of levels for FPGAs with restricted interconnects. It is also possible to balance between delay and area optimization as the situation may demand.

Table 1 shows the results of decomposition of a few benchmark circuits into five-input two-output cells, which is in fact the decomposition aimed at the Xilinx Logic Blocks. The comparison of results produced by DEMAIN with the other published results shows that the proposed method does not suffer because of its universality; in fact, it provides better solutions in many cases.

Table 2 compares the minimum logic depth solutions generated by DEMAIN with those generated by other mapping algorithms designed for delay minimization.

It can be seen that, in spite of the universal character of DEMAIN, the results obtained are not worse than the results of the other programs aimed exclusively at delay optimization.

Because of arbitrary number of cell inputs accepted by DEMAIN, its applications are wider than other FPGA-oriented tools. This is because the logic cell is treated as a universal cell capable of implementing any k-input l-output Boolean function. For example, in the XILINX 3000 family, each CLB can be programmed to implement any single-output function of up to 5 input variables or any two-output function of up to 5 variables, with each output depending

Table 1 Results on number of Xilinx LBs

Name	*DEMAIN*	*TRADE*	*TOS*	*MIS-PGA*	*Chortle*
rd84	8	8	10	9	28
rd73	5	5	7	5	15
z4	4	4	4	4	4
Sao2	23	27	23	28	26
9sym	5	6	7	7	51
root	23	21	–	–	–
b9	32	29	–	32	30
alu2	17	22	47	96	94
Clip	18	29	22	23	25

Table 2 Results for number of LBs/number of levels

Name	*DEMAIN*	*Chortle-d*	*DAG-Map*	*Flow-Map*	*MIS-PGA*	*ASYL*
rd84	8/3	61/4	43/4	43/4	13/3	14/3
rd73	5/4	52/4	–	–	8/2	–
Sao2	23/6	58/4	–	–	45/5	30/8
9sym	5/4	63/5	61/5	61/5	7/3	8/3
alu2	17/3	227/9	169/8	162/8	122/6	60/6
Clip	18/5	83/4	–	–	54/4	33/6

on at most 4 input variables. Similarly, the ALTERA FLEX cell can implement any function of up to 4 input variables. Even the ACTEL cell which consists of three interconnected 2-to-1 multiplexers can be treated as a cell capable of implementing any Boolean function of two variables. An internal structure of the cell is irrelevant for DEMAIN. This leads us to conclude that our system is also suitable for other types of FPGAs, like those of Algotronix and Quick Logic (Selvaraj et al. 1993).

5 REFERENCES

Abouzeid P., Babba B., Crastes de Paulet M., Saucier G. (1993) Input-Driven Partitioning Methods and Application to Synthesis on Table-Lookup-Based FPGAs. *IEEE Trans on CAD*, **12**, No. 7, 913-925.

Chen K.C., Cong J., Ding Y., Kahng A.B., and Trajmar P. (1992) DAG-Map: Graph-based FPGA technology mapping for delay optimization. *IEEE Design and Test of Computers*, Sept. 7-20.

Cong J, and Ding Y. (1994) FlowMap: An Optimal Technology Mapping Algorithm for Delay Optimization in Lookup-Table Based FPGA Designs. *IEEE Trans. on CAD*, **13**, No. 1, 1-12.

Francis R., Rose J, Vranesic Z. (1991) Chortle-crf: Fast Technology Mapping Look-up Table-Based FPGAs, *Proc. 28-th ACM/IEEE Design Automation Conf.*, 227-233.

Jozwiak L., Volf F. (1992) An Efficient Method for Decomposition of Multiple-Output Boolean Functions and Assigned Sequential Machines, *Proc. European Conference on Design Automation*, 114-122.

Karplus K. (1991) Xmap: A Technology Mapper for Table-Lookup Field Programmable Gate Arrays, *Proc. 28th ACM/IEEE Design Automation Conf.*, 240-243.

Luba T., Markowski M., Zbierzchowski B. (1992) Logic Decomposition for Programmable Gate Arrays, *Proc. Euro-ASIC'92*, 19-24.

Luba T. (1994) Multi-level logic synthesis based on decomposition. *Microprocessors and Microsystems*, **18**, No. 8, 429-437.

Murgai R., Shenoy N., Brayton R.K., Sangiovanni-Vincentelli A. (1991) Improved logic Synthesis Algorithm for Table Look Up Architectures, *Proc. IEEE International Conf. on Computer Aided Design*, 564-567.

Schlichtmann U. (1993) Boolean Matching and Disjoint Decomposition for FPGA Technology Mapping. *IFIP Workshop on Logic and Architecture Synthesis*, 83-102.

Selvaraj H., Czerczak A., Kraśniewski A., Łuba T. (1993) A Generalized Decomposition of Boolean Function and its in FPGA-Based Synthesis. *IFIP Workshop on Logic and Architecture Synthesis*, 147-166.

Selvaraj H. (1994) FPGA-based Logic Synthesis. *Ph.D. Dissertation.* Warsaw University of Technology.

Wan W., Perkowski M.A. (1992) A New Approach to the Decomposition of Incompletely Specified Multi-Output Function Based on Graph Coloring and Local Transformations and Its Application to FPGA Mapping, *Proc. European Design Automation Conf.*, 230-235.

6 BIOGRAPHIES

All the authors are from the Warsaw University of Technology, Institute of Telecommunications. Currently Tadeusz Łuba and Andrzej Kraśniewski are Professors. Henry Selvaraj has just received his Ph.D. degree from this University and Mirosława Nowicka is a member of research staff in the Institute.