Stanisław DENIZIAK, **Mariusz WIŚNIEWSKI**, Karol WIECZOREK
KIELCE UNIVERSITY OF TECHNOLOGY, DEPARTMENT OF COMPUTER SCIENCE,
7 Tysiaclecia Panstwa Polskiego Ave., 25-314 Kielce, Poland

Synthesis of multiple-valued logic networks for FPGA implementation using developmental genetic programming

Abstract

This paper presents a method of FPGA-oriented synthesis of multiple-valued logical networks. A multiple-valued network consists of modules connected by multivalued signals. During synthesis the modules are decomposed into smaller ones. For this purpose the symbolic decomposition is applied. Since the decomposition of modules strongly depends on the encoding of multiple-valued inputs and outputs, the result of synthesis depends on the order, in which the consecutive modules are implemented. Experimental results showed that our approach significantly reduces the cost of implementation.

Keywords: multiple-valued functions; FPGA; symbolic synthesis.

1. Introduction

Complex logical functions of digital systems usually are specified as a network of simpler functions. High-level specification of a digital system may contain variables represented by enumeration types. Moreover, the high-level synthesis process creates new variables, representing state variables, conditions and control signals, which may also be multivalued (MV). During synthesis, all symbolic values should be encoded using binary patterns and all logic functions should be minimized and decomposed, to find the best implementation of the system [1].

In [2], we proposed the symbolic functional decomposition of multi-valued logic (MVL) functions, targeted at FPGA implementations. We showed that integrated encoding and functional decomposition gives significant improvement in FPGA implementation of the function, as far as the cost of implementation is concerned. We proposed efficient methods for functions with MV inputs, outputs and finite state machines. In the case of MV networks, modules share MV signals and it is not possible to find optimal encoding for all functions sharing the same MV variable. Thus, it is necessary to decide for which function encoding should be optimized. It is determined by the proper ordering of modules subjected to decomposition.

The implementation of logic functions in LUT-based FPGAs consists of logic synthesis and technology mapping. The most efficient synthesis method for LUT-based FPGAs is the functional decomposition based on the blanket algebra [3] or on the theory of information relationship measures [4]. The blanket algebra was also generalized to MVL functions [5] and the symbolic decomposition [2] for MVL functions was proposed. The method was successfully applied to logical functions with MV inputs/outputs and FSMs[6] giving significant improvements in the cost of FPGA implementations. It was also observed that the symbolic decomposition minimizes the number of logic levels [2].

In this paper, we propose the method for determining the optimal or suboptimal order of modules that are decomposed during FPGA-oriented synthesis of MV-networks. The order is optimized using developmental genetic programming (DGP) [8]. During optimization the cost estimation function is used to evaluate the quality of each solution. Finally, the MV-network is decomposed according to the order determined by the best solution.

The paper is organized as follows. The next section presents the problem of FPGA-oriented synthesis of MV networks. Our approach is described in Section 3. Section 4 presents experimental results. Conclusions are presented in Section 5.

2. Symbolic synthesis and MVL networks

Let F be a MVL function, such that Y=F(X), where X is a set of binary/MV inputs and Y is a set of binary/MV outputs. The function may be decomposed in a parallel or in a serial fashion. The parallel decomposition expresses function F through functions G and H with disjoint sets of output variables. The serial decomposition expresses F(X) through functions G and H, such that H(U, G(V)), where U∪V=X.

Assume that X_i/Y_i is an MV input/output. During the decomposition the variable may be encoded with variables X_i' and X_i''. Both variables may be separated during the decomposition. It was observed [2] that the proper encoding of MV variables leads to significant improvement of decomposition efficiency, i.e. result functions are more simplified.

The efficiency of decomposition has the most influence on the result of synthesis. Fig. 1 presents the sample MVL network, where *A*, *B*, *C*, *D*, *E*, *F* and *G* are MV signals, *FS4*, *FS7* and *FS8* are state variables, x_i and y_i are binary signals. The synthesis may be performed by decomposing the consecutive modules until all modules can be implemented in one LUT cell. But since modules share variables, the encoding optimized for one module may not be optimal for others. For example in Fig. 1, the encoding of variable *A* may be performed during the decomposition of *F1* or *F4* (but not both). Thus, the order of synthesis of modules has a great influence on the final result.

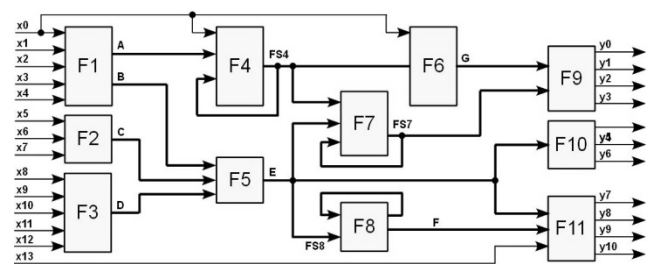

Fig. 1. Sample MVL network

To find the best FPGA-based implementation of MVL network, we should find the optimal order in which all modules will be decomposed. Implementation of one module may reduce the possibilities of optimization for other modules. It should be noted that it is a typical optimization problem. Since for *n* modules we have *n!* possible orders, thus for a large network only heuristic optimizations may be performed. One approach to this problem was presented in [7], but to obtain good results more sophisticated heuristics should be applied.

The synthesis of FSM modules is the most complex one. Moreover, the result of optimization also strongly depends on what MV variables are shared with other modules. The most interesting is a case when two or more FSMs are connected to each other – it happens in the circuit (Fig. 1) between modules *F4*, *F7* and *F8*. First, assume that the synthesis of *F8* will be performed before optimization of modules *F11* and *F7* (variables *E* and *FS8* were optimized to minimize the cost of *F8* – encoding of these variables was obtained using symbolic FSM synthesis [2]). In this case, the cost of implementation is 16 LUT cells.

Second, assume that the synthesis of module *F8* will be performed after optimization of modules *F7* and *F11*. In this case, the input *E* and output *F* will be encoded binary before optimization of *F8*. In this case, the total cost of implementation for such encoding equals to 21 LUTs. Finally, consider the case. where module *F8* is optimized before module *F11* and after *F7*. In this case, input *E* is encoded during symbolic decomposition of module *F7*. In this case, the total cost of synthesis of F8 was equal to 17 LUT cells, therefore it is very close to minimal.

The above discussion showed that the results of synthesis of MVL network strongly depend on the order of modules subjected to decomposition. Since MV variables are shared by modules, we should decide which module should be optimized by the encoding of shared variables.

3. Scheduling of modules for optimal decomposition

Genetic programming (GP) creates computer programs using a genetic algorithm (GA). In the developmental genetic programming (DGP) [8], methods creating solutions evolve instead of computer programs. In this method, the genotype and the phenotype are distinguished. The genotype is a procedure that constructs the solution of a problem. The phenotype represents the target solution.

In our method, genotypes are represented by binary trees specifying the decomposition strategy. Each node (gene) specifies the symbolic decomposition of one module, selected by the strategy corresponding to a type of the gene. Then, the rest of the network is divided into 2 subnetworks passed to offspring nodes for further symbolic decomposition.

Tab. 1 presents the description of all genes, where: S_I/B_I is the number of MV/binary inputs, S_O/B_O is the number of MV/binary outputs, S_S is the number of states, *O(X)*, *O(Y)*, *O(S)* are sums of symbolic values for all MV inputs/outputs and state variables.

Tab. 1. Criteria for selection of the module

Type	Rules I	Rules II	Rules III
C1	Maximal S_I+S_O	Maximal *O(X)+O(Y)*	Maximal B_I+B_O
C2	Minimal S_I+S_O	Minimal *O(X)+O(Y)*	Minimal B_I+B_O
C3	Maximal S_I	Maximal *O(X)*	Maximal B_I
C4	Minimal S_I	Minimal *O(X)*	Minimal B_I
C5	Maximal S_S	Maximal *O(S)*	Maximal S_I+S_O
C6	Minimal S_S	Minimal *O(S)*	Minimal S_I+S_O
C7	FSM with minimal S_S	Minimal *O(S)*	Minimal S_I+S_O
C8	FSM with maximal S_S	Maximal *O(S)*	Maximal S_I+S_O
C9	Maximal S_O	Maximal *O(Y)*	Maximal B_O
C10	Minimal S_O	Minimal *O(Y)*	Minimal B_O

First, the set Rules I is applied. When more than one module meets the criterion, then the set Rules II is concerned for these modules. If still several modules meet the criterion, the criterion from set Rules III is applied. Finally, any module satisfying all three criteria is chosen.

We assume that the module chosen according to the specified criterion will be decomposed, therefore all of its inputs/outputs will be encoded. Next, the module is removed and the rest of the network is cut into 2 subnetworks using the min-cut algorithm.

The initial generation consists of individuals generated randomly. The "embrion" (root of the genotype tree) corresponds to the randomly selected module. In this way, the module which does not meet any of the criteria from Tab. 1, may also be decomposed first. In the most DGP approaches, the root node has a special meaning i.e. it builds the embrion. It cannot narrow the search space. In our approach, the decomposition may start from any module. The evolution will eliminate "wrong" embrions.

The size of the genotype should be attuned to the complexity of the network. Therefore the number of levels *L* in each genotype is calculated according to the formula $L = \lceil \log_2(N) \rceil + 1$, where *N* is the number of modules in the network.

We assume that the synthesis of modules is performed by traversing the genotype tree in the depth-first order and choosing the modules according to the corresponding criteria. The phenotype is a final order of modules.

Two exceptions are possible: first, the node has successors but further ordering is not necessary, second, the node is a leaf but the MVL network requires further ordering. In the first case, the processing of a given subnetwork is not continued and useless nodes are removed from the genotype tree. In the second case, a leaf corresponds to an MV network, modules are synthesized starting from inputs towards outputs.

A sample genotype, describing the synthesis of the network from Fig.1, is shown in Fig.2. The embrion *g0* corresponds to decomposition of module *F7*. Next, the network is divided into subnetworks passed to offspring nodes. Since *F7* will be synthesized first, signals *FS4*, *FS7* and *E* will be binary in the subnetworks. Gene *g1* selects module *F1* (according to *C5*) and divides the network into subnetworks {*F2,F3*} and {*F4,F5*}. Gene *g2* chose module *F8* (*C9*), and so on. Finally, this genotype constructs the following phenotype [*F7,F1,F8,F3,F5,F9,F10, F11, F2, F4,F6*].

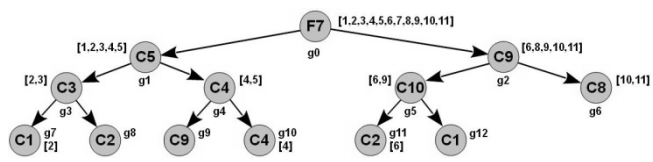

Fig. 2. Sample genotype

Since evolutionary optimization requires numerous individuals to evolve, it would not be efficient to perform full decomposition for each solution. Thus, we decided to apply a simplified and fast method for evaluation of the quality of each solution. This method is based on the probability of optimization. According to our previous research [2] we observed that the optimization possibilities are higher for functions with a greater number of MV variables and for variables where the set of symbolic value is larger. First, for each module *Fi* the best implementation I^i_{opt} is found. Next, for each MV variable v_i a minimal encoding length $L_{min}(v_i)$ is computed. Finally, the probability of optimization for module F_i with MV inputs $i_0,\ldots,i_{k-1}$, state variable s_k and outputs $o_{k+1},\ldots,o_m$ is computed as:

$$P^i_{opt} = 1 - 0.5 * \left(\sum_{j=1}^{m} L_{\min}(v_j) * E_j \right) / \left(\sum_{j=1}^{m} L_{\min}(v_j) \right) \qquad (1)$$

where E_i=1 if variable v_i is binary encoded, otherwise E_i=0.

The expected implementation cost of the module is estimated as $EIC_i = I^i_{opt} / P^i_{opt}$. We assume that without optimized encoding the cost of the implementation will be 2 times higher, on average. The goal of optimization is to find such an order of synthesis that the total expected implementation cost will be as small as possible.

During the decomposition of adjacent modules of F_i, shared variables will be binary encoded. Hence, P^i_{opt} may decrease. Thus, it is very difficult to find the optimal order of modules, because optimization of one module decreases the probability of optimization of other modules. This is a classical optimization problem, where greedy approaches may not be efficient.

Each generation contains the same number of individuals. The size of the generation depends on the number of modules (N) of the MVL network. Thus the number of individuals is calculated using the formula:

$$G = N * \Omega \tag{2}$$

where Ω is a genetic parameter defined by the user.

Genotypes are ranked according to the expected implementation cost of the corresponding phenotypes. Solutions that require less LUT cells for implementation (lower EIC) have a higher position in the ranking. All genotypes are evolved randomly with the P_i that depends on the quality of the solution given as formula: $P_i = (2 * (G - R_i))/(G * (G-1))$, where R_i is the position in the ranking. It is similar to the roulette selection but instead of the fitness the rank position is taken for computing the probability. Reproduction and mutation operators are implemented in similar ways as in other DGP approaches [9].

A new generation is created using the above operations in the following way: $r=\alpha*G$ individuals are reproduced, $c=\beta*G$ individuals are created using cross-over, and $m=\gamma*G$ individuals are created using mutation. G (using formula (2)), and α, β and γ are the DGP parameters, that are tuned experimentally. The following requirement must be fulfilled $\alpha + \beta + \gamma = 1$. If the algorithm does not find any better solution in λ succeeding generations, the evolution stops. Parameters Ω and λ determine the time of the evolution process. They are determined by the user.

4. Experimental results

As an example, the synthesis of a sample network (Fig. 1) will be presented. Since there are no standard benchmark sets for MVL networks it is difficult to compare our approach with other methods. But the complexity of the example given in Fig. 1 is enough to show the advantages of our methodology.

Tab. 2 presents the results of separate synthesis of each module using symbolic synthesis (assumed as I_{opt} during DGP optimization) and Quartus II. The columns contain the number of 4/1 LUTs required for implementation.

Tab. 2. Cost of implementation of modules

Tool	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11
Symb Dec	6	3	6	7	4	9	4	16	4	3	7
Quartus II	10	3	12	31	11	7	26	36	18	3	17

Tab. 3. Cost of implementation of the MV network

Tool		Order	Result
DGP	B1	F6,F3,F9,F4,F8,F10,F11,F2,F7,F1,F5	47
	B2	F3,F6,F9,F2,F8,F10,F11,F4,F7,F1,F5	48
	B3	F2,F8,F11,F6,F10,F4,F9,F3,F7,F1,F5	48
	B4	F2,F6,F9,F4,F8,F10,F11,F1,F3,F5,F7	46
	B5	F8,F10,F11,F6,F9,F3,F4,F1,F2,F5,F7	47
	M1	F1,F6,F10,F2,F8,F9,F11,F3,F4,F5,F7	51
	M2	F10,F6,F9,F2,F7,F8,F11,F1,F3,F4,F5	52
	M3	F4,F8,F9,F3,F6,F10,F11,F1,F2,F5,F7	52
	M4	F9,F3,F8,F1,F2,F5,F10,F4,F6,F7,F11	52
	M5	F6,F1,F10,F4,F8,F9,F11,F2,F3,F5,F7	51
	W1	F5,F3,F11,F1,F2,F7,F9,F4,F6,F8,F10	59
	W2	F7,F4,F11,F5,F8,F9,F10,F1,F6,F2,F3	61
	W3	F5,F1,F11,F2,F10,F3,F7,F8,F9,F4,F6	57
	W4	F5,F7,F10,F1,F2,F9,F11,F3,F4,F6,F8	62
	W5	F5,F3,F7,F1,F2,F10,F11,F8,F9,F4,F6	56
Manual		F1,F2 F3,F4,F5,F7,F8,F6,F9,F10,F11	66
Symbolic synthesis		Sum of costs of modules synthesized separately	69
Quartus II		Cost of MV- network	163

Next, we performed the optimization of the synthesis of MVL network from Fig. 1 using DGP method (DGP parameters: Ω = 50, α = 0.05, β = 0.7, γ = 0.25, λ = 10). Tab. 3 presents the results represented by the final generation of solutions (obtained in 5 trials). Each time, the best EIC was found not later than in the 10th generation, thus the algorithm stopped after about 20 generations. To verify the effectiveness of our estimation method, the results of synthesis for some best solutions (*B1-B5*) as well as middle (*M1-M5*) and worst (*W1-W5*) ones are given – the method seems quite accurate: the best results were obtained for the orders ranked as the best ones. We also performed the synthesis using a manual method as well as the synthesis in Quartus II. In both cases, the results were significantly worse than those obtained using the DGP optimization.

5. Conclusions

This paper presents the method of FPGA-based optimization of MVL networks, which determines the optimal or suboptimal order of modules that are synthesized using the symbolic synthesis. For this purpose the developmental genetic programming was applied. To increase the efficiency of our approach, a new method for estimation of FPGA implementation was developed. The experimental results showed that our approach minimizes the cost of the implementation in LUT-based FPGAs. For the sample MVL network, a reduction by 72% in comparison with a commercially available tool was obtained. Also the computation time for each synthesis was less than 5 seconds (on 1.7GHz CPU). These results showed that the symbolic decomposition with optimized ordering of modules is very efficient for the FPGA-based synthesis of MV logic networks.

6. References

[1] M. Gao, J.H. Jiang, Y. Jiang, Y. Li, A. Mishchenko, S. Sinha, T. Villa, R. Brayton: Optimization of multi-valued multi-level networks, Proc. of the 32nd IEEE Int. Symp. on Multiple-Valued Logic, 2002.

[2] S. Deniziak, M. Wisniewski: Symbolic functional decomposition of multivalued functions, Journal of Multiple-Valued Logic and Soft Computing, vol.24, no.5-6, pp.425-452, 2015.

[3] J.A. Brzozowski, T. Łuba: Decomposition of boolean functions specified by cubes, Journal of Multiple-Valued Logic & Soft Computing, vol. 9, no.4, pp. 377-417, 2003.

[4] L. Józwiak, A. Chojnacki: Effective and efficient FPGA synthesis through general functional decomposition, Journal of Systems Architecture, vol.49, issue 4-6, pp. 247–265, 2003.

[5] J.A. Brzozowski, J.J. Lou: Blanket algebra for multiple-valued function decomposition, Proc. of the Intern. Workshop on Formal Languages and Computer Systems, in. Algebraic Engineering, C.L. Nehaniv and M. Ito, eds. World Scientific, pp. 262-276, 1999,

[6] S. Deniziak, M. Wiśniewski: FPGA-based state encoding using symbolic functional decomposition, Electronics Letters, vol.46, no.19, pp.1316-1318, 2010.

[7] S. Deniziak, M. Wiśniewski: A symbolic RTL synthesis for LUT-based FPGAs, Proc. of the IEEE Symposium on Design and Diagnostics of Electronic Circuits & Systems, pp. 102-107, 2009.

[8] R. Keller, W. Banzhaf: The evolution of genetic code in genetic programming, Proc. of the Genetic and Evolutionary Computation Conf., pp. 1077–1082, 1999.

[9] S. Deniziak and K. Wieczorek: Evolutionary optimization of decomposition strategies for logical functions, Proc. of ICAISC 2012, Lect. Notes in Comp. Science, vol. 7269, 2012, pp. 182-189.

Received: 20.05.2016 *Paper reviewed* *Accepted: 01.07.2016*

Prof. of PŚk Stanisław DENIZIAK, DSc, PhD, eng.

He graduated from the Faculty of Electronics of the Warsaw University of Technology, defended his doctoral thesis in 1994, and habilitation in 2006. He is the head of the Division of Computer Science of the Kielce University of Technology. His research interests include design of embedded systems, Internet of things, logic synthesis for FPGAs. He is a member of IEEE and IEEE Computer Society.

e-mail: S.Deniziak@computer.org

Mariusz WIŚNIEWSKI, PhD, eng.

He graduated from the Faculty of Electrical Engineering of Kielce University of Technology, defended his doctoral thesis in 2010 at the Silesian University of Technology. Currently he is working as an Assistant Professor at the Department of Computer Science of Kielce University of Technology (Poland). His interests include programming engineering, designing of algorithms – particularly applicable in embedded systems and logic synthesis for FPGA.

e-mail: m.wisniewski@tu.kielce.pl

Karol WIECZOREK, MSc, eng.

He graduated of Faculty of Electrical Engineering of Kielce University of Technology in 2008. Currently he is working as Research Assistant in Department of Computer Science of the Kielce University of Technology. His research interests include evolutionary algorithms, methods of synthesis of logial functions. He prepares his doctoral dissertation in Silesian University of Technology.

e-mail: k.wieczorek@tu.kielce.pl