Stanisław DENIZIAK, Mariusz WIŚNIEWSKI, Konrad KURCZYNA
DEPARTMENT OF INFORMATION SYSTEMS, KIELCE UNIVERSITY OF TECHNOLOGY, POLAND
7 Tysiaclecia PP Ave., 25-314 Kielce, Poland

Optimization problems in the synthesis of multiple-valued logic networks

Abstract

The paper discusses some aspects of FPGA-oriented synthesis of multiple-valued logic (MVL) network, i.e. a network of modules connected by multiple-valued signals. MVL networks are built during high-level synthesis, as a source specification of logical systems or during re-synthesis of gate-level circuits. FPGA-oriented synthesis of MVL is based on decomposing modules into smaller ones, each fitting in one logic cell. In this paper, we show that the order, according to which the modules are decomposed, has a great influence on the efficiency of the synthesis. This paper presents the case study which demonstrates the above problem as well as some experimental results and conclusions.

Keywords: multiple-valued logic network, symbolic decomposition, FPGA, logic synthesis.

1. Introduction

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

The synthesis of MVL functions has been the subject of many previous studies. In [2] the symbolic functional decomposition of multiple-valued logic (MVL) functions, targeted at FPGA implementations, is proposed. There is shown that the integration of encoding and functional decomposition gives significant improvement of results of logic synthesis for FPGA devices. It should be noted that the symbolic decomposition may minimize the cost as well as the number of logic levels of the system. In both cases, the optimization handles the functions sharing the common inputs or outputs. However, connection of at least two modules causes that it is not possible to find optimal encoding for all functions sharing the same MV variable. Thus, it is necessary to decide which function should be optimized first. According to the optimization strategy, it is important to find the proper order of modules during the decomposition process.

In this paper, the case study of decomposition of an MVL network is presented. The target platform for implementation is a LUT-based FPGA device. We studied different orders of decomposition of modules to show the influence on the quality of synthesis as far as the cost of implementation is considered. We believe that the discussion would be helpful to develop an efficient method for synthesis of MVL networks.

The paper is organized as follows. The next section reviews the related work. In Section 3 the problem of FPGA-oriented synthesis of MV networks is presented. The example of the MVL network is described and a case study is discussed in Section 4. Finally, the conclusions are presented in Section 5.

2. Related works

The implementation of logic functions in LUT-based FPGAs consists of logic synthesis and technology mapping. The most efficient synthesis method for this type of FPGA device is the functional decomposition based on the blanket algebra [3] or on the theory of information relationship measures [4]. Logic synthesis for MVL functions [5] has attracted researches for many years. A lot of methods for input and output encoding [6], state assignment [7] as well as logic minimization [8] have been developed. The blanket algebra has also been generalized to MVL functions [9] and the symbolic decomposition [2] for MVL functions has been proposed. The method has successfully been applied to logical functions with MV inputs and outputs and FSMs giving significant improvements in the cost of FPGA implementations.

Several methods for synthesis of MVL networks are presented in literature. In [10], a method of synthesis of MVL networks using multiple-valued decision diagrams (MDD) is described. But the method is dedicated to gate implementations, moreover it does not consider encoding of multiple-valued variables. MDD representation is also used in MVSIS [11]. It is a set of interactive tools performing technology independent optimizations. The decomposition is performed using algebraic factorization, MVSIS also supports a method of MVL network encoding that minimizes the number of factors in multilevel implementation [12]. A discussion about possible optimizations of MVL networks is given in [13]. The authors claimed that optimization based on MVL could lead to the improved optimization quality. For all the above methods no experimental results concerning FPGA implementations are given.

3. 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 (expresses function F through functions G and H with disjoint sets of output variables) or in a serial fashion, which expresses F(X) through functions G and H, such that H(U, G(V)), where $U \cup V = X$

Assume that $X_i(Y_i)$ is a MV input(output). During the decomposition the variable may be encoded with variables $X_i'(Y_i')$ and $X_i''(Y_i'')$. Both variables may be separated during the decomposition. The proper encoding of MV variables leads to significant simplification of result functions [2].

4. Synthesis of MVL network – a case study

The FPGA-based optimization usually minimizes the number of LUT cells required for implementation. Therefore, the efficiency of decomposition has the greatest influence on the result of synthesis. Fig.1 presents the example of an MVL network, where *A*, *B*, *C*, *D*, *E*, *F* and *G* are MV signals, FS4, FS7 and FS8 are state variables, xi and yi 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 may be optimized only for one module sharing the same variable. For example, in Fig.1 optimal 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.

To find the best FPGA-based implementation of the MVL network, we should find the optimal order in which all modules will be decomposed. Implementation of one module may decrease possibilities of optimization for others. It should be noted that it is a typical optimization problem. For *n* modules we have *n!* possible orders, therefore for a large network only heuristic optimizations may be performed. An approach to this problem is presented in [14], but to obtain good results more sophisticated heuristics should be applied.

First, we performed separate decompositions of the following modules, using the symbolic decomposition [2] – there was used a research tool, developed for symbolic decomposition studies. The results, given in LUT4 cells, are as follows: F1, F3 – 6, F2, F10 – 3, F4, F11–7, F5, F7, F9–4, F6–9 and F8–16. Since we did not consider that modules shared variables, these results are given for reference only, i.e. to show the complexity of the modules.

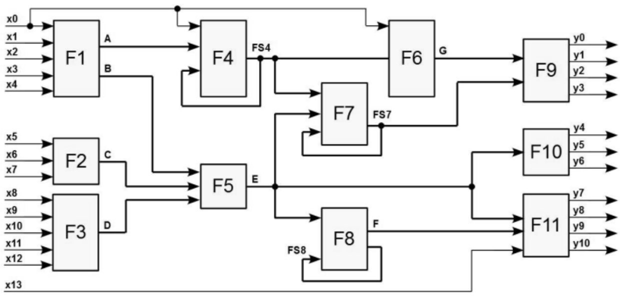

Fig. 1. The modules forming MVL network

Next, we performed decompositions of modules using many different orders (determined using a separate algorithm). Some results of synthesis are presented in Tab.1. The columns contain the best, the medium and the worst 5 results. The implementation cost was obtained using the symbolic functional decomposition [2]. LUTs with 4 inputs and 1 output were assumed as the target FPGA cells (given in the “Cost” column). It should be noted that many modern FPGAs are based on LUTs with more inputs, but LUT4 cells cause the greatest challenge during synthesis and require more sophisticated approach.

Tab. 1. Results of decompositions for example MVL network

No	The best results		The medium results		The worst results	
	Order	Cost	Order	Cost	Order	Cost
1	F6,F3,F9,F4,F8,F10,F11,F2,F7, F1,F5	47	F1,F6,F10,F2, F8,F9,F11,F3, F4,F5,F7	51	F5,F3,F11,F1, F2,F7,F9,F4, F6,F8,F10	59
2	F3,F6,F9,F2,F8,F10,F11,F4,F7, F1,F5	48	F10,F6,F9,F2, F7,F8,F11, F1,F3,F4,F5	52	F7,F4,F11,F5, F8,F9,F10, F1,F6,F2,F3	61
3	F2,F8,F11,F6, F10,F4,F9,F3,F7, F1,F5	48	F4,F8,F9,F3,F6, F10,F11,F1,F2, F5,F7	52	F5,F1,F11,F2, F10,F3,F7, F8,F9,F4,F6	57
4	F2,F6,F9,F4,F8,F10,F11,F1,F3, F5,F7	46	F9,F3,F8,F1,F2, F5,F10,F4,F6, F7,F11	52	F5,F7,F10,F1, F2,F9,F11, F3,F4,F6,F8	62
5	F8,F10,F11,F6, F9,F3,F4,F1,F2,F5,F7	47	F6,F1,F10,F4, F8,F9,F11,F2, F3,F5,F7	51	F5,F3,F7,F1, F2,F10,F11,F8, F9,F4,F6	56

We may observe that the best results were obtained when module F8 was decomposed before F5, F7 and F11. It seems obvious, because F8 is the most complex, thus it should be optimized first. Similar rules may be observed for other modules. But the above observation is not enough to find the optimal order of decompositions. First, sometimes the optimization of smaller modules may give better results than that of greater modules. Second, some modules may have strong connections with others, e.g. this happens in the case of F5. In such cases, decomposition of such a module may significantly reduce the possibilities of optimization of many others.

Let us consider the synthesis of module F5. This module has got MV variables only and is connected with 7 other modules. As was shown in Tab.1, the decomposition of module F5 at the first stage of the synthesis process causes that the final result of synthesis is not the best. In general, module F5 has great influence on the entire synthesis process. This is caused by complexity of logic, contained in the modules connected with F5. In this case, the connection F5 with F2 or F3 causes appearance of logical redundancy. This is visible after the decomposition process – Tab. 2 presents the optimized encoding of MV variables corresponding to orders F1/2/3, F5 and F5, F1/2/3.

Tab. 2. Decomposition of module F5: a) order F1/2/3,F5, b) order F5,F1/2/3

Module F5:

```
B  C  D   | E
B0 C0 D0  | E0
B1 C1 D1  | E1
B0 C1 D0  | E1
B1 C1 D2  | E2
B2 C2 D2  | E2
B1 C3 D3  | E3
B2 C4 D4  | E4
B0 C5 D5  | E0
B0 C6 D6  | E0
B0 C7 D7  | E0
B0 C0 D8  | E0
B0 C0 D9  | E0
B1 C0 D9  | E1
B0 C0 D10 | E0
```

(a) Serially decomposed:

- function G:

```
c0 b0 b1 | g0
0  0  0  | 0
0  1  0  | 0
0  0  1  | 1
1  1  0  | 1
```

- function H:

```
d0 c1 g0 b0 | E
0  0  0  0  | E0
0  1  0  1  | E1
0  1  0  0  | E1
1  1  0  1  | E2
1  0  1  0  | E2
0  1  1  1  | E3
0  0  1  0  | E4
0  0  0  1  | E1
```

(b) Signals encoded in other modules:

```
b0 b1 c0 c1 c2 c3 d0 d1 d2 d3 | e0 e1 e2
0  1  0  0  0  1  0  0  0  0  | 1  0  0
1  0  1  0  0  0  0  0  0  1  | 0  1  0
0  1  1  0  0  0  0  0  0  0  | 0  1  0
1  0  1  0  0  0  1  1  0  0  | 1  1  1
0  0  0  0  0  0  1  1  0  0  | 1  1  1
1  0  0  0  1  1  1  1  1  1  | 0  0  0
0  0  0  0  1  0  1  0  1  1  | 0  0  0
0  1  0  1  1  0  0  0  1  0  | 1  0  0
0  1  0  1  1  1  0  0  1  1  | 1  0  0
0  1  0  1  0  1  0  1  1  0  | 1  0  0
0  1  0  0  0  1  0  1  1  1  | 1  0  0
0  1  0  0  0  1  1  0  0  1  | 1  0  0
1  0  0  0  0  1  1  0  0  1  | 0  1  0
0  1  0  0  0  1  0  1  0  0  | 1  0  0
```

After decomposition of F5 (Tab. 2(a)) the encoding causes that signal *D* will be reduced to one bit, signals *B* and *C* need 2 bits for encoding each, and *E* needs 3 bits. The implementation cost of F5 is 4 LUT4. In the case presented in Tab. 2(b), the signals *A, B, C* and *E* were encoded at the earlier stages of MVL network synthesis, so there was not MV signals in module F5. After binary synthesis, this module needs 3 LUT cells for implementation. It should be noted that the synthesis of module F5 also affects the synthesis of two other modules connected with it, i.e. F7 and F8.

It was also observed that FSM modules have the great influence on the synthesis of MVL networks. In most cases, this kind of modules should be decomposed first. Of course there are exceptions from this rule. However in the case of symbolic decomposition, the synthesis of FSM has some limitations, and good result were obtained if FSM was decomposed at as early stage as possible. At this point it should be noted that the modules may be grouped. For example, module F5 and its neighbors (F1, F2, F3, F7 and F8) have the great influence on the synthesis. In this set, two modules are FSMs. So, according to the above observation, the decomposition should be first performed for F7 or F8. In this case, grouping of modules allows exclusion of FSMs from decomposition of F5, thus F7 and F8 can be optimized separately.

Tab.3 shows the comparison of decomposition when FSM F8 was decomposed as the first or as the last. In Tab. 3(c) the first case is presented. The module is optimized as the first, so the full optimization was performed. After parallel decomposition, the binary variables e0, e1 and e2 were obtained as the encoding of MV signal E. The signal FS8 (state variable) was encoded through state encoding [2] which resulted in the creation of binary inputs (s0...s5) and binary outputs (s0'...s5') corresponding to the next state. The MV output could be encoded using any 3 bits long binary code. The total cost of implementation of module F8 is 16 LUT cells. When module F8 was decomposed as the last, only optimization based on state encoding was possible (Tab. 3(b)). In this case, the cost of 21 LUTs was obtained.

5. Conclusions

In this paper, we have presented the case study showing the optimization problems of the FPGA-based synthesis of MVL networks. We have shown that the final cost of implementation strongly depends on the order in which the following modules are decomposed. We have discussed some cases and we have made a few observations that may be useful for determining the optimal order of decompositions. But it seems that more sophisticated, heuristic method should be developed for this purpose.

Tab. 3. Decomposition of modules: F5, F7 and F8

(a) Module F8:

E	FS8	FS8'	F
E0	S0	S1	F0
E1	S1	S2	F1
E3	S2	S4	F2
E4	S4	S0	F3
E4	S5	S0	F4
E4	S6	S0	F5
E4	S7	S0	F6
E4	S8	S1	F0
E4	S9	S1	F0
E4	S10	S1	F0
E4	S11	S1	F0
E4	S12	S2	F1
E4	S13	S2	F1
E4	S14	S2	F1
E4	S15	S3	F2
E4	S16	S0	F0
E1	S0	S2	F1
E2	S0	S5	F6
E2	S9	S6	F6
E2	S2	S7	F6
E2	S3	S8	F6
E2	S4	S9	F6
E2	S5	S10	F0
E2	S6	S11	F0
E2	S7	S12	F0
E2	S8	S13	F0
E2	S10	S14	F0
E2	S11	S15	F0
E2	S12	S16	F0

(b) State encoding only (21 LUT cells):

e0	e1	e2	FS8	FS8'	f0	f1	f2
0	0	0	S0	S1	0	0	0
0	0	1	S1	S2	1	0	1
1	0	1	S2	S4	0	0	1
1	1	0	S4	S0	1	0	0
1	1	0	S5	S0	0	1	1
1	1	0	S6	S0	0	1	0
1	1	0	S7	S0	1	1	1
1	1	0	S8	S1	0	0	0
1	1	0	S9	S1	0	0	0
1	1	0	S10	S1	0	0	0
1	1	0	S11	S1	0	0	0
1	1	0	S12	S2	1	0	1
1	1	0	S13	S2	1	0	1
1	1	0	S14	S2	1	0	1
1	1	0	S15	S3	0	0	1
1	1	0	S16	S0	0	0	0
0	0	1	S0	S2	1	0	1
1	0	0	S0	S5	1	1	1
1	0	0	S9	S6	1	1	1
1	0	0	S2	S7	1	1	1
1	0	0	S3	S8	1	1	1
1	0	0	S4	S9	1	1	1
1	0	0	S5	S10	0	0	0
1	0	0	S6	S11	0	0	0
1	0	0	S7	S12	0	0	0
1	0	0	S8	S13	0	0	0
1	0	0	S10	S14	0	0	0
1	0	0	S11	S15	0	0	0
1	0	0	S12	S16	0	0	0

(c) Parallel decomposed, state encoding (16 LUT cells):

e0	e1	e2	s0	s1	s2	s3	s4	s5	s0'	s1'	s2'	s3'	s4'	s5'	F
1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	F0
0	1	0	0	0	0	0	0	1	1	1	0	1	1	1	F1
0	0	0	1	1	0	1	1	1	1	1	0	1	0	1	F2
0	0	0	1	1	0	1	0	1	0	0	0	0	0	0	F3
0	0	0	1	0	0	1	0	1	0	0	0	0	0	0	F4
0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	F5
0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	F6
0	0	0	1	1	0	0	0	0	0	0	0	0	0	1	F0
0	0	0	1	1	0	1	1	0	0	0	0	0	0	1	F0
0	0	0	1	0	0	1	1	1	0	0	0	0	0	1	F0
0	0	0	0	1	0	1	1	0	0	0	0	0	0	1	F0
0	0	0	0	1	0	0	1	0	1	1	0	1	1	1	F1
0	0	0	0	0	0	0	1	0	1	1	0	1	1	1	F1
0	0	0	0	0	0	1	1	0	1	1	0	1	1	1	F1
0	0	0	0	1	0	0	0	1	0	0	0	0	1	1	F2
0	0	0	0	1	0	0	1	1	0	0	0	0	0	0	F0
0	1	0	0	0	0	0	0	0	1	1	0	1	1	1	F1
1	1	1	0	0	0	0	0	0	1	0	0	1	0	1	F6
1	1	1	1	1	0	1	1	0	1	0	0	1	0	0	F6
1	1	1	1	1	0	1	1	1	1	0	1	1	0	0	F6
1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	F6
1	1	1	1	1	0	1	0	1	1	1	0	1	1	0	F6
1	1	1	1	0	0	1	0	1	1	0	0	1	1	1	F0
1	1	1	1	0	0	1	0	0	0	1	0	1	1	0	F0
1	1	1	1	0	1	1	0	0	0	1	0	0	1	0	F0
1	1	1	1	1	0	0	0	0	0	0	0	0	1	0	F0
1	1	1	1	0	0	1	1	1	0	0	0	1	1	0	F0
1	1	1	0	1	0	1	1	0	0	1	0	0	0	1	F0
1	1	1	0	1	0	0	1	0	0	1	0	0	1	1	F0

6. References

[1] Gao M., Jiang J.H., Jiang Y., Li Y., Mishchenko A., Sinha S., Villa T., Brayton R.: Optimization of Multi-Valued Multi-Level Networks, 32nd IEEE Int. Symp. on Multiple-Valued Logic, 2002.Deniziak S., Wisniewski M.: Symbolic Functional Decomposition of Multivalued Functions. Journal of multiple-valued logic and soft computing, 01/2015; 24(5-6), pp. 425-452.

[2] Brzozowski J.A., Łuba T.: Decomposition of Boolean Functions Specified by Cubes, Journal of Multiple-Valued Logic & Soft Computing, (2003) vol. 9, pp. 377-417.

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

[4] Brayton R.K., Khatri S.P.: Multi-valued Logic Synthesis. Proc. of the Int. Conference on VLSI Design, 1999, pp.196-205.

[5] Saldanha A., Villa T., Brayton R.K., L.Sangiovanni-Vincentelli A.: Satisfaction of input and output encoding constraints, IEEE Transactions on CAD, vol.13, 1994, no.5, pp.589–602.

[6] Ashar P., Devadas S., Newton A.R.: Sequential Logic Synthesis. Norwell: Kluwer Academic Pub., 1992.

[7] Malik S., Lavagno L., Brayton R.K., Sangiovanni-Vincentelli A. Symbolic minimization of multilevel logic and the input encoding problem. IEEE Trans. on CAD, 1992, vol.11, no.7, pp. 825–843.

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

[9] Drechsler R., Thornton M., Wessels D.: MDD-Based Synthesis of Multi-Valued Logic Networks. Proc. of ISMVL, 2000.

[10] Gao M., Jiang J.-H. R., Jiang Y., Li Y., Mishchenko A., Sinha S., Villa T., Brayton R.K.: Optimization of multi-valued multi-level networks. Proc. ISMVL, 2002, pp.168-177.

[11] Jiang J.H.R., Jiang Y., Brayton R. K.: An Implicit Method for Multi-Valued Network Encoding. Proc. of IWLS, 2001.

[12] Mishchenko A., Brayton R.K.: A Boolean Paradigm in MultiValued Logic Synthesis. Proc. IWLS`02, 2002.

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

Received: 23.02.2016 *Paper reviewed* *Accepted: 04.04.2016*

Stanisław DENIZIAK, MSc, PhD, DSc, eng.

He graduated from Faculty of Electronics of the Warsaw University of Technology, defended his doctoral thesis in 1994, and habilitation in 2006. He is a professor of Kielce University of Technology, and the head of the Division of Computer Science of the University. 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, MSc, PhD, eng.

He graduated from Faculty of Electrical Engineering of Kielce University of Technology, defended his doctoral thesis in 2010 in the Silesian University of Technology. Currently he is working as a Assistant Professor in 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

Konrad KURCZYNA, MSc, eng.

He graduated from Faculty of Electrical Engineering of Kielce University of Technology. Currently he is working as a lecturer in Department of Computer Science, Kielce University of Technology, Poland. His research interests includes techniques of programming engineering, embedded systems and logic synthesis for FPGA devices.

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