Piotr BIBILO, Wlodzimierz ROMANOW
ZJEDNOCZONY INSTYTUT PROBLEMÓW INFORMATYKI NAN BIAŁORUSI

Transformation of the RTL-description for the area minimization of circuits implemented in FPGA

Dr. Piotr BIBILO

In 1980 Piotr Bibilo defended his PhD thesis. In 1992 he defended his doctor of science thesis "Decomposition methods for the synthesis of digital devices programmable matrix circuits". Since 1994 he has been heading the logic design laboratory. Main scientific results of P. Bibilo are methods for designing digital control systems and methods for designing custom VLSI circuits.

e-mail: bibilo@newman.bas-net.by

Dr inż. Wlodzimierz ROMANOW

Vladimir Romanov graduated from Belarus State University (Minsk) in 1976. In 1986 he defended PhD thesis "Optimization of use of resources in the problem-oriented dialog programming systems" Now his research interests are development of the managing programs for systems of logic designing and creation of man-machine interfaces of the large program complexes.

e-mail: rom@ newman.bas-net.by

Abstract

The problem of transforming RTL-descriptions of Boolean function systems to obtain the simpler combinational logical circuits implemented in FPGA at redesign in LeonardoSpectrum system is considered. An optimization algorithm is suggested. The results of experimental investigation are given that show the efficiency of the suggested algorithm.

Keywords: FPGA, VHDL, resynthesis, combinational circuit.

Streszczenie

Referat omawia problem przekształcenia opisów systemów funkcji logicznych na poziomie RTL w celu uzyskania prostszych układów kombinacyjnych zaimplementowanych w FPGA przez ponowne projektowanie przy pomocy LeonardoSpectrum. Został zaproponowany algorytm optymalizacji. Przytoczone są wyniki eksperymentów, demonstrujące skuteczność algorytmu.

Słowa kluczowe: FPGA, VHDL, ponowna synteza, układ kombinacyjny.

1. Introduction

The synthesis of logical FPGA (Field-Programmable Gate Arrays) circuits can be carried out in various methods [1,2] and systems (synthesizers), among which the most commonly used are XST synthesizer of WebPack ISE design system, Synplify, and LeonardoSpectrum (below, Leonardo). Apart from FPGA and CPLD (Complex Programmable Logic Devices) circuits, Leonardo synthesizer allows carrying out the synthesis of circuits in the synthesis library given by the designer indirectly [3]. Such circuits can be implemented as the parts of ASICs (Application-Specific Integrated Circuits).

Below, we mean the used synthesizer by Leonardo. The synthesizer can optimize circuits using various criteria (area, delay) and take into account various technological restrictions. The synthesis of circuits from algorithmic descriptions in VHDL, Verilog language is divided into two stages: the high level synthesis whose result is so called RTL (Register Transfer Level) description, and the technology mapping [4]. RTL-description can be obtained in Leonardo (it differs advantageously from other synthesizers by this) from the structural description of the synthesized logical circuit by means of „unmap" instruction. We call that description RTL0. Naturally, RTL0 description is identical functionally to the initial algorithmic VHDL description. The *resynthesis* of a FPGA circuit can be carried out using RTL0 description. It was noticed in the design practice that the newly constructed circuit has often better characteristics than the circuit constructed using the initial VHDL description. So, Leonardo synthesizer allows decreasing the complexity of the circuit by repeated (iterative) synthesis. But such reduction of the circuit complexity does not always take place. Because of this, an algorithm for transformation of RTL descriptions allows often decreasing the complexity of synthesized FPGA circuits. We consider FPGA circuit complexity at the stage of logical synthesis as the number of programmable FPGA – LUT (Look Up Table) elements.

2. Transformations of RTL-descriptions

Leonardo synthesizer and the control of the process are described in detail in [3]. Let us consider in more details the iterative repeated synthesis from algorithmic VHDL descriptions of combinational circuits (only such circuits are considered in the paper). There are only the operators of setting signals (<=) and logical operators: *NOT* (negation), *AND* (conjunction), *OR* (disjunction), *XOR* (exclusive *OR*).

The decrease of complexity using the repeated synthesis is based on widening of logical equations that are in RTL descriptions, i.e. on elimination of some internal variables or on change of the form of representation of internal or output variables. Further, we suggest just such an algorithm for transforming RTL descriptions, the following designations of logical operations are being used to write down logical equations: * (conjunction), + (disjunction), ^ (negation). Operation (*a XOR b*) is replaced by expression (*^a*b + a*^b*). The suggested algorithm is implemented in computer program and included in LOCON system [5] of hardware implementation of parallel algorithms of logical control and in SiVer system [6] of synthesis and verification of combinational circuits.

Let *X*, *Y* and *Z* be the sets of input, output and internal variables of a logical circuits correspondingly. Each equation in RTL description gives one of the variables $v \in Y \cup Z$ in the form of a logical equation. The algorithm for transforming the equations may be represented as follows.

Grouping initial equations in clusters, every of which is represented by one equation depending at most on *B* variables. By that, we call the variables indicated in the right part of an equation the arguments of the equation. The number *B* constrains the quantity of the arguments of an equation and is a *parameter* of the algorithm. After that reordering the obtained system of clusters in such a way, that for *i*-th equation, all internal variables used in it as arguments are determined in the set of equations whose numbers are less than *i*.

The sum of the numbers of input pins of *AND* gates, *OR* gates and inverters in the circuit constructed directly from a cluster is taken as the *complexity estimation* of the cluster.

The base scheme of the algorithm for grouping (stage 1) can be represented by the sequence of steps 1.1 – 1.3:

1.1. Constructing the list of variables *list_C*, for which the clusters will be created. First, all output variables are included into that list.

1.2. Successive looking through all the variables from *list_C* and for every of them, the procedure of constructing the cluster is executed, during which the corresponding equation of initial system is modified and, perhaps, *list_C* is widened.

1.3. Algorithm finishing its work when there is no variable which is not looked through.

In the framework of the procedure of constructing a cluster (step 1.2), the list of variables (*list_E*) is constructed that play the role

of arguments of the formed cluster. First, that list is empty. The procedure is fulfilled on the base of the following algorithm (steps 1.2.1 – 1.2.5):

1.2.1. All variables that are the arguments of the equation determining the variable under consideration are included in *list_E*.

1.2.2. If there are no internal variables in *list_E*, constructing the cluster comes to the end.

1.2.3. All internal variables from *list_E* are looked through and for each of them, the possibility of "substitution" is checked. The substitution is reduced to the two actions:

- replacing all occurrences of the name of the represented variable by the expression from the right part of the equation that determines this variable;
- replacing the name of the represented variable in *list_E* by the list of the arguments of the equation that determines this variable.

The possibility of the substitution is determined by the condition $|list\_E+| \le B$ where *list_E+* determines the contents of *list_E* after fulfilling the substitution. The cardinality of a set A is designated by $|A|$. It must be noted that every variable from *list_E* is always presented as a single specimen. It is kept track of at the stage of widening *list_E*. The variables are placed in *list_E* in lexicographic order.

1.2.4. If the substitution is possible for some variable from the list, it is fulfilled and the transition to step 1.2.2 is carried out.

1.2.5. If the substitution is possible for no variable from *list_E*, constructing the cluster comes to end and all internal variables remained in *list_E* are carried to *list_C*.

Let us consider an example of transforming RTL description by the suggested algorithm. Let the initial RTL description (*X={a, b, c, d}* is the set of input variables; *Y={y1, y2, y3, y4}* is the set of output variables; *Z={z1, ..., z8}* is the set of internal variables) contains 11 equations and is as follows:

```
z4 = a * ^c;
z5 = a * b + ^a * b;
z6 = b + d;
z7 = c * d;
z3 = z4 * ^b + ^z4 * b;
z2 = ^z6;
z1 = ^z8 + d;
z8 = z3 * z5;
y1 = z1 + z2;
y2 = z8;
y3 = z6 * ^z7;
```

Executing the algorithm for B = 3 looks as follows:

```
y1 = z1 + z2;  → y1 = (^z8 + d) + z2;  →
y1 = (^z8 + d) + ^z6  →
y1 = (^z8 + d) + ^(b + d);
y2 = z8;  →  y2 = (z4 * ^b + ^z4 * b) * z5;  →
y2 = (z4 * ^b + ^z4 * b) * (a * b + ^a * b);  →
y2 = ((a * ^c) * ^b + ^(a * ^c) * b) * (a * b + ^a * b);
y3 = z6 * ^z7;  →  y3 = (b + d) * ^z7;  →
y3 = (b + d) * ^(c * d);
z8 = z3 * z5;  →  z8 = (z4 * ^b + ^z4 * b) * z5;  →
z8 = ((a * ^c) * ^b + ^(a * ^c) * b) * z5;  →
z8 = ((a * ^c) * ^b + ^(a * ^c) * b) * (a * b + ^a * b);
```

The described above base algorithm for grouping was modified made to be adjustable parametrically. The parameters determining the peculiar properties of the algorithm behavior are the group of four binary variables: **R**eduction, **I**ntersection, No_**M**erge, **O**pen.

1. ***R**eduction = TRUE*. In this case, before grouping, the initial system of equations is reduced by elimination of the equations that have only one argument. In the remaining equations, the occurrences of the eliminated variables are replaced by the right parts of the corresponding equations.

2. ***I**ntersection=TRUE*. In this mode, point 1.2.5 is executed differently. If the substitution of no internal variable from *list_E* is possible, the possibility of their joint substitution is checked in addition

3. *No_**M**erge = TRUE*. In this case, in executing point 1.2.3, the condition of the possibility of substituting an internal variable is modified: in addition to the restriction on the number of arguments of the formed cluster, the substitution of the variables that are arguments of more than one equation of the system is not allowed.

4. ***O**pen = TRUE*. This mode is used only in common with the previous one and serves to its additional determination: the substitution of variables that are in more than one equation is allowed still for those of them which depend only on input variables.

Combining the values of the listed parameters, one can obtain different variants of the algorithm execution (Table 1).

Tab. 1. Użycie parametrów w algorytmu
Tab. 1. Using parameters in algorithms

Parameter \ Conventional name of algorithm at a given set of parameters (B = 4, 5)	*B*	*Bi*	*Br*	*Bri*	*Bim*	*Bimo*	*Brim*	*Brimo*
Reduction	–	–	+	+	–	–	+	+
Intersection	–	+	–	+	+	+	+	+
No_**M**erge	–	–	–	–	+	+	+	+
Open	–	–	–	–	–	+	–	+

Sign + in the table shows that the corresponding parameter takes the value *TRUE*, and it determines the actions that are used in the variant of the algorithm. For example, algorithm "*5imo*" does not use only the transformations designated as Reduction.

3. Experimental investigation

To estimate the possibility of reducing the complexity of FPGA circuits the experiments 1 – 4 were carried out.

Experiment 1 (base). Synthesis of FPGA circuits using initial VHDL descriptions.

Experiment 2 (iter). Repeted iterative synthesis when five iterations (RTL1, …, RTL5) of synthesis of a FPGA circuit are executed beginning with RTL0 description of an initial circuit. The best solution selected from all the five iterations is designated below by *best_iter*.

Experiment 3 (re_RTL0). Repeated synthesis using the suggested algorithm from RTL0 descriptions obtained after the synthesis of a FPGA circuit from its initial VHDL description.

Experiment 4 (re_best_iter). Repeated synthesis using the suggested algorithm from the best (of the least complexity) RTL descriptions *best_iter* that instruction "*unmap*" is applied to.

Target chip FPGA. The chip XC2S100 of SPARTAN-II family was used as the target chip FPGA [7]. When Experiments 3 and 4 were carried out, the values of parameter *B* were supposed to be equal to 4 and 5 because there are LUTs with the number of input pins 4 and 5 consisting in the target chip.

Initial data. 20 examples of PLA circuits from the library [8] were selected to carry out the experiments. 7 examples (PLA circuits) from the class "*mathematical*" and 13 examples from the class "*industrial*" were included in the experimental set.

Optimization criteria and controlling the synthesis. In all the cases, the synthesis was carried out equally: the optimization criterion was the area of the circuit and the control of the synthesis of the circuits was fulfilled equally by scripts [3].

4. Experimental results

The results of Experiment 1 are given in Table 2. The following designations are used in Table: *n* is the number of inputs of a PLA circuit, *m* is the number of outputs, *k* is the number of intermediate wires (the number of terms in the implemented system of DNFs of Boolean functions); S_{FPGA} is the complexity of a FPGA circuit (the number of LUTs). The best solutions obtained in the corresponding experiments are in Table 2 (and other tables) in bold.

Tab. 2. Wyniki eksperymentów 1 i 2
Tab. 2. Results of Experiments 1 and 2

N	Name of	n	m	k	Exp. 1	Exp. 2