Software package for optimizing digital circuits[1]

Maxim Gromov, Natalya Kushik

***Abstract*—In this paper, we describe a software tool for optimizing the path length from primary inputs to primary outputs as well as the number of gates in digital circuits. A frame for optimization is extracted from a digital circuit; the extracted frame is divided into two parts and the maximum flexibility for each part is determined by the largest solution to an appropriate FSM equation. We check whether one or some output functions of each part can be replaced by a simple function of two primary input variables that can be implemented as a single gate, while preserving the behavior of the overall fragment. A developed software package can deal with digital circuits which have around 500 gates, and 40 primary inputs and outputs. Experiments were performed for a pack of benchmarks that were first resynthesised by ABC tool [1]. Our results show that the developed package can improve around 15% of benchmarks.**

***Index Terms*—Digital circuit, FSM equation, optimization**

I. Introduction

THE complexity of digital circuits increases quickly and there still are no tools which can guarantee the design of an optimal circuit. For this reason, usually optimization tools run for already designed circuit. There are a number of optimization criteria such as reliability, fault-tolerance, minimal number of communication lines, delay, area etc. The problem of optimal design remains a challenging problem for developing new information technologies. The best approach for the optimization has been shown to be an iterative component optimization that can be based on solving an appropriate Finite State Machine (FSM) equation. A largest solution, i.e. the solution with maximum flexibility can be viewed as a reservoir for all possible optimizations of a frame of interest, from which an optimal frame implementation can be chosen. However, the complexity of solving an FSM equation generally is exponential in the number of states of its coefficients (FSMs). For this reason, a so-called window approach for optimizing digital circuits is used for optimization [2]. We iteratively extract a frame of an appropriate size from a given digital circuit, divide it into two parts and optimize these parts with respect to the given criteria. The procedure terminates when we are satisfied with the optimization results.

In this paper, when optimizing a circuit, we extract a combinational frame and then divide it into two component circuits (head and tail components) and optimize them based on the idea that in general, a number of combinational circuits can replace a head (or a tail) component without changing the behavior of the overall frame. All permissible replacements are represented as a nondeterministic circuit [3] that is derived as the largest solution to an appropriate FSM equation. For each primary output of a circuit component, we check whether the corresponding output function can be replaced by a simple function of two input variables. In this case, this function can be implemented by a single gate. Correspondingly, in the frame, length of some paths from primary inputs to primary outputs can be shortened as well as some gates can be deleted, i.e., there is a chance that the number of gates in the frame can be reduced.

The structure of the paper is as follows. Section II contains preliminaries. Section III is devoted to solving an equation for the head and the tail component. Section IV discusses software for optimizing digital circuits. Section V describes experimental results while Section VI concludes the paper.

II. Preliminaries

In this paper, we use a *behavioral function* in order to represent a digital circuit behavior. For a combinational circuit the behavioral function Ψ is defined over input and output variables of the circuit and $\Psi(\mathbf{x}, \mathbf{y}) = 1$ if and only if the circuit produces the output vector $\mathbf{y}$ to the input vector $\mathbf{x}$. Consider the combinational circuit in Figure 1.

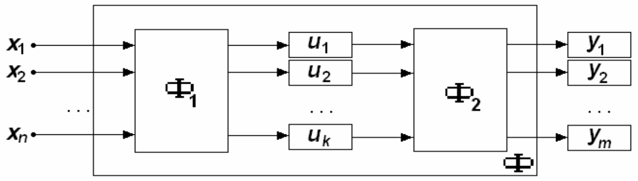

Figure 1. The combinational composition of two circuits

The circuit implements a system of Boolean functions Φ (an SBF Φ) and can be described by a corresponding behavioral function $\Psi_\Phi(x_1, \ldots, x_n, y_1, \ldots, y_m)$: $\Psi_\Phi(X_1, \ldots, X_n, Y_1, \ldots, Y_m) = 1$ if and only if $Y_1 = \varphi_1(X_1, \ldots, X_n), \ldots, Y_m = \varphi_m(X_1, \ldots, X_n)$. We

[1] This work is partly supported by RFBR grant N 07-08-12243 and by RFBR-NSC grant N 06-08-89500.

say that a function Ψ is an SBF-behavioral function if Ψ is a behavioral function of some system of Boolean, possibly non-deterministic, functions. Given a Boolean function θ, we denote M_θ^1 the set of variable values, for which the function equals 1. Given Boolean functions θ and Ψ such that $M_\theta^1 \subseteq M_\psi^1$, we denote this fact as $\theta \leq \Psi$. The head component implements the SBF Φ_1; the behavioral function Ψ_{Φ_1} of the head component is specified over the set $\{x_1, \dots, x_n, u_1, \dots, u_k\}$ of variables and we extend it over the set of variables $\{y_1, \dots, y_m\}$. The tail component implements the SBF Φ_2 and the behavioral function Ψ_{Φ_2} of the tail component is specified over the set $\{u_1, \dots, u_k, y_1, \dots, y_m\}$ of variables and we extend it over the set of variables $\{x_1, \dots, x_n\}$. The behavioral function Ψ_Φ of the overall circuit which implements the SBF $\Phi = \Phi_2(\Phi_1)$ is specified over the set $\{x_1, \dots, x_n, y_1, \dots, y_m\}$ of variables and $\Psi_\Phi = (\Psi_{\Phi_1} \wedge \Psi_{\Phi_2})_{\downarrow \mathbf{x},\mathbf{y}}$.

In order to optimize the head or the tail component of the frame we should replace a circuit component with another one preserving the external behavior of the composition. All such replacements are captured by a largest solution to a corresponding FSM equation. According to optimization criteria, an optimal circuit can be then extracted from a largest solution. In this paper, for each circuit component, we study whether it is possible to replace a component with another circuit which has less number of gates or has shorter paths from primary inputs to primary outputs.

III. Solving an Equation over the Head Component and the Tail Component

A. *Solving an equation over the head component*

The most flexibility for the head component can be captured by the largest solution to a corresponding FSM equation $\overline{(\Psi_{\Phi_2} \wedge \overline{\Psi_\Phi})_{\downarrow \mathbf{x},\mathbf{u}}}$, where Ψ_Φ is extended over the set $\{u_1, \dots, u_k\}$ of variables and a digital circuit that implements the SBF Φ_3 can replace the head component if and only if $\Psi_{\Phi_3} \leq \overline{(\Psi_{\Phi_2} \wedge \overline{\Psi_\Phi})_{\downarrow \mathbf{x},\mathbf{u}}}$ [4], where $\overline{\varphi}$ is the inversion of the function φ.

The above statement gives a guide how to determine an SBF that can replace SBF Φ_1 without changing the behavior of the overall system. We, thus, check whether one or more functions of the head component can be selected as functions of two input variables or as functions equal to the constant 1 (or to the constant 0) preserving all other functions. In this case, this output functions can be implemented by a single gate and all the gates of the path from inputs to a corresponding output which do not influence other output functions, can be deleted from the head component.

B. *Solving an equation over the tail component*

The set of all permissible behaviors of the tail component can be captured by a partial FSM that is defined only for *u*-patterns which are output patterns of the head component. Thus, in order to get *u*-inputs where the behavior of the tail component cannot be changed we take the projection $(\Psi_{\Phi_1} \wedge \Psi_{\Phi_2})_{\downarrow \mathbf{u},\mathbf{y}}$. This function is not really a behavioral function, since it describes only a part of behavior. If for some *u*-pattern there is no *y*-pattern in the set M_ψ^1 then the behavior of the tail component for this *u*-pattern can be selected in an arbitrary way (so-called input don't care conditions). So we consider $(\Psi_{\Phi_1} \wedge \Psi_{\Phi_2})_{\downarrow \mathbf{u},\mathbf{y}}$ as a largest solution for the tail component and check whether there exits y_i, $i = 1, \dots, m$, that can be replaced by a function of two input variables or by a function equal to the constant 1 or constant 0.

In our software we use Binary Decision Diagrams (BDD) [5] for all operators over Boolean functions. We use operators of the BDD package that is well known and is widely used when manipulating with digital circuits.

IV. Software

In this section, we briefly describe the software package that is developed for optimizing digital, possibly sequential circuits. At the first step, a combinational frame up to 100 gates and 20-23 inputs is extracted. At the second step this frame iteratively is divided into two sequential parts which are optimized according to the above description and if the optimization occurs a component is replaced by a better implementation, the frame is divided again into two parts etc. The procedure terminates when we run out of time or are satisfied with the optimization results.

A. *Circuit representation*

In our software package, we represent a sequential circuit given in the bench format as a set of connected gates with integer numbers. Each number uniquely identifies a gate. Correspondingly, the information of all gate predecessors (or successors) is represented by a Boolean matrix. The optimization process relies only on integer arrays: all the operations such as extracting a frame, optimizing a component, composing two circuits after optimization result also take place in integer arrays. Only at the last step this representation is back converted into the benchmark format (bench format). The use of such (hash) representation accelerates the optimization process compared with the representation where original strings of gate names are used without hashing.

When operating with behavioral functions BDDs are of a big help. All the operations such as deriving the behavioral function for a circuit, given in the bench format, deriving the largest solution, checking whether one or several output functions can be selected as constants (1 or 0), checking whether an output function can be a simple function of two input variables are performed fast enough for circuits which have up to 50 input and output variables. We use CUDD-package to calculate a largest solution as BDD for the tail component and transform it into a sum of products, as in this case, such representation seems to be more convenient than BDD representation.

B. *Main methods of the software package*

Frame extraction. When extracting a frame we need to keep an eye on the correspondence between inputs and outputs of the extracted frame and gates of the initial circuit. We extract a frame without combinational loops and for this reason, we first order the combinational part of the initial circuit by layers depending on their distance from primary inputs and flip-flop outputs. If there are n layers then we extract a frame as the set of all gates which belong to layers j, $j + 1, \ldots, k$, $1 \le j \le k \le n$.

Deriving a behavioral function for a non-deterministic circuit that is the largest solution for the head component. We use the BDD package in order to derive a behavioral function for each component. The largest solution then is obtained by BDD manipulation. Using the BDD representation of the largest solution each output function is checked whether it can be replaced by a constant or by a simple function of two input variables preserving the behavior of the overall composition.

Deriving a behavioral function for a non-deterministic circuit that is the largest solution for the tail component. For the tail component BDD representation of the largest solution is converted into a sum of products, as this representation seems to be more convenient for dealing with a system of partially specified Boolean functions.

Optimization. If one or several output functions of a head (or tail) component can be replaced by a constant or by a simple function of two input variables then a corresponding gate is added to the component and all gates of the initial component which do not influence other outputs are taken away.

Insert operator is used for inserting the optimized component into the frame and then for inserting the obtained frame into the initial circuit.

V. Experimental results

We have conducted experiments using the proposed method with some benchmarks [6] in order to see how often our package can reduce the number of gates and the length of a path from primary inputs to primary outputs for a given combinational circuit. We used ABC for logic synthesis and verification. A given benchmark was first synthesized as a logical circuit using ABC and our package was used for the circuit optimization. Extracted frames have up to 25 inputs and path length from primary inputs to primary outputs varies from 5 to 19 being 10 on average. Ten functions of two variables, such as AND, OR, etc., were used for optimization; all of them can be easily implemented by a single gate. The results show that the developed package can improve around 15% of benchmarks. The optimization is not huge but on the other hand, those benchmarks were already optimized many times using other packages.

VI. Conclusions

In this paper, we described the software tool for optimizing the number of gates in digital circuits as well as the path length from primary inputs to primary outputs. A combinational frame extracted from a digital circuit is divided into two components. Each component then is optimized independently. We experimented on some benchmarks from [6] and our results clearly show that there exist a number of benchmarks such as s838.bench, s298.bench and s420.bench, etc. for which our package returns optimized circuits. More experiments with new benchmarks are needed in order to estimate the efficiency of the developed package.

References

[1] Berkeley Logic Synthesis and Verification Group, ABC: A System for Sequential Synthesis and Verification, http://www.eecs.berkeley.edu/alanmi/abc/

[2] S.Zharikova, M.Vetrova, N.Yevtushenko Optimization of a multi component digital circuit by solving a system of FSM equations // Proceedings Euromicro Symposium on Digital System Design Architectures, Methods and Tools, IEEE Computer Society. – Belek-Antalya, Turkey, 2003, pp. 62-68.

[3] A. Mishchenko, R. Brayton, R. Jiang, T. Villa, and N. Yevtushenko, "Efficient solution of language equations using partitioned representations", Proc. DATE, 2005, pp. 412-417.

[4] N. Kushik, G. Sapunkov, S. Prokopenko, N. Yevtushenko. Minimizing path length in digital circuits based on equation solving. In Proc. of IEEE EAST-WEST design&test symposium, October, 2008, pp. 365-370.

[5] CUDD [Electronic resource] –http://vlsi.colorado.edu/~fabio/CUDD/

[6] Education: Virginia Tech: The Bradley Department of Electrical & Computer Engineering / College of Engineering; ISCAS89 Sequential Benchmark Circuits. – Access mode to an electronic resource. http://www.ece.vt.edu/mhsiao/iscas89.html is free.