ISBN: 978-1-6654-4502-3

Part Number: CFP21DTW-USB

# 2021 IEEE East-West Design & Test Symposium (EWDTS) Proceedings

**Batumi, Georgia, September 10 – 13, 2021**

# Optimizing Components of Finite State Machines Composition Based on Don’t Care Input Sequences in Hardware Implementation

Ekaterina Shirokova
Tomsk State University
Tomsk, Russia
k@shir.su

Larisa Evtushenko
Higher School of Economics
Moscow, Russia
evtlarisa@mail.ru

Andrey Laputenko
Tomsk State University
Tomsk, Russia
laputenko.av@gmail.com

***Abstract*—The paper is devoted to the problem of optimization of discrete multi-module systems. The proposed optimization is based on iterative component-wise optimization. This approach has established itself as a promising approach due to its good scalability. We consider a technique that is based on the use of the so-called "don't care" input sequences of the component, i.e. sequences that cannot appear at the input of a component under optimization in the composition and for this purpose, windows with sequential networks are selected. Preliminary experimental results illustrate that a proposed approach for the component-wise iterative optimization is effective for FSM networks when components are optimized for the further FPGA implementation.**

***Keywords—Finite State Machine (FSM), Synchronous composition of FSMs, FSM equation, FPGA***

## I. Introduction

The optimization of multi-module systems always attracted a lot of attention and one of promising approaches is the component-wise composition. In this paper, we assume that the behavior of each component is described by a finite state machine (FSM) and the interaction of the subsystems can be described using the operation of synchronous composition of FSMs [1] that corresponds to the instantaneous composition between component FSMs.

In the general case, to optimize an FSM component B in the composition, an FSM equation of the form $A \bullet X \cong A \bullet B$ [2] has to be solved where $A$ represents the combined behavior of all other composition parts. But the procedure of finding a general solution to such an equation from which an optimal B implementation can be extracted, has high complexity, which is exponential [1]. Also, there is a possibility of selection of the “window” and optimization of the component of interest in this window [2]. It is known that the complexity of FSM equation solving is lower for special topologies, in particular, for the case when the window contains the sequential composition and then the tail component of this sequential composition is under optimization.

In [3], an approach based on the selection of “windows” is also considered. However, in the [3], more complex “windows” are considered, where the composition is not necessarily sequential. Moreover, in [3], the optimization is discussed with respect to such criteria as communication lines, fault tolerance with respect to certain malfunctions, and others. The authors do not consider optimization with respect to the complexity of further Field Programmable Gate Array (FPGA) implementation. FPGA technology [4] is often used to implement modern algorithms, including the implementation of machine learning models. In [5], the optimization of networks of structural machines using internal “don’t care” sequences based on solving the satisfiability problem for Boolean formulas is proposed. The authors of [6], in addition to input “don’t care” sequences, consider output “don’t care” sequences but the complexity immediately becomes higher. In [7], the authors show that in some cases, when optimizing systems, it is more effective to solve a system of FSM equations, but not a set of them.

This work is a continuation of [8, 9]. Work [8] discusses the optimization of FPGA components of sequential composition with respect to the number of adaptive logic modules (ALMs). In our previous work [9], we considered various topologies of binary loop-free FSM composition and proposed ways of their optimization based on “don’t care” input sequences. Proposed approaches are based on constructing the network equivalent for the tail component of the loop-free composition of two FSMs. This paper describes an approach for optimizing binary composition with feedback using “don’t care” input sequences. The purpose of this work is to establish the effectiveness of the described approach for FSMs implementation in FPGA. This paper also presents experiments comparing the implementations of the tail component and its partial network equivalent on the FPGA in terms of the number of adaptive logic modules (ALM) and the maximum frequency of the circuit.

In Section II, we give basic notions and definitions. Section III contains the problem statement and description of the approach for constructing the network equivalent of the tail component for different topologies of a binary FSM composition. Section IV presents some preliminary experimental results.

## II. Preliminaries

### *A. Finite State Machines*

A *Finite State Machine* (FSM) is a 4-tuple $S = (S, I, O, T_S)$ [10], where $S$ is a nonempty finite set of states; $I$ is a non-empty finite set of input symbols (the input alphabet); $O$ is a non-empty finite set of output symbols (the output alphabet); $T_S \subseteq S \times I \times O \times S$ is a set of transitions. In the state $s \in S$ the FSM $S$ can transform the input symbol $i$ into the output symbol $o$ and go to the next state $s'$ if the 4-tuple $(s, i, o, s') \in T_S$, i.e. if the 4-tuple $(s, i, o, s')$ is a transition in the FSM $S$. An FSM $S$ with a highlighted initial state $s_0$ is called an *initial* FSM, that is, the initial FSM is the 5-tuple $S = (S, I, O, T_S, s_0)$. An FSM $S$ is called *complete* if for each pair $(i, s) \in I \times S$ there is at least one pair $(o, s') \in O \times S$ such that $(i, s, s', o) \in T_S$. Otherwise, the FSM $S$ is called *partial*. An FSM $S$ is called *deterministic* if for any pair $(i, s) \in I \times S$ there is at most one pair $(o, s') \in O \times S$ such that $(i, s, s', o) \in T_S$, otherwise the

FSM is called *non-deterministic*. The *language*, or the *behavior*, of the FSM $S$ in state $s$ (written, $L_S(s)$) [2], is the set of sequences of input-output pairs in the alphabet $I \times O$ obtained by successive transitions from state $s$.

## B. Relations between FSMs

The states $s_i$ and $s_j$ of complete deterministic FSMs $S_1$ and $S_2$ are *equivalent* if the state machine $S_1$ in the state $s_i$ and the state machine $S_2$ in state $s_j$ generate the same reaction under each stimulus [10, 12]. If reactions are different, the states $s_i$ and $s_j$ are *distinguishable*. If two initialized deterministic complete FSMs has equivalent initial states, then such FSMs are called *equivalent*. If any two different states of FSM are distinguishable, then FSM is called *reduced* (*minimal*). A (*state) reduced form* (*minimal form*) of $S$ is a reduced FSM equivalent to $S$.

The notion of equivalence can't be introduced for partial FSMs, because the behavior of FSM may not be defined for some input sequences. In this case, the concept of *quasi-equivalence* is introduced. Two states $a$ and $b$ of the FSMs $A$ and $B$, respectively, are quasi-equivalent if the responses of the FSMs in these states are the same under all inputs that are defined. Two initialized FSMs are quasi-equivalent if their initial states are quasi-equivalent. The initialized, possibly partial, reduced FSM $A$ is a *reduced form* of the partial FSM $B$ if the FSM $A$ is quasi-equivalent to the FSM $B$. There could be more than one reduced form for partial FSM, which are not isomorphic to each other.

It should be noted that there exist efficient techniques how to construct the minimal form of a deterministic complete FSM. However, there are still many works devoted to the problem of constructing a minimal form for a deterministic partial FSM (for example [10, 13]).

## C. Synchronous composition of FSMs

Usually multi-module complex systems (for example, digital circuits) are represented as a composition of interacting components and thus, the next step is to define the composition operator. In our work, we consider the synchronous composition of initialized complete deterministic FSMs [1], in which every input signal is processed in one clock cycle. Such composition is good for description of behavior of hardware modules.

The general topology of the composition of the FSMs $A$ and $B$ is presented in Figure 1.

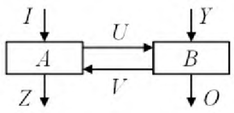

Fig. 1. The synchronous composition of the FSMs A and B

For the FSM $A$ an input alphabet is the set $I \times V$ and an output alphabet is the set $U \times Z$; for the FSM $B$ the *set* $Y \times U$ is the input alphabet and the set $V \times O$ is the output alphabet. The synchronous composition [3] of the FSMs $A$ and $B$ (or simply composition $A \bullet B$) has an input alphabet $I \times Y$ and an output alphabet $Z \times O$. An input/output symbol $(iyzo) \in I \times Y \times Z \times O$ belongs to the composition language if and only if there exists a matching pair of internal symbols $uv \in U \times V$ such that $(ivuz) \in L_A$ and $(yuvo) \in L_B$. In our work, we consider a special case of composition, namely, the sequential composition of FSMs $A$ and $B$ over alphabets $I, U, O$.

## III. Component wise optimization of synchronous composition of FSMs

For component-wise optimization of the synchronous composition of FSMs two approaches can be considered. In the first case, the components are optimized as "independent" machines independently of other components; in this case, the FSM component with minimal number of sates is usually derived, because, as shown by the experiments in [8], for the reduced form of the FSM, the number of ALMs of the corresponding FPGA implementation usually decreases. In the second case, the components are optimized depending on the behavior of other components. The authors of work [2] show that the behavior of an FSM network is preserved even if the FSM component changed to another nonequivalent FSM. Generally, in this case an explicit or implicit FSM equations' solving is involved.

## A. Optimization of the tail component of a sequential binary composition

In the Figure 2, the sequential composition of complete FSMs $A$ and $B$ is shown. For the tail component one can construct a partial FSM $B'$ whose behavior is determined only for the output sequences of the head component; moreover, for such sequences the behavior of the FSMs $B$ and $B'$ is the same.

The *network equivalent* [14] of the tail component FSM $B$ is the partial FSM $B'$ that is defined only over output sequences of the head component FSM, and $B$ is quasi-equivalent to $B'$.

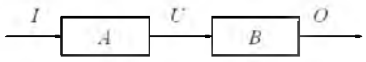

Fig. 2. The sequential composition of the FSMs $A$ and $B$

Given an FSM $S = (S, I_S, O_S, T_S, s_0)$, a "*reverse*" FSM for $S$ is an FSM $S^* = (S, O_S, I_S, T^*_S, s_0)$, $T^*_S \subseteq S \times O_S \times I_S \times S$, that is an FSM where the inputs and outputs are interchanged [14].

We use the following steps for constructing the network equivalent of the tail component [14]. At the first step we construct a "reverse" FSM for the head component $A$ and delete inputs at all transitions. We obtain an automaton $A^*$, which has the same alphabet as the output alphabet of the original FSM and the set of traces is the set of output sequences that can be produced by the FSM $A$.

At the next step, if necessary, determinize the resulting automaton $A^*$ and obtain $Det(A^*)$.

At the final step, we construct the intersection of the obtained deterministic automaton $Det(A^*)$ and the tail component $B$: construct all pairs of states of the automaton $Det(A^*)$ and tail component B, then determine the transitions between these pairs in the following way. If for $Det(A^*)$ there is a transition $(a, u, a')$ and for $B$ there is a transition $(b, u, o, b')$, then the transition $(ab, u, o, a'b')$ is defined in the intersection. The intersection states which cannot be reached from the initial state $(a_0, b_0)$ are excluded. The resulting partial FSM $B'$ is the network equivalent for the tail component $B$.

**Proposition 1** [14]. The tail component FSM $B$ is quasi-equivalent to the $B'$ constructed by the above steps. In addition, the behavior of the FSM $B'$ is defined on those and only on those sequences that are output sequences of the head component $A$.

According to the above proposition, the tail component $B$ can be changed by any reduced form of $B'$ without changing the external behavior of the sequential network.

Figure 3 shows an example of a sequential composition of two FSMs *A* and *B*.

At the first step, a reverse automaton *A** for the head component *A* is constructed (Figure 4a). Next, it is necessary to determinize the automaton *A**, i.e., to obtain *Det*(*A**). In order to obtain the network equivalent *B*' of the tail component *B*, we then intersect the obtained automaton *Det*(*A**) and the FSM *B*. FSM *B*' is partial and a reduced form of *B*' is shown in Figure 4b.

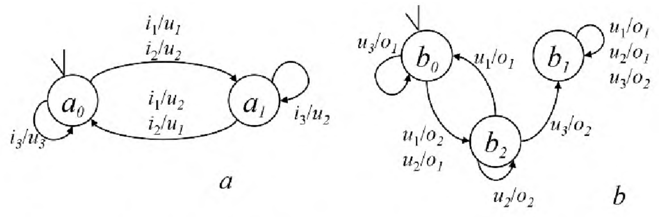

Fig. 3. (a) The FSM *A*, (b) The FSM *B*

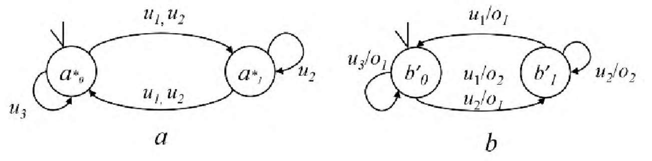

Fig. 4. (a) The reverse automaton *A**, (b) The network equivalent *B*'

Comparing the obtained network equivalent *B*' with the original FSM *B*, one can see that the FSM *B*' has a smaller number of transitions and states than the FSM *B*. Those undefined transitions are in fact, don't care transitions and can be augmented in any way in *B* hardware implementation.

## B. *Optimization of the tail component of a binary loop-free composition*

In [9], we considered two cases of a binary loop-free composition, when the tail component has additional inputs. When constructing the network equivalent of the tail component, these inputs must be taken into account.

In the first case the FSM *A* has a set of inputs *I* and set of outputs *U*, while the FSM *B* has a set of inputs $I \times U$ and a set of outputs *O* (Figure 5a). The reverse automaton *A** for the FSM *A* has the alphabet $I \times U$ (it is necessary to escape the character " / " at each transition). To obtain the network equivalent *B*' of the tail component *B*, it is necessary to construct the intersection of the automaton *A** and the FSM *B* and minimize the resulting FSM.

In the second case, we added the input *V* to the FSM *B* (Figure 5b). The reverse automaton *A** for the FSM *A* is constructed in the same way as in the first case. To obtain the network equivalent *B*' of the tail component *B*, it is necessary to construct the intersection of the automaton *A** and the FSM *B*. Next, it is necessary to minimize the resulting FSM and extend each of its transitions to the alphabet *V*.

## C. *Optimization of the tail component of a binary composition with feedback*

A binary composition with feedback is shown in Figure 6a. Let the FSM *A* (Figure 6b) have a set of inputs $I \times O$ and a set of outputs *U*. The FSM *B* (Figure 6c) has a set of inputs *U* and a set of outputs *O*.

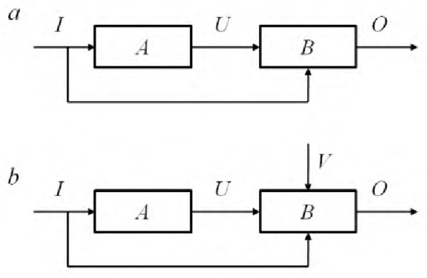

Fig. 5. The synchronous composition of *A* and *B*: (a) the first case, (b) the second case

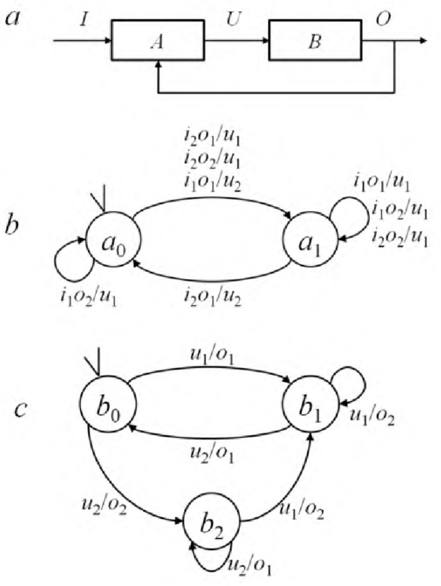

Fig. 6. (a) The synchronous composition with feedback of the FSMs *A* and *B*, (b) The FSM *A*, (c) The FSM *B*

To build a reverse automaton *A** for the FSM *A*, one has to follow several steps. Firstly, at all transitions of the FSM *A*, it is necessary to remove symbols from the alphabet *I*. Further, at each transition, delete the symbol " / " and swap the symbol from the alphabet *O* and the symbol from the alphabet *U*. Thus, the resulting automaton *A** has the alphabet $U \times O$ (Figure 7a). Next, we determinize the automaton *A** and obtain an automaton *Det*(*A**).

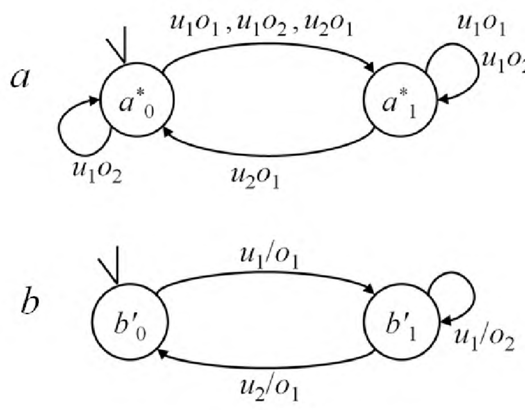

Fig. 7. (a) The reverse automaton *A**, (b) The network equivalent *B*'

In order to obtain the network equivalent *B*' (Figure 7b) of the component *B*, we intersect the automaton *Det*(*A**) and the FSM *B*.

### D. Optimization of components of multi-module synchronous composition based on iterative selection of "windows"

As it is known, the behavior of a FSM network is not necessary changed if the FSM component is replaced by a FSM that is not equivalent to it; accordingly, components can be optimized depending on the behavior of other (neighbor) components, and explicit or implicit FSM equations' solving is usually involved for such optimization. The general solution of the FSM equation of the form $A \bullet X \cong A \bullet B$ [2], where the FSM $A$ describes the joint behavior of all components except $B$, can be considered as a reservoir for choosing the optimal solution (with respect to a given optimization criterion).

In compositions with several components, we can choose a "window" with two FSMs components $A$ and $B$ (Figure 8). Reactions of component $B$ are essential for the overall composition only on a subset of input sequences that may come from component $A$.

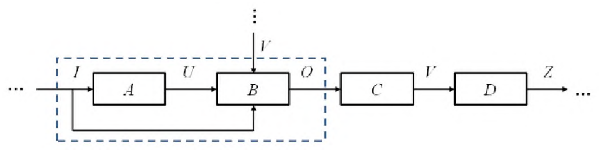

Fig. 8. The network of three FSMs

Thus, in multi-module systems it seems helpful to select iteratively the sequence of "windows", i.e. select the "window", optimize the tail component, and then select another "window".

## IV. EXPERIMENTAL RESULTS

In this paper, we evaluated hardware FSM implementations on FPGA. The number of adaptive logic modules and the maximum operating frequency of the circuit were chosen as the estimated parameters. The values of the selected parameters were taken from the compilation report in Quartus II software [15]. A Cyclone V device with a 5CGXFC5C6F27C7 FPGA chip was chosen as the target device. One ALM of the Cyclone V platform has an 8-input truth table (Lookup Table), 2 full adders and 4 registers.

At the first stage, random FSMs were generated [16] for the head component with the number of states from 4 to 6 and the number of inputs from 4 to 6 and the number of outputs from 2 to 16. The FSMs of the tail components had the number of states from 4 to 8 and the number of inputs from 2 to 16.

According to the described approach, a partial network equivalent of the tail component of the sequential composition was constructed. The SIS [17] tool was utilized to minimize the partial equivalent, represented in KISS format. After the state encoding procedure made by SIS, an FSM was converted to BLIF [18] format. Using the ABC tool [19], a BLIF file was converted to a description in the Verilog language. Further, the system was synthesized using Quartus II. From the Quartus II report we can see the parameters of the synthesized circuit: the number of ALMs and the maximum frequency of the circuit. During experiments 90 sequential compositions were generated. According to the results, the largest reduction in the number of ALMs was 66,7 % for the tail component with 5 states and 2 input symbols. The maximum increase in the frequency of the circuit was 42,6 % for the tail component with 6 states and 4 input symbols. A decrease in the number of ALMs occurs in 30% of cases, an increase of frequency in 55% of cases. About 30 FSM compositions (topology as in Figure 5a) were generated, tail FSM has the number of inputs from 6 to 32, both head and tail components has from 5 to 6 states. The number of ALMs was reduced by a maximum of 72,3%. The maximum increase in frequency was 71,7%. A decrease in the number of ALMs occurred in 55,6% of cases, and an increase in frequency in 61,1% of cases.

## V. CONCLUSION

In this paper we have considered the problem of optimization of discrete multi-module systems, namely an approach for optimizing binary composition with feedback.

The results of the performed experiments demonstrate the effectiveness of the proposed approach for optimizing the further FPGA implementation of the FSM composition.

As a part of our further research, we plan to study the influence of the FSM components' properties on the complexity of their hardware implementations; to consider various types of compositions and the effect of the composition topology on the proposed method for the component-wise optimization.

## REFERENCES

1. T. Villa, N. Yevtushenko, R.K. Brayton, A. Mishchenko, A. Petrenko, A.L. Sangiovanni-Vincentelli, "The Unknown Component Problem: Theory and Applications," Springer, p. 323, 2011.
2. N.V. Yevtushenko, M.V. Rekun and S.V. Tihomirova, "Non-deterministic FSMs: analysis and synthesis. Part 2. Solving FSM Equations," Tomsk, Tomsk State University, p. 111, 2009. (In Russian).
3. S.V. Tikhomirova, "Optimization of multicomponent discrete systems based on the solution of FSM equations," The dissertation for the degree of candidate of technical sciences, Tomsk, Tomsk State University, 2008. (In Russian).
4. Zhao, Ritchie, et al. "Accelerating binarized convolutional neural networks with software-programmable FPGAs." Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, 2017.
5. A. Mishchenko, R.K. Brayton, "SAT-based complete don't-care computation for network optimization," The Proceedings of the Design, Automation and Test in Europe Conference, Volume 01, pp. 412-417, March 2005.
6. J.-K. Rho, F. Somenzi, Don't care sequences and the optimization of interacting finite state machines," IEEE Trans. Comp. Aided Des., Volume 13, No. 7, pp. 865-874, 1994.
7. N. Yevtushenko, S. Zharikova, M. Vetrova, "Multi component digital circuit optimization by solving FSM equations," Proceedings of the Euromicro Symposium on Digital Systems Design, DSD '03, pp. 62–68, 2003.
8. V.A. Schwarzkop, "Optimization of components of multimodularsystems based on the solution of automaton equations," Master's thesis, Tomsk, Tomsk State University, 2019. (In Russian).
9. E. Shirokova, L. Evtushenko, A. Laputenko and N. Yevtushenko, "Optimizing Components of Multi–Module Systems Based on don't Care Input Sequences," IEEE East–West Design & Test Symposium (EWDTS), Varna, Bulgaria, pp. 1–5, 2020.
10. G.P. Agibalov, A.M. Oranov, "Lectures on the theory of FSMs," Tomsk, Tomsk Publishing House. Univ., p. 186, 1984. (In Russian).
11. N.V. Yevtushenko, A.F. Petrenko, M.V. Vetrova, "Non-deterministic FSMs: analysis and synthesis. Part 1. Relations and operations: Textbook," Tomsk: Tomsk State University, p. 142, 2006. (In Russian).
12. A. Gill, "Introduction to the theory of finite-state machines," Moscow, Nauka, p. 272, 1966.
13. A.D. Zakrevsky, Yu.V. Pottosin, L.D. Cheremisinova, "Logical principles of designing discrete devices," Moscow, Fizmatlit, p. 592, 2007. (In Russian).
14. M.V. Vetrova, "Development of synthesis and testing algorithms for finite-automaton compensators," The dissertation for the degree of

candidate of technical sciences, Tomsk, Tomsk State University, 2003. (In Russian).

15. Quartus II Handbook website [Electronic resource], URL: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/qts/quartusii_handbook.pdf
16. Test generation for Finite State Machine. URL: http://fsmtestonline.ru/
17. Sentovich, Ellen M., et al. "SIS: A system for sequential circuit synthesis.", 1992
18. Berkeley logic interchange format (BLIF) [Electronic resource], URL:https://www.cse.iitb.ac.in/~supratik/courses/cs226/spr16/blif.pdf
19. Berkeley University website [Electronic resource] “ABC: A System for Sequential Synthesis and Verification,” URL: https://people.eecs.berkeley.edu/~alanmi/abc