MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE

KHARKOV NATIONAL UNIVERSITY OF RADIOELECTRONICS

ISBN 966-659-088-3

Proceedings of East-West Design & Test Workshop (EWDTW'04)

Yalta, Alushta, Crimea, Ukraine, September 23 – 26, 2004

CONTENTS

USE OF PARALLELISM IN FINITE STATE MACHINES. MATHEMATICAL LEVEL

O. NEMCHENKO, G. KRIVOULYA

Kharkiv National University of Radio Electronics

tel.: +38 (057) 70-21-326, e-mail: avn@nemchenko.net

Abstract. The method of the description of parallel systems using finite state machines is examined. The mathematical model proposed in this article can use both for the description and for synthesis of synchronous and asynchronous parallel systems with usage of abstractions of parallel programming: threads, processes, flags, mutexes, semaphores and complex synchronizations.

Keywords: Parallelism, FSM, thread, process, asynchronous automation

Introduction

One of methods of computing systems productivity rising is usage of parallelism in operation. Exists two ways of parallel data processing: strictly parallelism and command pipe [1]. The command pipe is present at personal computers, parallel processing is present, as a rule, in specialized computing systems - on parallel computers.

Parallel computer architectures strongly differ one from each other. In There is a problem of code portability - the program fulfilled effectively on one system, practically does not use resources on another. In other words, there is weak program portability and the compiled code for parallel computers.

Parallelism can be created on the basis of several computing units. Such system becomes complicated over availability of such units, commutative equipment and interconnection interfaces. It is much better to create the parallel arrangement in one chip with the required architecture. In this case the programmable logic CPLD, FPGA, can be used.

In this article the mathematical model of a parallel finite state machine (FSM) digital automaton is discussed which can be used for the description of parallel algorithms and for its synthesis.

Mathematical level of the description of parallel FSM

For a parallel FSM at the same time several states can be active. Therefore, in a context parallel FSM it is necessary to consider *the set* of active states.

The sequential FSM is described by the formula

$$S_| = \{A, X, Y, \delta, \lambda, a_1\}$$

where $A = \{a_1, a_2, ..., a_n\}$ is the set of states, $X = \{x_1, x_2, ..., x_m\}$ – set of input signals, $Y = \{y_1, y_2, ..., y_k\}$ – set of output signals, $\delta(a(t-1), X(t))$ – the function of transitions, $\lambda(a(t-1), X(t))$ – the function of outputs. The parallel FSM may be described as follows:

$$S_{||} = \begin{cases} A = \{a_1, a_2, ..., a_n\}; \\ X = \{x_1, x_2, ..., x_k\}; \\ Y = \{y_1, y_2, ..., y_1\}; \\ A(0) \subseteq A; \\ A(t) = \delta(A(t-1), X(t)); \\ Y(t) = \lambda(A(t-1), X(t)); \end{cases} \quad (1)$$

First three equations are similar to the sequential FSM.

The fourth formula defines set of active states for an initial instant. This set can consist one or several states.

The fifth formula determines the function of transitions which receives set of active states and inputs. The answer will be set of active states for the following instant.

The sixth formula determines outputs. It depends on set of active states and inputs.

The Mile and Moore FSM differs by a rule of definition of output set. It also is applicable for parallel FSM: $Y(t) = \lambda(A(t-1), X(t))$ for a parallel Mile FSM and $Y(t) = \lambda(A(t))$ for Moore FSM.

In (1) system the set of active states in initial instant A (0) can coincide with set of all states A. For sequential FSM coincidence of these sets means that the FSM stays in the same state. For parallel FSM incidence of these sets means, that in an initial instant all states are active simultaneously. But in the following moment some from them can cease to be active.

It is enough (1) system for the description of operation of synchronous parallel FSM. Threads, processes, flags, synchronization, mutexes and semaphores [3] can be applied to the description of parallel FSM.

The thread is a set of sequential operations. There is no parallelism inside one thread. Actually, it is sequential FSM. The process in operating systems is set of parallel streams which are closely interconnected. The flag is some logical condition, a Boolean variable. This variable can be set in one place, and to be used in another. Mutex and semaphore is more complex variant of a flag. In fact, the mutex or semaphore is the flag that is used in several threads and processes.

System (1) allows to realize all these concepts. But the mathematical description will be very bulky. It is possible to use flow-chart for the description parallel FSM or parallel algorithms. Usage of parallel flow-char considerably simplifies the description of complex algorithms and allows to describe algorithms for real systems. So, the description with using a set of equations represents a mathematical level, and usage of parallel flow-chart with processes, streams, flags and others is called *a logic level*. In this article the mathematical level is discussed.

Synchronous and asynchronous parallel FSM

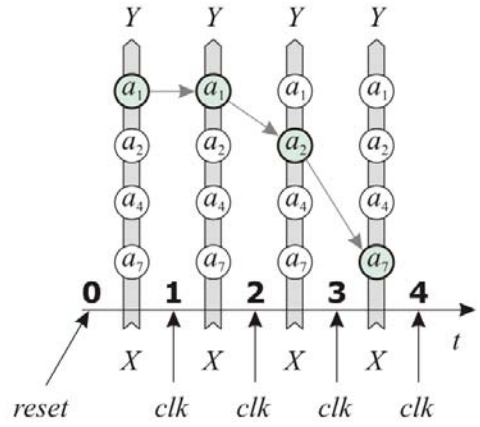
Fig. 1a shows the operation of sequential FSM. It is clear, that the time mode of sequential FSM operation is determined by two input signals - a signal of reset and synchropulse clk. By a reset signal the FSM passes in an initial state. Then on clock ticks of synchropulse it changes the state under effect of input signals and forms output signals.

Parallel FSM works similarly but the difference is that in some given instant can be active several states at the same time.

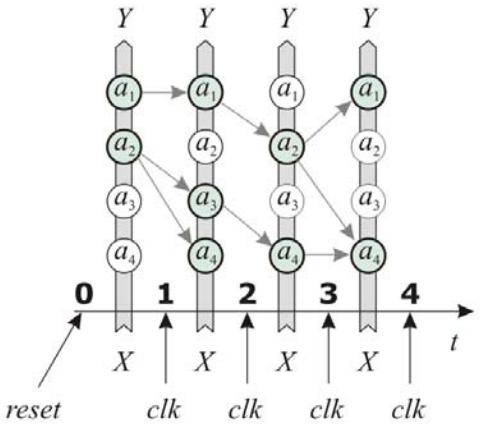
As shown on fig. 1b, state can not generate new state, generate only one or several new states. Also several states can generate one.

Fig. 1b shows that states changes simultaneously ("simultaneity" is abstraction of automatic time). But it is possible the time mode shown on fig. 2.

The Fig. 1b shows the synchronous parallel FSM, fig. 2 shows asynchronous parallel FSM. The concept of "synchronism" for parallel FSM differs from the same concept for sequential FSM [2]. As it is shown on fig. 2 asynchronous parallel FSM decomposes on several separate parallel FSMs width different time modes. They have common inputs X and common outputs Y. Also they can influence each other (for example, state a1 has generated state a4).



a



b

Fig. 1. Synchronous FSM: a – sequential, b – parallel

System (1) determines synchronous parallel FSM, however it cannot determine asynchronous parallel FSM. It is obvious, that it is necessary to extend it. For this purpose it is necessary to consider some concepts.

The thread is one sequence of states inside which there is no parallelism. Branching and loops are possible inside. Any sequential FSM is thread. Therefore, definition of sequential FSM determines one thread, and fig. 1à illustrates its operation mode.

The process is a set of parallel threads that have common time mode. Actually, system (1) determines one process, and fig. 1á illustrates it.

The parallel FSM is a set of processes that may have common (synchronous parallel FSM) or different (asynchronous parallel FSM) time mode. The example of such FSM is shown on fig. 2.

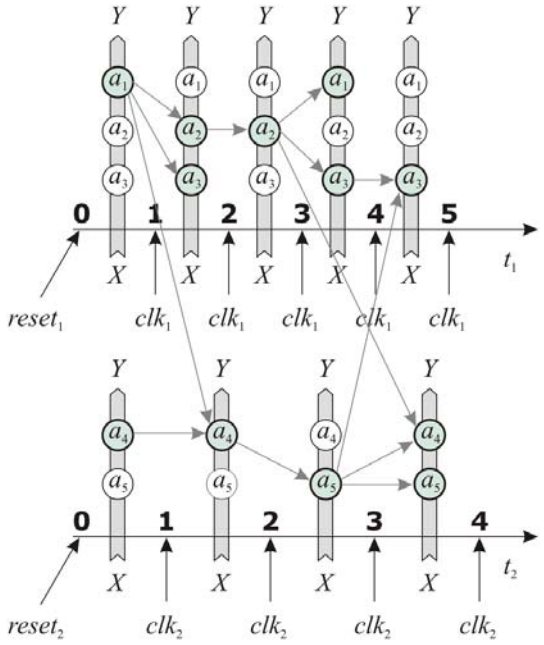


Fig. 2. Asynchronous parallel FSM

The synchronizer is a rule of operation for several processes – the beginning of operation, the end and its suspend.

The flag is a condition of transitions. The flag is determined by a set of active states.

Let determine synchronous and asynchronous mathematical model for parallel FSM using new concepts.

The model of thread coincides with a model of sequential FSM:

$$\mathrm{Th} = \mathrm{S}_{|} = \begin{cases} \mathrm{A} = \{a_1, a_2, ..., a_n\}; \\ \mathrm{X} = \{x_1, x_2, ..., x_k\}; \\ \mathrm{Y} = \{y_1, y_2, ..., y_1\}; \\ a_0 \in \mathrm{A}; \\ a(t) = \delta(a(t-1), X(t)); \\ Y(t) = \lambda(a(t-1), X(t)); \end{cases} \quad (2)$$

The thread is determined by set of states, inputs and outputs; *one* initial state, the function of transitions having as arguments *one* current state and inputs returning *one* following state; the function of outputs with *one* current state and inputs.

The process comprises one or several threads with the same time mode. Therefore, the time mode should be determined in a mathematical model:

$$\mathrm{P} = \begin{cases} \mathrm{A} = \{a_1, a_2, ..., a_n\}; \\ \mathrm{X} = \{x_1, x_2, ..., x_k\}; \\ \mathrm{Y} = \{y_1, y_2, ..., y_1\}; \\ \mathrm{A}(0) \subseteq \mathrm{A}; \\ \mathrm{A}(t) = \delta(\mathrm{A}(t-1), \mathrm{X}(t)); \\ \mathrm{Y}(t) = \lambda(\mathrm{A}(t-1), \mathrm{X}(t)); \\ \underset{\mathrm{CLK}=0\to1}{} \\ t \rightarrow t+1; \\ \underset{\mathrm{RESET}=1}{} \\ t \rightarrow 0; \end{cases} \quad (3)$$

In system (3) the seventh string shows, that after rising edge of synchropulse *CLK* the time of the given process increases. The eighth string illustrates the fact, that at setting of a reset signal *RESET* the given process passes in the initial state. Both these strings determine the function of automatic time *Ft* (*CLK*, *RESET*).

Inputs are the set of input signals *X*, synchropulse *CLK* and a reset signal *RESET*.

In system (3) threads are determined implicitly through the set of states and functions of transitions. Rules of their beginning, stopping and suspending – synchronizers - are defined by the function of transitions. However threads can be specified explicitly:

$$\mathrm{P} = \begin{cases} \mathrm{Th} = \{\mathrm{Th}_1, \mathrm{Th}_2, ..., \mathrm{Th}_m\}; \\ \mathrm{S} = \{\sigma_1, \sigma_2, ..., \sigma_p\}; \\ \underset{\mathrm{CLK}=0\to1}{} \\ t \rightarrow t+1; \\ \underset{\mathrm{RESET}=1}{} \\ t \rightarrow 0; \end{cases} \quad (4)$$

In system (4) synchronizers are determined by the set *S*. Synchronizers from this set are used in functions of transitions in threads.

Flags can be used at interaction of threads and processes. A flag is a Boolean function arguments of which are facts of activity of some states. Flags can be used for clocking various streams.

Parallel FSM can include one process or more. Some input / output signals and flags can be used for inerprocess communication.

The following system is enough for the description of parallel FSM:

$$S_{||} = \begin{cases} X = \{x_1, x_2, ..., x_k\}; \\ Y = \{y_1, y_2, ..., y_1\}; \\ P = \{P_1, P_2, ..., P_p\}; \end{cases} \quad (5)$$

Inputs and outputs and the set of processes are described in system (5). Each process is described by system (3) or by systems (4) and (2).

So, parallel FSM is described by the following three:

$$S_{||} = \{X, Y, P\}, \quad (6)$$

X is input set, *Y* is output set, *P* is a set of processes. Each process is set by seven units:

$$P = \{A, X, Y, A_0, \delta, \lambda, Ft\}, \quad (7)$$

A is a set of state, *X* is input set, *Y* is output set, A_0 is a set of initial states, $\delta(A(t-1), X(t))$ is the function of transitions, $\lambda(A(t-1), X(t))$ is the function of outputs, *Ft* (*CLK*, *RESET*) is the function of automatic time.

The process also can be set by three units:

$$P = \{Th, S, Ft\}, \quad (8)$$

$S(A(t-1), X(t))$ is a set of Boolean functions of synchronizers, *Th* is a set of threads specified as follows:

$$Th = \{A, X, Y, a_0, \delta, \lambda\}. \quad (9)$$

For the Moore parallel FSM the outputs function depends on only state.

Not-deterministic automation

At the definition of sequential FSM it is necessary to consider the condition of an uniqueness: from the same state can exist no more than one transition with the same entry conditions. Violation of this condition generates not-deterministic automation (NDA).

It is possible to construct the parallel system on NDA basis. The article [4] describes the mathematical apparatus connected with formal representation of parallel operating algorithms as a model of NDA on the base of recurrent canonical equations system that describes all private events realized in the control system. Thus the term "non-deterministic" suggests as about something random, however there is nothing random in NDA [4]. The language of the description and synthesis of parallel algorithms is created.

Parallel FSM is other approach to the description and synthesis of parallel algorithms. However it can be also considered as a special case of NDA. From the mathematical point of view parallel FSM can be considered as a set of several sequential FSM for which the current state is used as an input / output signal. So, the process of synthesis for parallel FSM is decomposed to synthesis of several sequential interconnected FSM. It allows to use existing methods of states minimisation, coding and synthesis algorithms.

Conclusion

In the article the mathematical model of a parallel FSM has been considered. The model allows to describe and then to synthesise parallel algorithm. Conceptions used at parallel programming were used for the description and synthesis of parallel algorithms.

Parallel FSM can be applied for solution of various tasks. A good example of synchronous parallel FSM usage is control system where it is necessary to carry out a lot of mathematical calculations. Asynchronous parallel FSM can be applied in the cases when it is necessary to realize the system where some units work in different time modes. In particular, there are various systems where the data is read from several sensors. The sensors can use different protocols for operation and the different time mode.

References: 1. *Voevodin V., Voevodin. Vl.* Parallel calculations // BHV Petersburg, 2002. **2.** *Glushkov V.* Digital automation synthesis // Physmathgis, 1962. **3.** *Williams Al.* Windows 2000 Systems Programming. Black Book", The Coriolis Group, 2000. **4.** *Vashkevich N., Vashkevich S.* Nod-deterministic automation and their usage for control systems synthesis. Part 1. Equivalent conversions of non-deterministic automation // Penza: Publishing house, 1996.

ELECTRICAL TEST IS NOT ENOUGH FOR QUALITY

BENGT MAGNHAGEN

JONKOPING UNIVERSITY, SWEDEN

bengt.magnhagen@ing.hj.se

Electrical test means Functional Test (FT), In Circuit Test (ICT) or Boundary Scan Test (BST) or even a combination of these technologies. However, with modern technology, like SMD (Surface Mounted Devices) technology, BGA (Ball Grid Array) components and extremely small component dimensions, electrical test alone does not meet the quality requierments.

Electrical test can not identify bad soldering and bad alignment of components, as examples. Missing decoupling capacitors and so on can not be detected because of it is hard to get physical access for testprobes. Do not forget that digital designs contains a lot of analogue devices!

The tutorial will discuss today test technology with equipment for ICT and BST as well as its pros and cons. And as the addition of this, Inspection. Inspection has traditonally been performed manually but this is not realistic today with board crowded by components. Today Inspection is performed by machine vision. Optical technique named Automated Optical Inspection (AOI) and more advanced X-ray inspection (AXI). AOI and AXI is not the future, it is here today.

EMC /EMI is also a growing challenge and some new ideas will be discussed how to test for these phenomenas.

Підписано до дру*у 9.09.т004. Формат 60*84$^{1}/_{8}$.

Умов. дру*. ар*. ууи4. Облі*.-вид. ар*. у0и0. Зам. № т-875. Тираж №0 прим.

Віддру*овано в навчальнофнау*овому видавничофполіграфічному центрі ХНУРЕ.

6№№6иХар*івипросп. ЛенінаиМ4.