PRINCIPLE SCHEME OF APERIODIC FINITE STATE MACHINE

Dimitar S. Tyanev, Stamen I. Kolev

Abstract: In the present work, an original logic structure of control unit based on finite state machine, with hardwired rules (FSMHRCU), working with floating cycles duration is presented. The structure may be used as basis for the implementation of either Mealy or Moore finite state machine. The operation of the synthesized structure is clarified schematically and graphically. The possibility of the execution of nano- and pico-programs in the terms of the current micro-command from the sequence of micro-commands of the controlling algorithm, is presented. The structure supplies aperiodicity in these low levels and it is synthesized and presented below in the paper.

1. Introduction

The possible hardware structures for control unit based on finite state machines, with hardwired rules (FSMHRCU) are discussed in [1-19 and others.]. The methodologies for the design of synchronous FSMHRCUs are described in most of the referenced sources, while methodologies for the design of such asynchronous structures are rarely met. An example for the design and synthesis of entirely asynchronous FSMHRCU is shown in [3]. A classification and detailed description of the logical structures of finite state machines (FSM) are present in [2].

In the present article we discuss the task for the design of FSMHRCU in the terms of the two control approaches – synchronous and asynchronous. Our main goal is to achieve the maximum possible performance of the control execution algorithm combined with the simplicity of the technical implementation of the synchronous control. Such kind of finite state machines can be defined as state machines with floating clock rate duration or as aperiodic finite state machines and there principle schemes are not met in the sources

.

2. Logical structure of aperiodic finite state machine

In contrast with the finite state machines with microcoded rules, which may have the number of cycles needed for the completion of the micro-operation coded within the micro-operation code, the FSMHRCUs do not have this advantage. This is due to the fact that FSMHRCUs have storage only for their states. This means that the number of the cycles needed for one micro-operation, which number is function of the current state of the machine is not stored. From this follows, that the number of the cycles must be derived from the state code combination for each micro-operation.

Counter is used to determine the end of the micro-operation. The duration of each micro-operation is determined by the designer at design time and it is measured in number of clock cycles. For that purpose a code table (table1) for the duration encoding must be created.

Table 1. Example coding table

State S(t)	Duration T(t+1)
S0	1 cycle (001)
S1	3 cycles (011)
S2	4 cycles (100)
S3	2 cycles (010)
... ...	... ...

The state machine changes its state only if the current state has expired (in sense of cycle), which corresponds to the following state diagram.

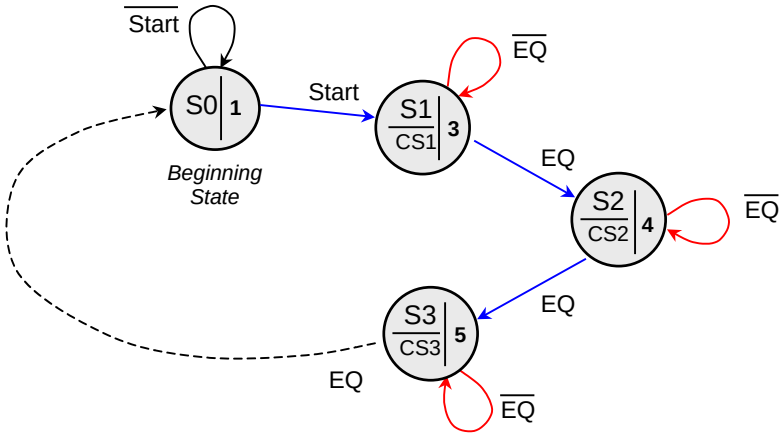

Fig. 1. State diagram for finite state machine with floating cycles duration

In the state diagram we can see that when the signal "*Start*" is missing, the state machine keeps its initial state for undefined number of cycles. The state transition begins after the appearance of the "*Start*" signal and it continues from one state to another only if the time of the current state is up, plus of course we take on count the additional jump conditions related to the concrete control algorithm. Now it can be easily concluded that the signal that writes the state code combination in the state register is function of the cycle counter value.

Along with the computation of the code combination of the new state S(t+1), the state transition circuit must also encode the duration of the next micro-operation T(t+1) as a function of the new state T(t+1)=f(S(t)) as is in table1 The newly computed duration must be stored in the counter as initial value. This storage procedure must be accomplished along with the storage of the new state code combination. Thus the counter can be considered as natural extension of the code condition register.

Regarding to the reasoning above, an additional circuit for the FSMHRCU's structure is designed to enable the FSMHRCU to work with floating cycle duration. As it can be seen in figure 2, in its lower part is depicted a finite state machine which corresponds to Moore state machine .In the center of the figure it can be seen the code condition register RG_S and the circuit realizing the output function Out_L is connected to the RG_S register. The new code condition combination S(t+1) is passed to the input of RG_S, which is computed in the current cycle by JF – that is the circuit realizing the state transition functions. The presented structure is designed as synchronous and is driven by the signal *Clock*. The main role of the additional circuitry is to select the correct clock pulses W and drive them to the write inputs of the counter and the code condition register.

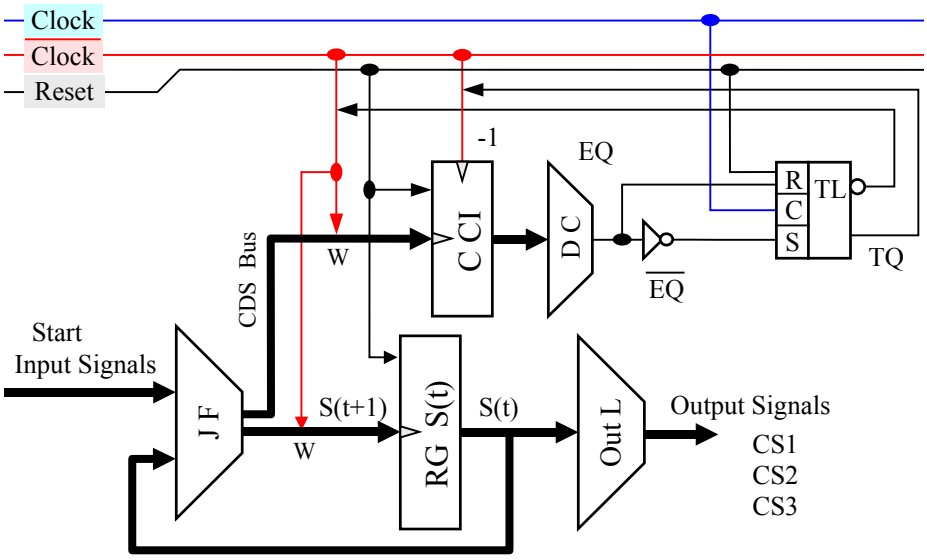

Fig. 2. Logical structure of automaton with floating cycles duration

The additional circuitry consists of down counter, which counts the clock pulses C_CI and decoder DC is connected to its outputs. The decoder recognizes zero state EQ (EQ=1, if (C_CI)=0). EQ and EQ bar drive the RS trigger *TLatch* state, which through its output TQ controls in which moment the inversed clock signal $\overline{Clock}$ should be used to drive the counter(-1) or should be passed as write signal W to RG_S and C_CI. The operation of the structure is depicted with the time diagram in figure 3. The write operations in the register RG_S and the counter are done according to the rising edge of the write signal W.

A sequence corresponding to the state diagram from figure 1 is shown. State S1 is hold for three clock cycles, state S2 is hold for 4 and state S3 is hold for 5 clock cycles. The initial state S0 with duration of one cycle is repeated several times in loop while the signal "*Start*" appears. To start the state transitions correctly, there are certain requirements about the signal "*Start*":

- The signal "*Start*" must appear synchronously with the rising edge of the *Clock* signal;
- Its duration must be minimum one period as shown in the time diagram.

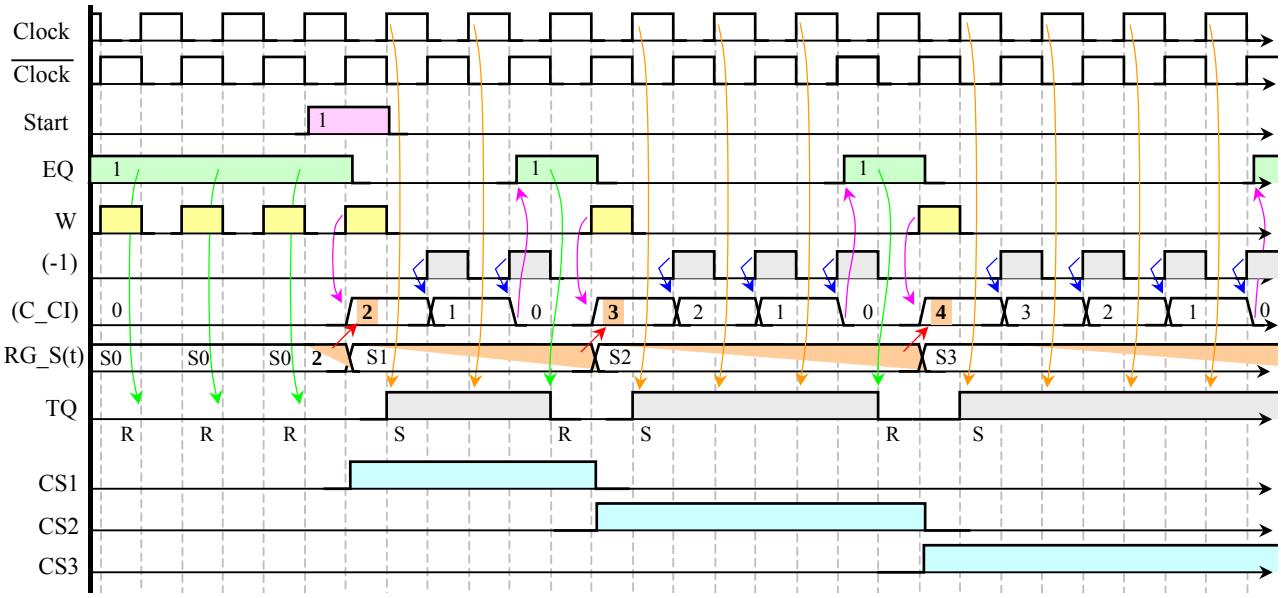

Fig. 3. Time diagram for the execution of the sequence (…S0$^{(1)}$), S1$^{(3)}$, S2$^{(4)}$, S3$^{(5)}$ …

According to the said, the circuit at the input JF should be able to compute the duration of state S1 (for the example it is 2) for the time between the rising and the falling edge of the *Clock* signal. For all of the rest states the time for the computations of the same type will be much more, which should be taken on count in the design process of the encoding circuit according to table 1.

The control signals CS1, CS2 and CS3, shown in the time diagram have different duration, accordingly to the different machine states accepted for the example.

The logic of the signals in the time diagram, implemented in circuits of the FSM is as follows:

- Write signal W :

$$W = \overline{Clock} \cap \overline{TQ} \,. \tag{1}$$

- Signal (-1) :

$$(-1) = \overline{Clock} \cap TQ \,. \tag{2}$$

- Control signal for the trigger TL :

$$\begin{aligned} R &= Reset \cup (Clock \cap EQ) \;\; ; \\ S &= Clock \cap \overline{EQ} \;\; . \end{aligned} \tag{3}$$

The designed structure, without any changes, could be combined with Mealy state machine, which defines it as universal.

3. Nano-commands. Pico-commands

The structure presented on figure 2 reveals the opportunity of adding and using nano- and pico-commands, which also can be with floating cycle duration, in the period of one micro command. For the purpose, the clock generator must be able to generate signals with different duration, for example frequency multiplier can be used. In this case all of the clock sequences will be multiple of the base one and absolutely synchronous with it. As a result the durations of the different nano- and pico-cycles will be always multiple of the period of the corresponding micro-command cycle sequence and they could be implemented as nano- and pico-programs in the time frame of the micro-command, they could be even implemented as floating cycle duration ones. The described above is illustrated with the time diagram in figure 4, where with ncij are shown three example nano control signals, which could realize one possible nano-program, corresponding to the control signal CS1, with clock sequence from the signal 2.*Clock*.

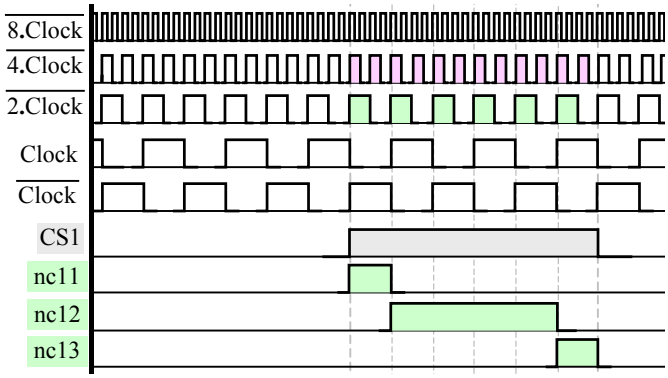

Fig. 4. Three cycle nano-program

Another opportunity is described in figure 5 where 4.*Clock* sequence is used. With pcij are shown eight example pico-commands, which represent eight cycle execution of the pico-program, corresponding to the same control signal CS1.

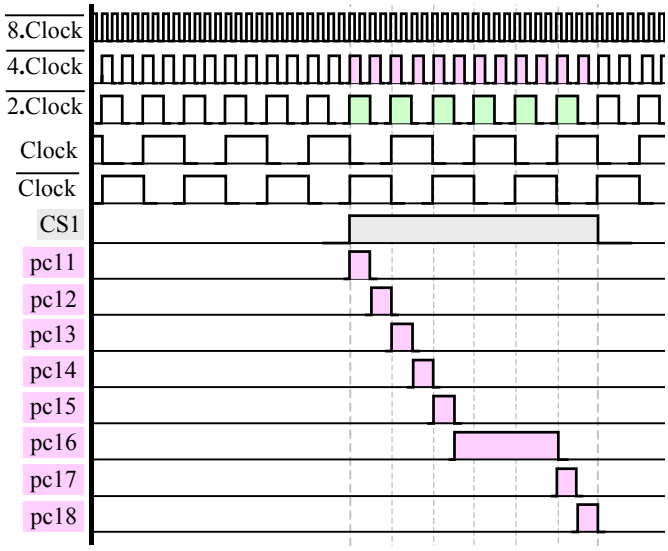

Fig. 5. Eight cycle nano-program

The circuit implementation of the shown on figure four and five example nano- and pico-programs is depicted in figure 6. The circuit from figure 6 should be considered as an extension to the one on figure 2. Also it should be made clear that the pico-comands are related with another control signal marked as CS7. The decomposition of one control signal to more than one sub levels is possible but technically it seems to be pointless.

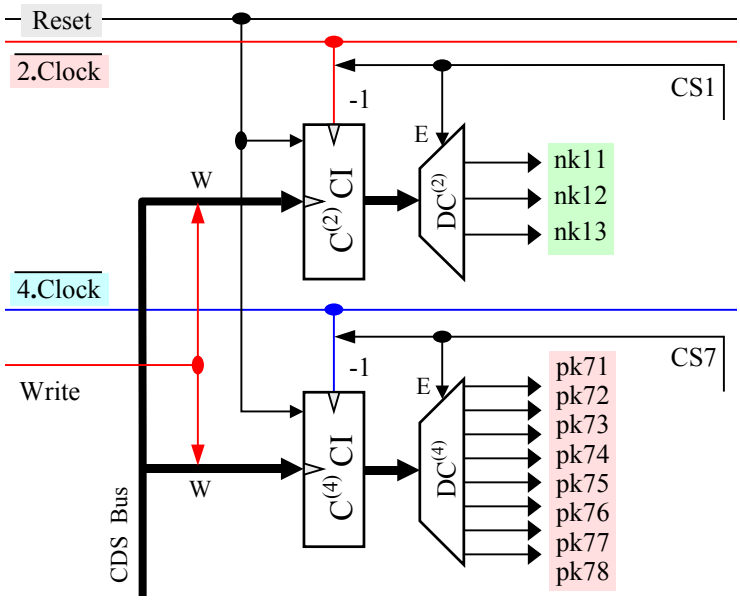

Fig. 6. Circuit realization of nano and pico-control signals

Each basic control signal CSi , with lower level programs, operates with its own counter for the related cycle sequence, in similar fashion as it is shown in figure 6, where the counters $C^{(2)}$CI and $C^{(4)}$CI operate with sequences 2.*Clock* and 4.*Clock* respectively. The initial values of these counters are set by the signal W (Write), which logic is shown in (1). The initial value in the counters is the value in C_CI from the basic structure multiplied with the related clock sequence multiple. Thus, for example, if CS1 has duration of three clock cycles of the signal *Clock*, in $C^{(2)}$_CI the value must be doubled and in $C^{(4)}$_CI must four times bigger. This is achieved easily with left shifted writing of the value in the corresponding counters on signal W.

The final value of the nano and pico control signals ncij, and pcij, is formed by the decoders $DC^{(2)}$ or $DC^{(4)}$ from figure 6. The nano-commands as well as the pico-commands, which have duration more than one period, are realized by the decoders as logical disjunction of the necessary count of sequential sates of the counter.

4. Conclusions

The synthesized circuits are general solution to the presented problem. They characterize with flexibility in achieving concrete practical solutions, they do not need any change in the synthesis methods for the synthesis of finite state machines with hardwired rules and they execute the control algorithm at times, which are at greatest degree close to the times of asynchronous execution. In the same these structures keep the simplicity of the technical implementation typical for the absolutely synchronous state machines. These characteristics greatly ease the practical configuration of real systems, in which they can be embedded.

References

[1]. Тянев Д.С., *ОРГАНИЗАЦИЯ НА КОМПЮТЪРА – том втори*, ISBN 978-954-20-0413-4, Издателство “ТУ-Варна”, 2008 год..

[2]. Соловьев В. В., *Проектирование цифровых систем на основе ПЛИС*, ISBN 5-93517-043-4, Издательство “Горячая линия-Телеком”, Тула, 2001 год.

[3]. Тянев Д.С., *Организация на компютъра (проектиране на логически структури),* ISBN 954-20-0259-0, Издателство “ТУ-Варна”, 2004 год.

[4]. Parag K. Lala, *Principles of Modern Digital Design*, A&M University–Texarkana, Texas, 2000.

[5]. Jurg Flum, Erich Grudel, Thomas Wilke, *Logic and Automata - History and Perspectives*, ISBN 978-90-5356-576-6, Amsterdam University Press, 2008.

[6]. Sivarama P. Dandamudi, *Fundamentals of Computer Organization and Design*, ISBN 0-387-9521I-X , Springer-Verlag, New York, Inc., 2003.
[7]. Charles H. Roth, *Fundamental of Logic Design,* 5-th Ed., Brooks/Cole Publishing, 2003.
[8]. Pong P. Chu, *RTL hardware design using VHDL*, John Wiley & Sons, Inc. ISBN 13: 978-0-471-72092-8, 2006.
[9]. Brian Holdsworth, Clive Woods, *Digital Logic Design*, Fouth Edition, Newnes, 2008.
[10]. Enoch O. Hwang, *Digital Logic and Microprocessor Design with VHDL*, Brooks/Cole, ISBN 0-534-46593-5, 2005.
[11]. Wakerly J. F., *Digital Design – Principles and Practices*, Third Edition, Prentice-Hall, 2000.
[12]. Charles Kime, Thomas Kaminski, *Logic and Computer Design Fundamentals*, Pearson Education, Inc. 2008.
[13]. Ian Grout, *Digital systems design with FPGAs and CPLDs*, ISBN-13: 978-0-7506-8397-5, Newnes, 2008.
[14]. Jack Ganssle, *The Art of Designing Embedded Systems*, Second Edition, Newnes 2008, ISBN 978-0-7506-8644-0.
[15]. Al Daves, Steven M. Nowick , *An Introduction to Asynchronous Circuit Design*, UUCS-97-013, 1997.
[16]. Kenneth J. Breeding, *Digital Design Fundamentals*, Second Edition, Prentice Hall, 1992.
[17]. Vojin Oklobdzija, *Digital design and fabrication*, Taylor & Francis Group, 2008, ISBN 978-0-8493-8602-2.
[18]. Stephen Broun, Zvonko Zvanesic, *Fundamentals of Digital Logic with VHDL Design*, Second Edition, McGraw Hill, ISBN 0-07-249938-9.
[19]. Richard Sharp, *Higher-Level Hardware Synthesis*, Lecture Notes in Computer Science, 2004, Springer-Verlag, ISBN: 3-540-21306-6.