A Method of Reduction of the Microinstructions of Synchronous Digital Systems

Al-Dahleh M. Z., Shehabat I. M.
Philadelphia University, Jordan.
ishehabat@yahoo.com

Journal of Digital Information Management

ABSTRACT: *In this paper, the author presents a method to synthesize the Algorithm State Machines (ASM) for synchronous digital systems using Combined Addressing. In This proposed method, the microinstructions are divided into subsets, and thus the Numbers of microinstructions are apparently minimized and the throughput time of the automation is reduced.*

Categories and Subject Descriptors
C.5 [Computer System Implementation]; B.4.3 [Interconnections]; Asynchronous/synchronous operation

General Terms
Synchronous Digital Systems,Algorithm state machines

Keywords: Digital Systems, Microinstruction (MI), Synchronized Systems, Combined Addressing, Throughput.

Received 11 Nov. 2006; Revised and accepted 17 Feb. 2007

1. Introduction

In the late 1940's Maurice Wilkes of Cambridge University started work on a stored program computer called EDSAC (Electronic Delay Storage Automatic Calculator). During this effort, Wilkes recognized that the sequencing of control signals within the computer was similar to the sequencing actions required in a regular program and that he could use a stored program to represent the sequences of control signals. Since Maurice Wilkes proposed the principle of Microprogramming in his paper published In 1951[1], several methods of designing Micro-Programmed Automata (MPA) have been developed based on Micro-Program and Hardware design [2, 3, 4, 5]. Various methods have been developed as Logical Scheme Algorithm, Transition Array and Algorithm State Machines (ASM). The ASM have been widely used in practice [2, 3, 6, 8] as a convenient way of specifying the sequence of procedural steps and decision paths of an algorithm.

Designing MPA by using ASM can be realized on Micro-Programmed as well as Hardware programmed logic [2, 3, 6, 9, 10]. In Micro-Programmed logic, several types of addressing have been used as Compulsory, Combined and Natural addressing [5, 6, 10]. The 1980's proved to be crucial turning point for traditional microprogramming. Without exception, modern-day RISC microprocessors are hardwired. The MC68000 macroinstruction set has also downsized, so that the implementation of the remaining core macroinstruction set can be hardwired [11]. Even recent processors for IBM System/390 mainframes are hardwired [12]. Microcode is still used, however, in the numerous Intel x86 compatible microprocessors like the Pentium 4 and AMD K6-2 [13].

The Combined addressing of Microinstructions is used in Micro-Programmed logic devices because it is a convenient and easy way of writing the MicroPrograms as well as the small size of the microinstruction [6, 9, 10].

This work presents a new method to minimize the number of the MI. The quantity of MI is considered a major disadvantage of the combined addressing because of the large number of unconditional transfers that must be entered to insure the correct function of the automaton [4, 5, 7, 8]. The present method is based on dividing the MI into subsets to minimize the MI numbers.

2. Existing Scheme

The Combined Microinstructions (CMI) are used in MicroProgram Memory (MPM). The control words can be stored, with the following format, in various devices such as Programmable Logic Array (PLA), read only memory (ROM) and others. See Figure 1 below:

FMO	FLC	FFA

Figure 1. Microinstruction Format

The microinstruction format shown in Figure 1 has the following fields [4, 6, 8]:

a. Field of Micro-Operation (FMO) contains the codes of executed Micro- Operations y_n $y_n \in Y$ = {y_1 ,..., y_N} 1withword length N.
b. Field of Logical Conditions (FLC) contains the code of logic conditions (LC) $\chi_1 \in X = \{\chi_l ,..., \chi_L\}$, Which defines the transitions of the MicroProgram With word length *L*.
c. Field of False Addresses (FFA) contains the False Addresses of transition, which is initiated, if the LC equals zero, with word length R.

The automaton with combined addresses presented in Figure 2 below includes thefollowing structure [2, 3, 4, 8]:-

a. The Address Circuit (CA), which analyses the fields FLC depending on the value of LC $\chi_1 \in X = \{\chi_l ,..., \chi_L\}$ forms one of the two signals φ_1 or According to .1and .2 the Register Address MicroProgram Memory (RAMPM) responds by transferring either FFA or by adding one to the current address.
b. RAMPM gives the first address of the MicroProgram on signal φ_0.
c. MicroProgram Memory (MPM) contains the MicroProgram with the word length r = int ($\log_2 m$)
d. The Control Signals Circuit (CSC) forms the micro-operation $y_n \in Y$ = {y_1 ,..., y_N} depending on field FMO, as well as the stop signal Z.

With the start signal, the automaton gives the φ_0 -signal which gives the first address of the MicroProgram according to the LC and the input X. The CA determines whether to pass the FFA or to add one to the current address by accessing φ_1 or φ_2 signals.

The RAMPM determines the next address and passes the FMO field to the CSC. The CSC defines the output microcommand and the stop signal Z.

To design the MPM, a sequence of conditions must be considered [5, 8, 9, 10], as the sequence of the LC according to the ASM. In case of a sequence of operational node transfers to the same decision node, an unconditional transfer must be added to the MPM, which leads to expand and complicate the micro-program.

To reduce the number of microinstructions a new additional addressing circuit is suggested based on dividing the logical conditions into two subsets. This leads to expand the microinstruction format but decreases the number of microinstructions stored in the MPM.

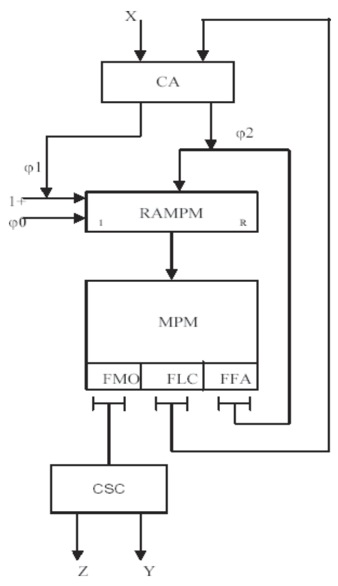

Figure 2. Standard Structure Automaton with Combined Addressing

3. The Proposed Method

To reduce the number of microinstructions, an additional addressing circuit is proposed. This leads to a new format of microinstruction as shown in Figure 3 below as well as a new structure of the automata as shown in Figure 4 below.

The additional field in the microinstruction format will be called henceforth the Flag Field and will be assigned as FF. This field determines the output of circuit one CA1 or circuit two CA2.

FF	FMO	FLC	FFA

Figure 3. Proposed Microinstruction Format

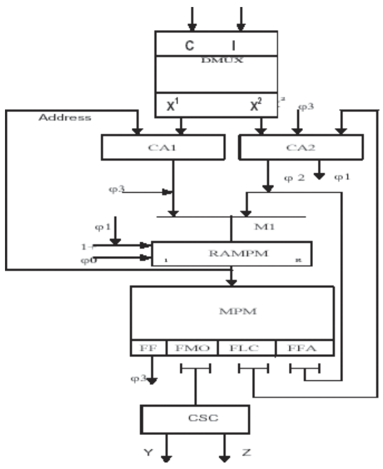

Figure 4. Modified Structure Automaton with Compulsory Addressing

The additional circuit CA2 contains all the logical conditions that depend only on one logical condition while the address circuit CA1 formulates all the transitions that depend on more than one logical condition.

Therefore the main idea is to divide the microinstructions into two subsets as below:

1. T^1 contains all the possible transitions that contain more than one logical condition $x_1 \in X^1$ and will be stored in CA1.
2. T^2 contains all the remaining transitions $x_1 \in X^2$ including the unconditional transitions and will be stored in CA2.

All Microinstructions

Figure 5. Existing Strategy (CA)

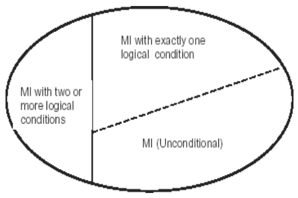
MI with exactly one logical condition
MI with two or more logical conditions
MI (Unconditional)

Figure 6. Dividing Strategy (CA1 + CA2)

I give the following algorithm DMI to divide the microinstruction set T into two disjoint subsets T^1 and T^2

Algorithm DMI (T, n, T^1, T^2)

```
1. input T = {$i_1, i_2, ..., i_n$}   :: input MI set T
2. input n                  :: input the cardinality
3. $T^1 = \phi, T^2 = \phi$  :: initialize with null set
4. for j = 1 to  n  do
     4.1. if $i_j$ contains two or more logical conditions.
     4.2.      then $T^1 = T^1 \cup \{i_j\}$
5. $T^2 = T - T^1$
6. Stop
7. End
```

Moreover, the synthesis of ASM is derived as in the example in G1 Figure 7 below:

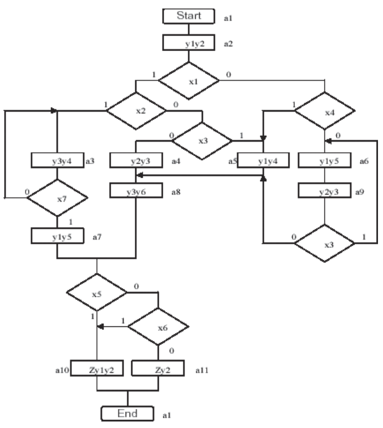

Figure 7. Example of ASM G_1

To synthesize the given example we have to build a State Table (ST) [2, 3, 7, 9] that includes the following columns:-

1. a_m is the present initial conditional state MPA, $a_m \in A$, where $A = \{a_1,\dots, a_M\}$ the set of state conditions.
2. $K(a_m)$ the corresponding conditional state code am has the length of $R = \text{int}\{\log_2 M\}$.
3. a_s, $K(a_s)$ are the next state and its corresponding code respectively.
4. x_l is the Logical conditions that determine the transfer from (am, as), and it is an element of the logical conditions (LC) $X = \{x_1,\dots,x_L\}$.
5. Yn is the output signal transfer by (a_m, a_s), $yn \subseteq Y$; where $Y = \{y_1,\dots,y_N\}$

set of micro-operations (MO).

6. $h=\overline{I,H}$ is the number of transfers. From the ASM in Figure 5, the following states $A=\{a_1,\dots,a_{11}\}$can be defined, to code these states the logical conditions must be taken in consideration to insure the correct work of the model. The states will be coded in a way that if there is a sequence of states following LC that equals to one the code between these states are increased by one. From the ASM we can notify the following states $<a_3,a_7,a_{10}>$, as well as $<a_6,a_9>$, it should be mentioned that the start and the end states have the same code. The remaining states are coded sequentially starting from the last code. In this case, the following codes are designated: a_1-0000; a_3-0001; a_7-0010; a_{10}-0011; a_6-0100; a_9-0101; the remaining sets are designated with the following codes:
a_2-0110; a_4-0111; a_5-1000; a_8-1001; a_{11}-1010. After that, the ST Table 1 is built, analyzed and divided into two subtables: T^1 contains all the possible transition states that have more than one logical condition. T^2 contains the remaining conditions.

a_m	$K(a_m)$	a_s	$K(a_s)$	X_l	Y_n	H
a_1	0000	a_2	0110	1	-	1
a_2	0110	a_3	0001	x_1x_2	y_1y_2	2
		a_4	0111	$x_1x_2' x_3'$		3
		a_5	1000	$x_1x_2' x_3$		4
		a_5	1000	$x_1' x_4$		5
		a_6	0101	$x_1' x_4'$		6
a_3	0001	a_3	0001	x_7'	y_3y_4	7
		a_7	0010	x_7		8
a_4	0111	a_8	1001	1	y_2y_3	9
a_5	1000	a_8	1001	1	y_1y_4	10
a_6	0101	a_9	0100	1	y_1y_5	11
a_7	0010	a_{10}	0011	x_5	y_1y_5	12
		a_{10}	0011	$x_5' x_6$		13
		a_{11}	1010	$x_5' x_6'$		14
a_8	1001	a_{10}	0011	x_5	y_3y_6	15
		a_{10}	0011	$x_5' x_6$		16
		a_{11}	1010	$x_5' x_6'$		17
a_9	0100	a_6	0101	x_3	y_2y_3	18
		a_8	1001	x_3'		19
a_{10}	0011	a_1	0000	1	$Z\,y_1\,y_2$	20
a_{11}	1010	a_1	0000	1	$Z\,y_2$	21

Table 1. State Table of ASM G1

Through, analyzing Table 1 the following sets can be defined:

1. $T^1 = \{ T(a_2), T(a_7), T(a_8) \}$ that includes the following logical conditions $X^1 = \{x_1,x_2,x_3,x_4,x_5,x_6\}$.
2. $T^2 = \{ T(a_1), T(a_3), T(a_4), T(a_5), T(a_6), T(a_9), T(a_{10}), T(a_{11}) \}$ includes the following logical conditions $X^2 = \{x_3, x_7\}$.

From these subsets, the ST for CA1 and CA2 are built. For the realization of CA1 a PLA can be used, and a direct PLA table must be built [2, 3, 4, 9, 10] as shown in Table 2 below.

To realize MPM, the field FLC of microinstructions should contain only three codes 01, 10 and 00 corresponding to X7, X3 and the unconditional transition (UnT) respectively.

The circuit CA2 is shown in Figure 6. Signals φ_1 and $\varphi_2$are realized by the CA2, Signal φ_3 can be defined from the field FF. If FF equals zero, signal φ_3 is enabled And CA2 is activated otherwise the contents of fields FLC and FFA are ignored and CA1 is activated. The contents of the microprogram memory MPM are shown in Table 3 below. It is worth mentioning that the unconditional transition works on the Negative value of FF.

x_1	x_2	x_3	x_4	x_5	x_6	$K(a_m)$	$K(a_s)$	Note
Inputs							Outputs	
1	1	*	*	*	*	0 1 1 0	0 0 0 1	T(a2)
1	0	0	*	*	*	0 1 1 0	0 1 1 1	
1	0	1	*	*	*	0 1 1 0	1 0 0 0	
0	*	*	1	*	*	0 1 1 0	1 0 0 0	
0	*	*	0	*	*	0 1 1 0	0 1 0 1	
*	*	*	*	1	*	0 0 1 0	0 0 1 1	T(a7)
*	*	*	*	0	1	0 0 1 0	0 0 1 1	
*	*	*	*	0	0	0 0 1 0	1 0 1 0	
*	*	*	*	1	*	1 0 0 1	0 0 1 1	T(a8)
*	*	*	*	0	1	1 0 0 1	0 0 1 1	
*	*	*	*	0	0	1 0 0 1	1 0 1 0	

Table 2. Direct PLA Table for Circuit CA1

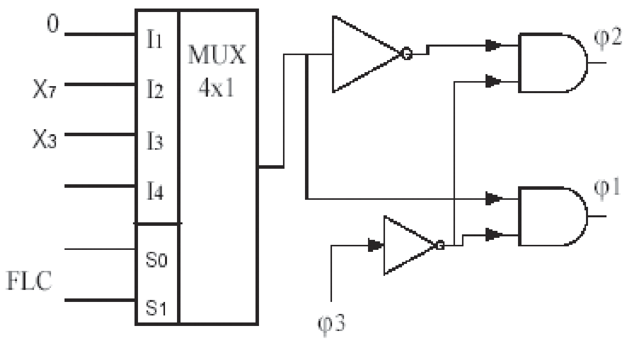

Figure 8. The Logical Circuit CA2

The modified automaton operates as follows:

Signal φ_0 gives the first address to the MPM. According to the FF the multiplexer M_1 selects either CA_1 or CA_2. If φ_3 equals zero, CA_2 is selected and depending on the FLC either .1 or .2 is activated. On the other hand, when φ_3 equals one, CA_1 is activated and FLC and FFA are ignored.

Address	FF	FMO	FLC	FFA	Note	
		Z y_1 y_2 Y_3 y_4 y_5 y_6	x_1x_0		a_m	
0000	0	0 0 0 0 0 0 0	00	0110	a_1	UnT a_2, CA_2 activated
0001	0	0 0 0 1 1 0 0	01	0001	a_3	CA_2 activated
0010	1	0 1 0 0 0 1 0	**	****	a_7	CA_1 activated
0011	0	1 1 1 0 0 0 0	00	0000	a_{10}	Unt a_0, CA_2 activated
0100	0	0 0 1 1 0 0 0	10	1001	a_9	CA_2 activated
0101	0	0 1 0 0 0 1 0	00	0100	a_6	UnT a_9, CA_2 activated
0110	1	0 1 1 0 0 0 0	**	****	a_2	CA_1 activated
0111	0	0 0 1 1 0 0 0	00	1001	a_4	UnT a_8, CA_2 activated
1000	0	0 1 0 0 1 0 0	00	1001	a_5	UnT a_8, CA_2 activated
1001	1	0 0 0 1 0 0 1	**	****	a_8	CA_1 activated
1010	0	1 0 1 0 0 0 0	00	0000	a_{11}	UnT a_0, CA_2 activated

Table 3. Modified Contents Table of MPM for Automaton with Combined Addresses

In analyzing the classical method of synthesizing the automaton with combined addressing microinstructions using the ASM G_1, the following parameters can be stated:

- The number of microinstructions = 17 bits;
- The word length fields of FMO = 7 bits, FLC = 3 bits and FFA =5 bits;
- The word length of microinstructions = 15 bits;
- The capacity of MPM = 17x15 = 255 bits.

Using the modified method for synthesizing ASM G1, the following parameters can be stated:

- The number of microinstructions = 11 bits;
- The word length fields of FMO = 7 bits, FLC = 2 bits, FFA = 4 bits and FF = 1 bit;
- The capacity of MPM = 11x14 = 154 bits.

4. Conclusion

In this work, a new method of reducing the microinstructions of synchronous digital systems is proposed. The main advantages of this design can be listed as follows:

- Reducing the number of microinstructions and the executed time of the micro-program occur according to the additional circuit of addresses the additional circuit of addresses minimizes the number of microinstruction that leads.
- To decrease the executed time of the algorithm.
- Reducing the capacity of the micro-program memory and the number of chips needed for its realization.

On the other hand, there are some points that must be taken in consideration:

- The cycle time of microinstruction is increased because of the additional multiplexers But, in general, as the number of microinstructions is decreased the general throughput of the algorithm is also decreased. In other word, as the new circuit CA_2 has been added the time cycle required to execute MI increases. In contrary, as long as the new method reduces the general number of MI, the time required to execute the algorithm decreases.
- The CA_1 looses its universality, as each ASM needs special design for the CA_2.

Such combination of advantages and defects permit us to generalize the method to be applied for realizing resident firmware, including high-branched micro-programs.

References

[1] Wilkes, M.V (1951). The Best Way to Design an Automated Calculating Machine, Machester University Computer Inaugural Conf., 1951, 16-18.

a. Reprinted In: MV Wilkes,"The Genesis of Microprogramming, *IEEE Annals of the History of Computing*, 8 (3) 1986. 116-126.

[2] Rafiquzzaman, M., Chandra, R.(1988). Modern Computer Architecture, West Publishing Company.

[3] Mano, M. M (2002). Digital Design, Third Edition, New Jersey: Prentice-Hall International Inc.

[4] Barkalov, A.A (1990). Synthesis of Control MicroProgram Device with ImplicitRepresentation of Logic Conditions, USiM. #1. 38-41.

[5] Baranov S.I., Sklarov V.A (1986) Digital Devices on Programmed LSIC with Array Structure, Moscow: Radio and Communication, 272.

[6] Barkalov A. A (1992). Synthesis of MicroProgram Control Devices. Donetsk: Donetsk State Technical University, Ukraine.

[7] Chao, H., Ong, S (1992).Design Optimization for Control Units Realized with PLA's, *IEEE transactions on Computers*, 3. 1091-1112.

[8] Jain R.P (1986). Modern Digital Electronics, McGraw-Hall.

[9] Majorov S.A., Novikov G.I (1974). Principles of Organization Digital Machines, Moscow: Engineering.

[10] Palagen, A. V (1993). Micro Processors Systems Manipulating Information. Kiev: Science.

[11] Motorolla ColdFire manuals. Available on-line at http://www.freescale.com/

[12] Web, C.F, Liptay, J.S (1997). A high-Frequency Custom CMOS S/390 Microprocessor, IBM Journal of Research and Development, 41 (4/5) 463-473.

[13] Shriver, B., Smith, B (1998). The anatomy of High-Performance Microprocessor: A System Perspective. Los Alamitos, CA .IEEE Computer Society Press.