하계학술회의
논문85-8-8

개인용 컴퓨터를 사용한 순차제어기의 설계

Personal Computer Based Design for the Sequential Machines

조 동 섭* 이화여대
김 민 환 서울대
김 준 현 조선대

Abstract

This paper deals with the personal computer based design for the sequential machines. Most part of sequential machine design have been implemented by using general purpose microprocessors in order to obtain the specific functions required for their system. But,they have some difficulties in design stages. Knowledge of systems design method and high technolegy are basically applied to all the design stages. Therefore ready made microcomputer system for personal use, personal computer, can be transformed to sequential machines by using the corresponding softwares and built-in personal computer input/output ports.

Following the state transition diagram or table,we can obtain the ROM type of sequential machines directly and need not to design input/output interface except actuators and samplers because of capability of personal computer. Our main purpose of this design method are quick, flexible, reliable, modifiable circuit design of the sequential machines. In this paper, we use APPLE-II plus personal computer as target machine.

I. introduction

The design process presented in this paper is directed toward bridging the gap between hardware engineering and software engineering. Software engineers have been using the top-down or structured design methods since Dijkstra introduced the five basic structuring constructs in the late 1960's. Structured programming results in readable and modifiable systems. Thus, its concepts are applied to design of sequential machines.

However, hardware engineers have not fully exploited the general principles with respect to structured design. In this paper, basic concepts required in the design process are presented. The design process is different from other structured design methods such as KARL or AHPL in that basic structured representaion of sequential machines is transformed to object program which is used in personal computer.

II. Basic architecture of sequential machines

From the behavioral point of view, clocked sequential circuits and pulse mode sequential circuits are synchronous sequential circuits in which every state is a stable state and no transient states may exist. Fundamental mode sequential circuits are asynchrous sequential circuits, which have transient state (unstable states) in addition to the usual stable states. Fig.1 shows the classifications of sequential machines.

Various type of commonly used flip-flops are introduced in implementing the sequential machines. Based on the application tables of the flip-flops, which are derived from their characteristic functions, a general procedure for synthesizing sequential machines by a clocked circuit is presented

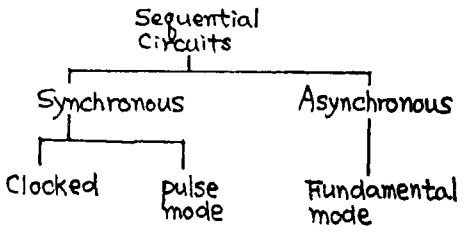


Fig.1 classes of sequential machine.

in Fig. 2.

In realizing sequential machines using the pulse mode and fundamental mode circuits, there are no problem in design of pulse mode sequential circuits because of their similar operation. But the design of fundamental mode sequential circuits are different from those of clocked sequential circuits. The undesirable transient phenomena of fundamental circuits (races and harzards) affects the reliability.

III. Structured approach to sequential machines

Fig. 3 present the basic structuring constructs, used in state transition diagram.

IV. Implementation of basic structuring constructs using personal computer

The basic structuring constructs discussed in chapter III are transformed to corresponding object module. Here are examples of these transformations shown in Fig.4.

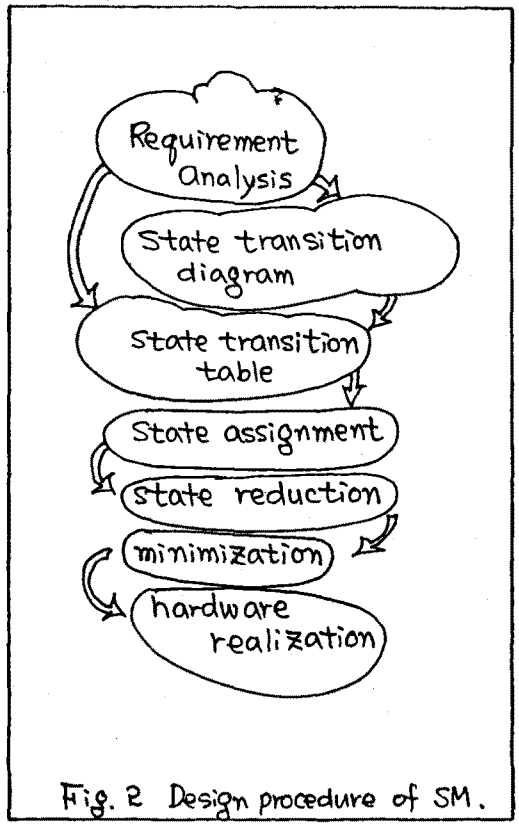
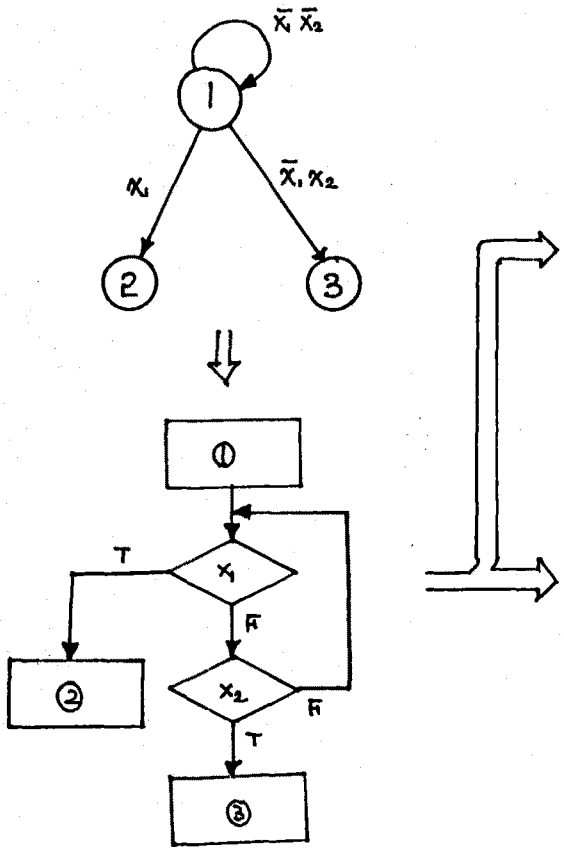


Fig. 2 Design procedure of SM.




```
STATE1 :  OUTPUT ( DATA1)
          IF CONDITION1 = X1
             THEN
                OUTPUT(DATA2)
                GOTO STATE2
          IF CONDITION2 = X2
             THEN
                OUTPUT(DATA3)
                GOTO STATE3
```

```
STATE1: OUTPUT(DATA1)
        REPEAT
          IF CONDITION1 = X1
             THEN
                OUTPUT(DATA2)
                GOTO STATE2
        UNTIL CONDITION2=X2
        OUTPUT(DATA3)
        GOTO STATE3
```

Fig.4 Example of transformation of state diagram.

V. Conclusions

In this paper, we propose the structured procedure implementing the sequential machines by using popular 8-bit personal computer.

Our main objective is to present the realizations of sequential machines from the state transition diagram or state transition table directly.

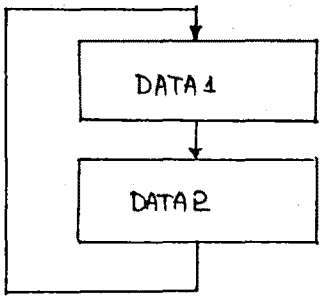
In order to transformation of state flow diagram to ROM types of sequential circuits, first of all, structured state transitions are obtained from the given state transition diagram or conceptual modeling of state transitions.

References

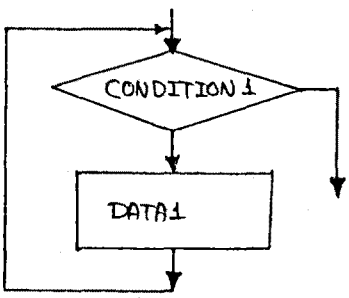
[1] F.J.Hill, R.E.Swanson,M.Masud,and Z.Navabi,"Structured specification with a procedural Hardware Description Language." IEEE Tran. on Computer., vol.c-30,no.2, pp.157-160, Feb, 1981.

[2] Wen C., Lin, "Microprocessor-based digital system design," Proc. of IEEE ,vol.65,no.8, pp.1138-1160,august 1977.

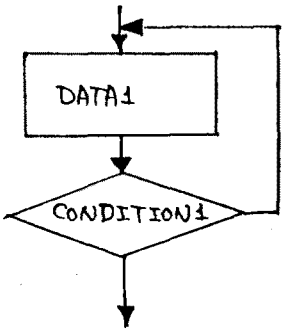
[3] Apple II reference manuals.



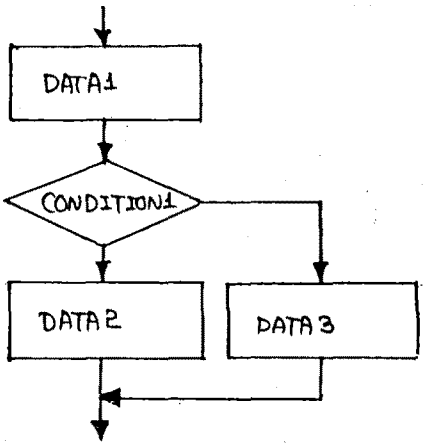
3-(a) Repeat ~ forever



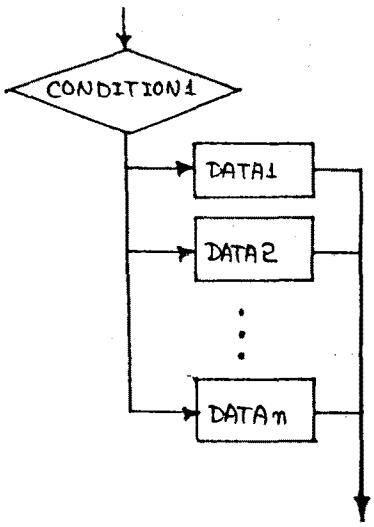
3-(b) While ~ do ~



3-(c) Repeat ~ Until ~



3-(d) If ~ then ~ else ~



3-(e) Case ~ of ~

Fig. 3 Basic structuring constructs.

Appendix : Code generation for the basic constructs.

```
 1 ;
 2 ; REPEAT statements FOREVER ;
 3 ;
 4           ORG $
 5 ;
 6 REPEAT:
 7 ;
 8 ;         OBJECT CODES FOR
 9 ;         REPEAT-FOREVER
10 ;
11           JMP REPEAT
12 ;
13           END
```

```
 1 ;
 2 ; WHILE expression DO statements ;
 3 ;
 4           ORG $
 5 ;
 6 WHILE:
 7 ;
 8 ;         OBJECT CODES FOR
 9 ;         expression
10 ;
11           BEQ ENDWHILE
12 ;
13 ;         OBJECT CODES FOR
14 ;         statements
15 ;
16           JMP WHILE
17 ENDWHILE:
18 ;
19           END
```

```
 1 ;
 2 ;
 3 ; IF expression THEN statements#1
 4 ;                ELSE statements#2 ;
 5 ;
 6           ORG $
 7 ;
 8 ;         OBJECT CODES FOR
 9 ;         expression
10 ;
11           BEQ ELSE
12 ;
13 ;         OBJECT CODES FOR
14 ;         statements#1
15 ;
16           JMP ENDIF
17 ;
18 ELSE:
19 ;
20 ;         OBJECT CODES FOR
21 ;         statements#2
22 ;
23 ENDIF:
24 ;
25           END
```

```
 1 ;
 2 ; IF expression THEN statements ;
 3 ;
 4           ORG $
 5 ;
 6 ;         OBJECT CODES FOR
 7 ;         expression
 8 ;
 9           BEQ ENDIF
10 ;
11 ;         OBJECT CODES FOR
12 ;         statements
13 ;
14 ENDIF:
15 ;
16           END
```

```
 1 ;
 2 ; REPEAT statements UNTIL expression
 3 ;
 4           ORG $
 5 ;
 6 REPEAT:
 7 ;
 8 ;
 9 ;         OBJECT CODES FOR
10 ;         statements
11 ;
12 ;         OBJECT CODES FOR
13 ;         expression
14 ;
15           BEQ REPEAT
16 ;
17           END
```

```
 1 ;
 2 ; FOR var:= exp#1 TO exp#2 DO ;
 3 ;
 4           ORG $
 5 ;
 6 ;
 7 ;         OBJECT CODES FOR
 8 ;         exp#1
 9 ;
10           JSR STATUS#1
11 ;
12 ;         OBJECT CODES FOR
13 ;         exp#2
14 ;
15           JSR STATUS#2
16 ;
17 FOR:
18           JSR SUBTRACT
19           BMI ENDFOR
20 ;
21 ;         OBJECT CODES FOR
22 ;         statements
23 ;
24           JSR INCvar
25           JMP FOR
26 ;
27 ENDFOR:
28 ;
29           END
```

```
 1 ;
 2 ; CASE expression OF
 3 ;       constants#1 : statements#1 ;
 4 ;       constants#2 : statements#2 ;
 5 ;           .               .
 6 ;           .               .
 7 ;           .               .
 8 ;       constants#m : statements#m ;
 9 ;
10           ORG $
11 ;
12 ;         OBJECT CODES FOR
13 ;         expression
14 ;
15           JSR PUSHEND
16           JSR TEST
17 ;
18 ;         DW n#1
19 ;         DW constants#1
20 ;
21           BNE CASE2
22 ;
23 ;         OBJECT CODES FOR
24 ;         statements#1
25 ;
26           RTS
27 ;
28 CASE2:
29           JSR TEST
30 ;
31 ;         DW n#2
32 ;         DW constants#2
33 ;
34           BNE CASE3
35 ;
36 ;         OBJECT CODES FOR
37 ;         statements#2
38 ;
39           RTS
40 ;
41 CASE3:
42 ;
43 ;
44 ;
45 CASEm:
46           JSR TEST
47 ;
48 ;         DW n#m
49 ;         DW constants#m
50 ;
51           BNE CASE-ERR
52 ;
53 ;         OBJECT CODES FOR
54 ;         statements#m
55 ;
56           RTS
57 ;
58 ENDCASE:
59 ;
60           END
```