Lucrarea 5.
Automat cu stari finite

Automatele secventiale pot fi de doua tipuri: de tip **Moore**, respectiv de tip **Mealy**, diferenta dintre ele constand in aceea ca in cazul celor de tip Mealy iesirile depind atat de starea curenta cat si de intrarile curente, pe cand in cazul celor de tip Moore iesirile depind numai de starea curenta. Un model general de automat secvential, consta dintr-o retea combinationala care genereaza iesirile precum si starea urmatoare, si un registru de stare (realizat de obicei cu bistabile D) care memoreaza starea curenta.

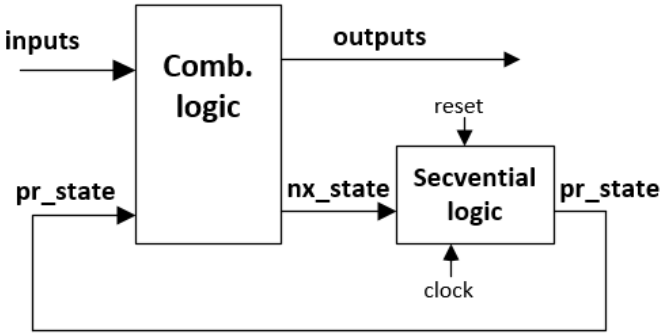


Fig. 5.1 structura generală automat cu stări finite

Se definesc doua functii F si G:
F -> functie care determina starea urmatoarea
G-> functie care detemina iesirea
F si G sunt implementate folosind exclusive circuite combinationale.

In cazul automatelor Mealy si Moore starea urmatoare este determinate de functia F, care are ca intrari starea curenta(pr_state) si intrarile automatului:
Starea urmatoare=F(starea curenta,intrari)

Automat secvential Mealy

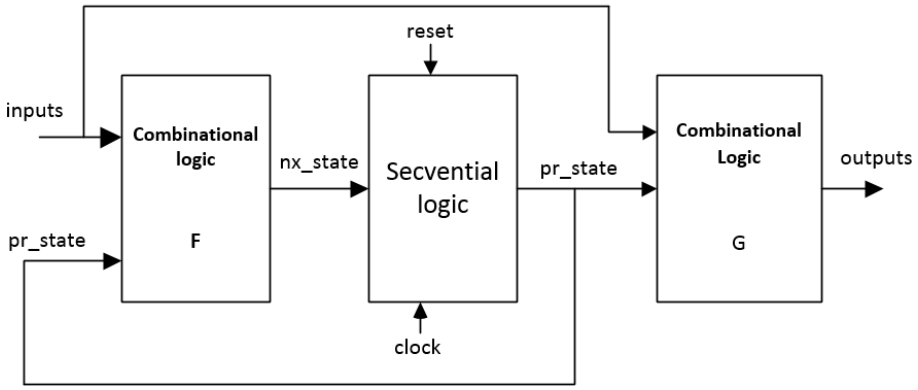
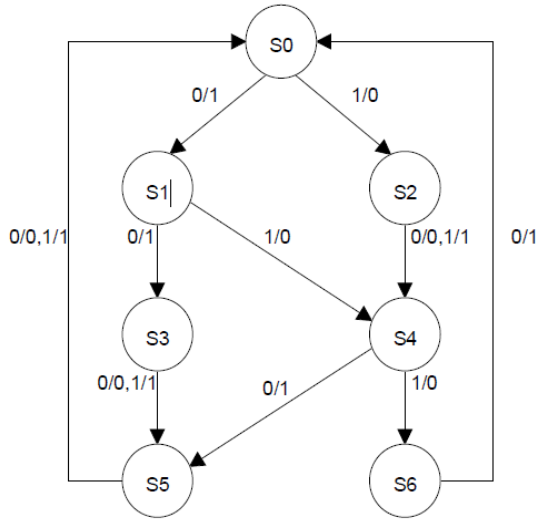


Fig. 5.2 Structura generala automat Mealy

In cazul automatului Mealy, iesirea acestuia depinde de starea curenta cat si de intrarile curente: Iesirea=G(starea curenta,intrari)

Functionarea unui astfel de automat este urmatoarea: dupa un anumit interval de timp (caracteristic retelei combinationale) de la modificarea intrarilor X, are loc setarea corespunzatoare a iesirilor Z, iar noua stare a retelei este memorata sincron cu tactul in registrul de stare. Deoarece noua stare este adusa la intrarile automatului, procesul se poate repeta pana in momentul atingerii unei stari stabile.

Consideram automatul secvential de tip Mealy (cu o singura intrare, X, si o singura iesire, Z), a carui diagrama si tabel de stare sunt prezentate in figura urmatoare:



PS	NS		Z	
	X=0	X=1	X=0	X=1
S0	S1	S2	1	0
S1	S3	S4	1	0
S2	S4	S4	0	1
S3	S5	S5	0	1
S4	S5	S6	1	0
S5	S0	S0	0	1
S6	S0	-	1	-

PS-present state
NS-next state

Fig. 5.3 Diagrama si tabel de stare problema 1

Diagrama de stare contine informatia din tabelul de stare/iesire reprezentata intr-o forma grafica. Pentru fiecare stare este alocat un nod simbolizat printr-un cerc, in fiecare nod fiind inscris numele starii.
Sagetile sau arcele de cerc reprezinta tranzitii.Fiecare sageata care paraseste un nod reprezinta o tranzitie spre o alta stare.
Daca, spre exemplu, automatul se afla in starea initiala, S0 si intrarea X are valoarea 0, atunci noua stare va fi S1 si iesirea Z va deveni 1, iar daca X are valoarea 1, atunci noua stare va fi S2, iesirea Z devenind 0.

Observatie: Toate tranzitiile dintr-o stare in alta se desfasoara sincron cu tactul, iar informatia continuta in etichetele sagetilor care marcheaza tranzitiile, este de forma:

<val. intrare>/<val. iesire>
[, *<val. intrare>/<val. iesire>*].

Pentru starea curenta si cea viitoare se va defini un nou tip de date de tip enumerare:
TYPE my_new_type **IS** (S0, S1, S2, S3,S4,S5,S6);

Reteaua combinationala si registrul de stare pot fi modelate utilizand cate un proces.
In randurile urmatoare este prezentat un sablon de implementare in VHDL a unui automat cu stari finite. Se considera un automat sincron pe front crescator al tactului precum si un semnal de reset asincron prioritar activ pe 1.

```
ENTITY <nume_e > IS
PORT (
        I: IN BIT;
        reset, clock:IN BIT;
        O: OUT <data_type>);
END < nume_e >;

ARCHITECTURE <nume_arh> OF < nume_e > IS
TYPE state IS (s0, s1, s2, s3, ...);
SIGNAL pr_state, nx_state: state;
BEGIN

PROCESS (reset, clock)
BEGIN
IF (reset='1') THEN
pr_state <= s0;
ELSIF (clock'EVENT AND clock='1') THEN
pr_state <= nx_state;
END IF;
END PROCESS;

PROCESS (input, pr_state)
BEGIN
        CASE pr_state IS
                WHEN s0 =>
                        IF (i = <value>) THEN
                                O <= <value>;
                                nx_state <= <Sx>;
                        ELSE ...
                        END IF;

                WHEN s1 =>
                        IF (i = ...) THEN
                                O <= <value>;
                                nx_state <= <Sx>;
                        ELSE ...
                        END IF;

                WHEN s2 => ……..

                ...
        END CASE;
END PROCESS;
END < nume_arh >;
```

Problema 1)

Implementati in VHDL automatul cu stari finite din fig. 5.3 utilizand sablonul prezentat antarior.

Construiti un Test Bench astfel incat sa avem urmatoarele tranziti:

S0 - S1 - S3 - S0 - S1 - S4 - S5 - S0 - S2 - S4 - S6 - S0

Automat secvential Moore

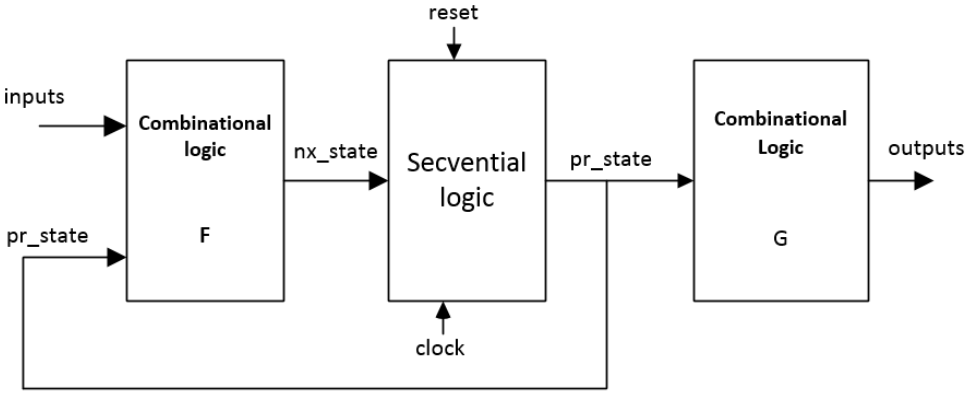


Fig. 5.4 Structura generala automat Moore

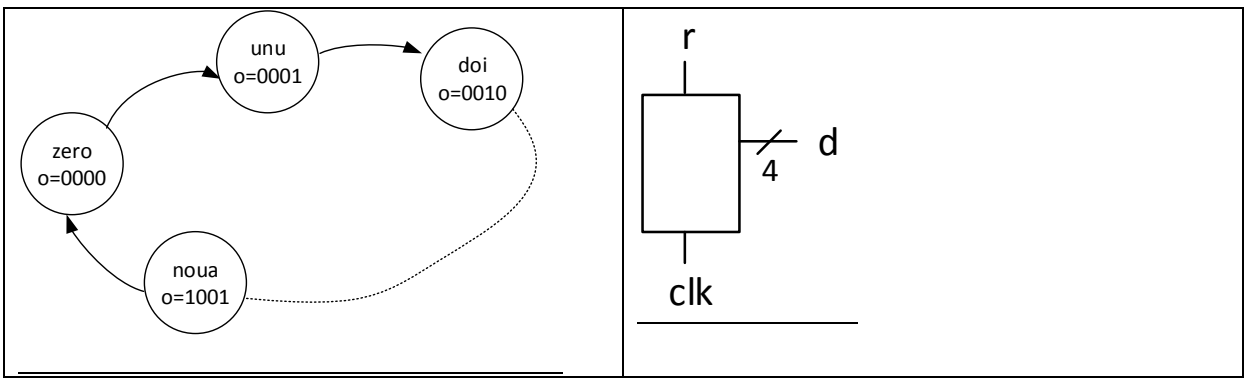
In cazul automatului Moore, iesirea acestuia depinde doar de starea curenta: *Iesirea=G(starea curenta)*

Diagrama de stare pentru un automat de tip Moore contine valorile variabilelor de iesire inscrise in interiorul nodurilor, acestea fiind functii de stare.

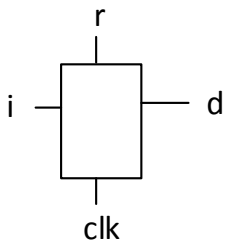
Pentru implementarea automatului Moore se poate folosi acelasi sablon VHDL, sunt necesare modificari doar la nivelul atribuirii valori iesirii.

```
PROCESS (input, pr_state)
BEGIN
        CASE pr_state IS
                WHEN s0 =>  O <= <value>;
                        IF (input = ...) THEN
                                nx_state <= <Sx>;
                        ELSE ...
                        END IF;

                WHEN s1 => O <= <value>;
                        IF (input = ...) THEN
                                nx_state <= <Sx>;
                        ELSE ...
                        END IF;

                WHEN s2 => ……..

                ...
        END CASE;
        END PROCESS;
```

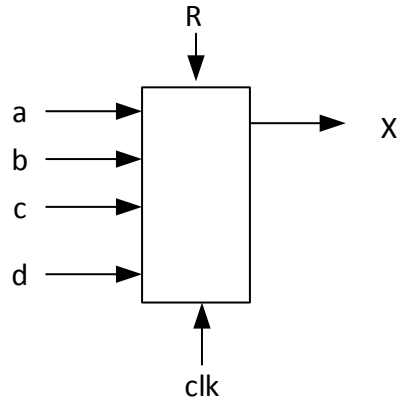
P.2) Implementați în VHDL un numărător BCD utilizând un automat Moore.

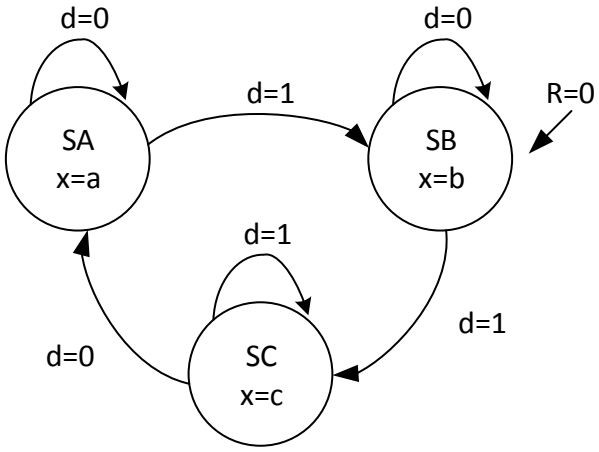


P.3) Implementați un detector de secvență care sa detecteze dacă ultimii biți receptionați sunt „1101”, se va implementa utilizând un automat Moore.
Daca ultimii 4 biți receptionați pe intrarea **i** sunt „1101” iesirea **d** devine 1.
ASF este sincron pe front crescător, cu reset sincron activ pe 1.



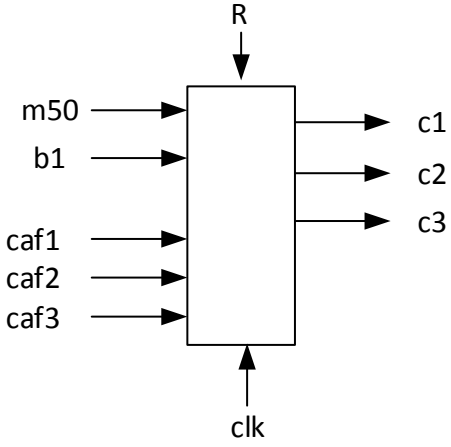
P.4) Impementați automat cu stări finite descris prin următoarea diagramă de stare. ASF este sincron pe front descrescător, cu reset asincron activ pe 0.





P.5) Implementați partea de comandă a unui automat de cafea.

- Automatul acceptă monede (50 bani) si bancnote (1 leu)
- Poate prepară trei tipuri de cafea (2lei)
- Comanda produsului se face după ce a fost plătit produsul
- Automatul nu da rest



Bibliografie

1. V.A. Pedroni, Circuit Design with VHDL, MIT Press, 2004.
2. C. H. Roth, Digital System Design using VHDL, PWS Publishing Company, 1998