S8. Automate de tip Mealy

Proiectarea mașinilor cu stări finite în Verilog

1. Scopul laboratorului

Implementarea Verilog în cinci pași a FSM-urilor

2. Desfășurarea laboratorului

2.1 Diagrama tranzițiilor de stare al FSM-ului

În teoria computațională, mașina Mealy este un FSM (Finite State Machine) a cărei valori de ieșire sunt determinate atât de starea curentă cât și de intrările actuale (în contrast cu mașina Moore a cărei valori de ieșire sunt determinate în mod absolut de starea curentă). Mașina Mealy este astfel un FSM deterministic.

În Figura 1 este prezentat un FSM de tip Mealy care are următoarele stări interne:

- S_0 – stare implicită
- S_1 – stare de tranziție unidirecțională
- S_2 – stare de tranziție bidirecțională

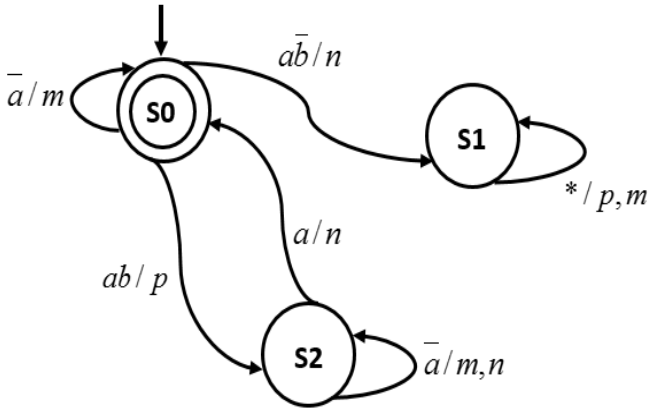

Fig.1 – State Transition Diagram Example

Interpretarea expresiilor de pe arcele de tranziții poate fi rezumată astfel:

condiție logică/ ieșire activată.

2.2 Proiectarea Verilog a mașinilor cu stări finite în 5 pași

2.2.1 Construirea constantelor cu nume pentru fiecare stare a mașinii

Prima etapă în implementarea FSM-urilor constă în declararea constantelor utilizând **localparam** și fiecăreia îi este atribuită o valoare distinctă, pe numărul corespunzător de biți. După o analiză sumară a FSM-ului prezentat (vezi **Fig. 1**) putem remarca ușor faptul că stările interne ale mașinii, notate de la 0 la 2 pot fi codificate pe un număr de 2 biți. În consecință vom scrie:

```
localparam S0 = 2'd0; // in memorie se va stoca 00

localparam S1 = 2'd1; // in memorie se va stoca 01

localparam S2 = 2'd2; // in memorie se va stoca 10
```

2.2.2 Definirea semnalelor de tip registru

Definirea a 2 semnale **reg** care vor păstra starea curentă (***st***) și starea următoare (***st_next***). Cele două semnale sunt declarate pe același număr de biți ca și constantele de stare.

```
reg [1:0] st;

reg [1:0] st_next;
```

2.2.3 Construirea stării următoare într-un bloc always combinațional

Din diagrama tranzițiilor de stare se construiește starea următoare, *st_next*, într-un bloc **always combinațional** (a se consulta Verilog modeling using always and initial blocks – Flavius Opriţoiu).

Folosind instrucțiunea *case(st)*, se prevăd ramuri pentru fiecare stare, în vederea evaluării condițiilor logice de tranziție la altă stare.

Important: Sunt eliminate ramurile de cod fără atribuiri lui *st* (de ex. ramuri ***else*** nefolosite)

```
always @ (*) begin

  st_next = S0; // initial starea urmatoare este starea implicita

  case (st)
```

```
    S0: if (!a)
        st_next = S0;
        else
        if (!b)
        st_next = S1;
    S1: // se va completa cu cod aici
    S2: // se va completa cu cod aici
    endcase
end
```

2.2.4 Construirea ieșirilor mașinii într-un block always combinațional

Din diagrama tranzițiilor de stare se construiesc ieșirile mașinii, într-un bloc **always combinațional**. Similar generării stării următoare, folosind instrucțiunea *case(st)*, se prevăd ramuri pentru fiecare stare, în care sunt activate ieșirile corespunzătoare fiecărei tranziții.

Important: Pentru evitarea ramurilor de cod fără atribuiri ale unora sau ale tuturor ieșirilor mașinii, toate ieșirile vor fi inițializate la valorile lor implicite anterior instrucțiunii *case(st)*.

```
always @ (*) begin
    m = 1'd0 ; // activ pe valoarea logica 1
    n = 1'd1 ; // activ pe valoarea logica 0
    p = 1'd0 ; // activ pe valoarea logica 1
    case (st)
```

```
S0: if (!a)
     m = 1'd1 ; // se activeaza iesirea corespunzatoare
    else
    if (!b)
     n = 1'd0 ; // iesirea se activeaza pe valoarea 0
S1: begin
     m = 1'd1 ;
     p = 1'd1 ;
    end
S2: // se va completa cu cod aici
```

2.2.5 Actualizarea stării curente într-un block always secvențial

În ultimul pas al proiectării FSM-urilor se va actualiza starea curentă într-un bloc **always secvențial**. Cu mici modificări, codul de mai jos poate fi folosit pentru implementarea oricărei mașini cu stări finite, cu condiția respectării etapelor anterioare.

```
always @ (posedge clk, negedge rst_b)  begin
  if (!rst_b)
    st <= INIT_ST ; // in cazul nostru INIT_ST va fi S0
  else
    st <= st_next ;
end
```

3. Referințe bibliografice

[Vlad12] M. Vlăduțiu, *Computer Arithmetic: Algorithms and Hardware Implementations*, 2012th ed. Springer, 2012.

[Opri17] F. Oprițoiu, *Interfațarea componentelor digitale pentru transfer de date*, Lab. 2017.