Codificarea One Hot pentru implementarea FSM-urilor

Oprițoiu Flavius
flavius.opritoiu@cs.upt.ro

3 decembrie 2023

Introducere

Obiective:

- Construirea mașinilor cu stări finite utilizând codificarea One Hot

De citit:

- Mircea Vlăduțiu: "Computer Arithmetic : Algorithms and Hardware Implementations", Anexa B [Vlad12]

Codificarea One Hot pentru un FSM având s stări, utilizează s elemente de stocare. Fiecare element de stocare are asociat o stare $\Rightarrow$ la fiecare moment de timp, unul și doar unul din cele s elemente de stocare este activ (are ieșire activă).

Implementarea prin codificarea One Hot, deși utilizează mai multe elemente de stocare decât cea prin metoda tabelului de stare, are avantajele unei implementări directe și a unei depanări facile.

Divizor de frecvență

Un divizor de frecvență având factorul de divizare n, primește la intrare un semnal de tact de frecvență f_{in} și generează la ieșirea un semnal de frecvență $\frac{f_{in}}{n}$. Semnalul generat nu trebuie sa aiba factorul de umplere de 50% (semnalul este activ jumatate din perioada si inactiv cealalta jumatate).

Dacă n este de forma 2^k, se folosește un numărător binar pe k biți, semnalul de tact divizat fiind cel mai semnificativ bit al ieșirii numărătorului. Dacă n nu este o putere a lui 2, se vor folosi numărătoare modulo-n.

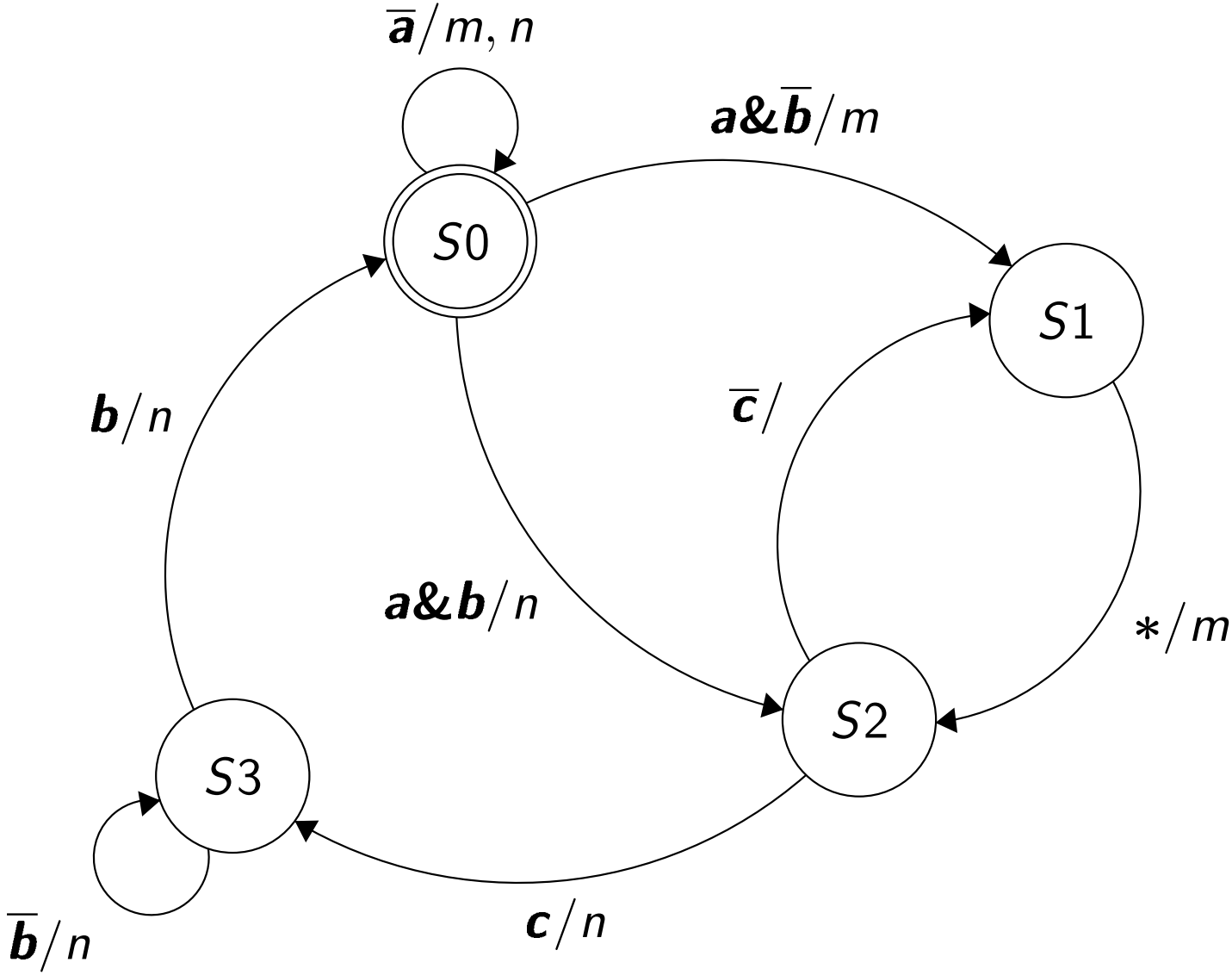
Interfața unui divizor de frecvență:

- intrarea *clk*: semnalul având frecvența f_{in}
- intrarea *rst_b*: intrare de inițializare, opțională
- ieșirea *dclk* (divided clock): semnalul de frecvență $\frac{f_{in}}{n}$

Studiu de caz

Implementarea unei mașini Mealy descrisă prin diagrama tranzițiilor de stare

Exercițiu: Să se implementeze următoarea mașină cu stări finite:



Studiu de caz (contin.)

Implementarea unei mașini Mealy descrisă prin diagrama tranzițiilor de stare

Designul folosește 4 bistabile de tip D: FF_0, FF_1, FF_2 și FF_3, cu intrările D_i și ieșirile Q_i, asociate celor 4 stări S_0, S_1, S_2 și S_3.

La fiecare moment unul și doar unul din cele 4 bistabile este activ, având ieșirea Q_i egală cu 1. Starea curentă este indicată de bistabilul cu ieșirea activă. Dacă Q_1 este activ, starea curentă este S_1, dacă Q_4 este 1 atunci S_4 este starea curentă, șamd.

La intrările D_i se conectează ecuațiile booleene care activează stările S_i. Designul FSM-ului se reduce la scrierea acestor ecuații. Condițiile în care starea următoare va fi S_1:

- starea curentă este S_0, a este 1 și b este 0, **sau**
- starea curentă este S_2 și c este 0

Deci $D_1 = Q_0 \cdot a \cdot \overline{b} \ + \ Q_2 \cdot \overline{c}$

Implementarea prin codificarea One Hot a FSM-urilor

Pasul 1

Pentru fiecare stare a FSM-ului se definește o constantă de stare, S_i, utilizând `localparam`. Fiecăre constantă de stare are o valoare distinctă, între 0 și $s - 1$ (s reprezentând numărul total de stări).

Pentru exercițul propus, aceste constante de stare pot fi definite ca mai jos:

```
1 localparam S0 = 2;
2 localparam S1 = 0;
3 localparam S2 = 3;
4 localparam S3 = 1;
```

Implementarea prin codificarea One Hot a FSM-urilor

Pasul 2

Se definesc stările curentă, *st*, și următoare *st_nxt* ca 2 vectori binari pe *s* biți (*s* fiind numărul de stări ale FSM-ului). Semnalul *st*, va fi declarat de tip `reg` iar *st_nxt* de tip `wire`.

Pentru problema de rezolvat, cele 2 semnale sunt definte astfel:

```
1 reg [3:0] st;
2 wire [3:0] st_nxt;
```

Implementarea prin codificarea One Hot a FSM-urilor

Pasul 3

Bitului *st_nxt[S_i]* (S_i este una din constantele de stare definite în pasului 1) îi este atribuită expresia booleană care activează starea S_i. De remarcat că aceste ecuații booleene sunt construite după modelul prezentat în slide-ul 5, cu precizarea că semnalele D_i devin *st_nxt[S_i]* iar semnalele Q_i sunt înlocuite cu *st[S_i]*.

Pentru problema de rezolvat, starea următoare se generează astfel:

```
1  assign st_nxt[S0] = ( st[S0] & (~a) ) |
2                      ( st[S3] & b );
3  assign st_nxt[S1] = ( st[S0] & a & (~b) ) |
4                      ( st[S2] & (~c) );
5  assign st_nxt[S2] = st[S1] |
6                      ( st[S0] & a & b );
7  assign st_nxt[S3] = ( st[S2] & c ) |
8                      ( st[S3] & (~b) );
```

Implementarea prin codificarea One Hot a FSM-urilor

Pasul 4

Ieșirilor FSM-ului le sunt atribuite expresii booleane construite direct din diagrama tranzițiilor de stare, enumerând prin operatorul logic SAU, toate condițiile în care ieșirea respectivă devine activă.
Exemplu: ieșirea *m* este activă:

- în starea S_0, dacă *a* este 0, **sau**
- în starea S_0, dacă *a* este 1 și *b* este 0, **sau**
- în starea S_1

Pentru problema de rezolvat, ieșirile sunt generate astfel:

```
1  assign m = ( st[S0] & (~a) ) |
2             ( st[S0] & a & (~b) ) |
3             st[S1];
4  assign n = ( st[S0] & (~a) ) |
5             ( st[S0] & a & b ) |
6             ( st[S2] & c ) |
7             ( st[S3] & (~b) ) |
8             ( st[S3] & b );
```

Implementarea prin codificarea One Hot a FSM-urilor

Pasul 5

Se actualizează starea curentă într-un bloc `always` secvențial. La fiecare front declanșator al semnalului de tact, semnalul de stare următoare devine starea curentă. Activarea intrării de reset declanșază aducerea automatului în starea inițială care pentru exemplul considerat, este starea S_0.

Pentru problema de rezolvat, actualizarea stării curente se face după modelul următor:

```
1  always @ (posedge clk, negedge rst_b)
2    if (rst_b == 0) begin
3      st <= 0;
4      st[S0] <= 1;
5    end else
6      st <= st_nxt;
```

References

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