Riconoscitore di stringhe

by Antonio Bernardini

Table of contents

1 Homework

1.1 Introduzione

Per questo homework, occorre progettare una *rete sequenziale* in grado di accettare sequenze di caratteri che rispettino il formato $ba(dc)*bac$, dove la notazione $*$ indica che possono esserci n ripetizioni della sottosequenza dc con $n \in [0, \infty)$. La sequenza di caratteri in input è infinita.

L'alfabeto di input è costituito dai caratteri $I = \{a, b, c, d\}$ rappresentati con le seguenti codifiche:

I	x_1	x_0
a	0	0
b	0	1
c	1	1
d	1	0

Table 1. Tabella con le codifiche dei caratteri di input

L'alfabeto di output è costituito dai caratteri $O = \{\text{no}, \text{sì}\}$ rappresentati con le seguenti codifiche:

O	z
no	1
sì	0

Table 2. Tabella con le codifiche dei caratteri di output

La rete dovrà essere costruita come *macchina di Mealy*. La *rete sequenziale* dovrà essere in grado di accettare o rifiutare correttamente le stringhe ammissibili secondo la definizione dell'espressione ed inoltre deve essere in grado di accettare correttamente anche sequenze *sovrapposte*.

2 Soluzione

2.1 Introduzione

Abbiamo a disposizione tutte le indicazioni per procedere alla progettazione della *rete sequenziale* però come al solito cercherò di implementare una soluzione piuttosto compatta tramite l'ausio del modulo ROM (Read Only Memory).

2.2 Design della rete sequenziale

Sappiamo che l'alfabeto di input è $I = \{a, b, c, d\} = \{00, 01, 11, 10\}$, mentre l'alfabeto di output è $O = \{\text{no}, \text{sì}\} = \{1, 0\}$. Pertanto costruisco una relazione tra gli stati utilizzando la *macchina di Mealy*:

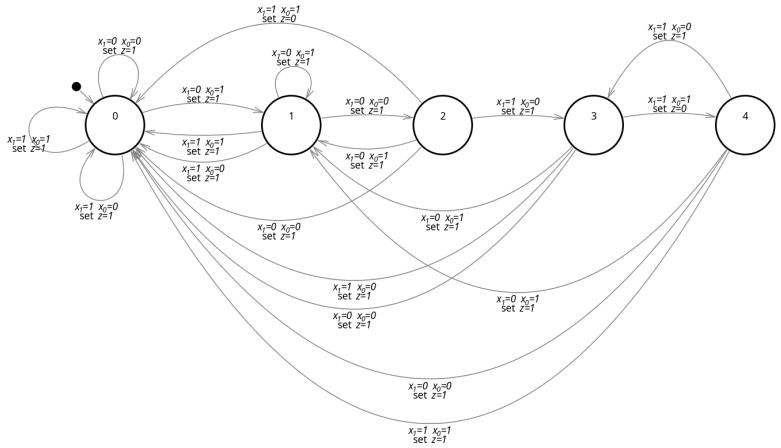

Figure 1. Relazione tra gli stati utilizzando la *macchina di Mealy*

Quindi, poichè ci sono 5 stati, abbiamo bisogno di 3 bit per la rappresentazione corretta degli stati (che indicheremo con y_2, y_1, y_0) e delle transizioni (che indicheremo con y_2', y_1', y_0'), che avviene tramite la seguente tabella (creabile dall'apposito menù `Create > State Transition Table`):

*Table

File New Edit Create K-Map

y2n	y1n	y0n	x0	x1	y2n+1	y1n+1	y0n+1	z
0	0	0	0	0	0	0	0	1
0	0	0	0	1	0	0	0	1
0	0	0	1	0	0	0	1	1
0	0	0	1	1	0	0	0	1
0	0	1	0	0	0	1	0	1
0	0	1	0	1	0	0	0	1
0	0	1	1	0	0	0	1	1
0	0	1	1	1	0	0	0	1
0	1	0	0	0	0	0	0	1
0	1	0	0	1	0	1	1	1
0	1	0	1	0	0	0	1	1
0	1	0	1	1	0	0	0	0
0	1	1	0	0	0	0	0	1
0	1	1	0	1	0	0	0	1
0	1	1	1	0	0	0	1	1
0	1	1	1	1	1	0	0	0
1	0	0	0	0	0	0	0	1
1	0	0	0	1	0	1	1	1
1	0	0	1	0	0	0	1	1
1	0	0	1	1	0	0	0	1
1	0	1	0	0	x	x	x	x
1	0	1	0	1	x	x	x	x
1	0	1	1	0	x	x	x	x
1	0	1	1	1	x	x	x	x
1	1	0	0	0	x	x	x	x
1	1	0	0	1	x	x	x	x
1	1	0	1	0	x	x	x	x
1	1	0	1	1	x	x	x	x
1	1	1	0	0	x	x	x	x
1	1	1	0	1	x	x	x	x
1	1	1	1	0	x	x	x	x
1	1	1	1	1	x	x	x	x

Figure 2. Tabella delle codifiche degli input, degli stati, delle transizioni e dell'output

Ora conviene esportare la tabella di verità in formato `HEX`, sempre dall'apposito menù `File > Export > HEX` (salvando il file con il nome `mealy.hex`), per poterla utilizzare all'interno del singolo modulo ROM avente le seguenti impostazioni:

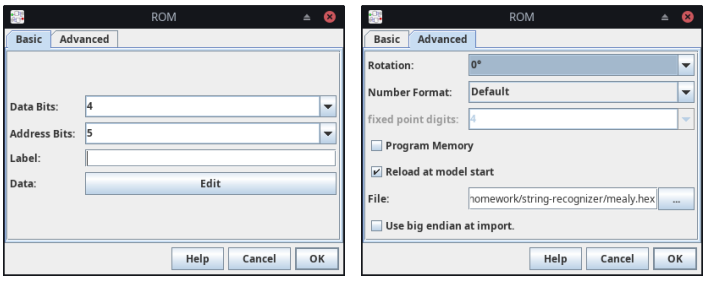

Figure 3. Impostazioni di ogni singolo modulo ROM

dove è importante impostare la voce `Reload at model start`, indicando il path del file `mealy.hex`. Chiaramente trattandosi di una *rete sequenziale* avremo bisogno di un modulo *clock*, in comune a 4 flip-flop D, ai quali verranno collegate in input le transizioni e, attraverso il componente *tunnel*, verranno portate le uscite Q_i agli stati y_2, y_1 e y_0 di input. Pertanto il circuito finale è il seguente:

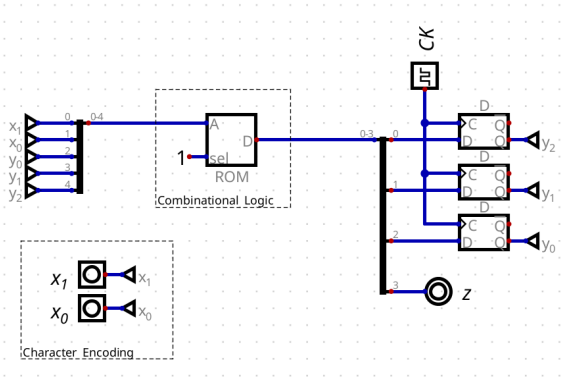

Figure 4. Circuito finale

Si noti che l'uso di un *bus dati* a 5 bit in input e a 4 bit in output implica che ogni input e ogni output non sono inseriti casualmente in tale *bus dati*, ma rispettano la disposizione della tabella di verità di Fig. 2.