Capitolo 2: Reti Sequenziali Elementari

Esercizio 3: Riconoscitore di sequenze

Progetto e architettura

Un riconoscitore di sequenza ha come obiettivo quello di individuare determinati pattern di segnali di ingresso, ovvero specifiche sequenze di ‘0’ e ‘1’. Una macchina di questo tipo può essere facilmente modellata come un automa a stati finiti di Mealy.

Il progetto richiede che sia riconosciuta la sequenza “101” e che la modalità sia determinata da un segnale (**M)**. In particolare, per M=0 la macchina osserva il segnale in input (**i**) alla ricerca di sequenze non sovrapposte e per M=1 vengono invece ricercata sequenze parzialmente sovrapposte. Ad esempio, per la successione di ingressi “00010101000”, nel primo caso si identifica una sola sequenza “101”, nel secondo caso due.

Il modulo **riconoscitore** ha quattro segnali di ingresso: i (dato di ingresso), **RST** (segnale di reset), **CLK** (segnale di clock), e **M** (segnale di controllo). L'uscita del modulo è rappresentata dal segnale **Y**.

La architettura comportamentale del codice implementa la logica del riconoscitore attraverso due process: un processo (stato\_uscita) la cui sensitivity list include i, M e il segnale **stato\_corrente** che calcola le funzioni di transizione dello stato e dell’uscita, ovvero aggiorna opportunamente **stato\_prossimo** e Y in base agli attuali valori degli altri segnali; l’altro processo (mem) è sensibile al solo segnale di clock e se il reset è alto porta il sistema nello stato iniziale altrimenti se si è visto un fronte di salita di CLK si modifica stato\_corrente mettendolo uguale a stato\_prossimo. Si osservi che il reset è, in questo contesto, sincrono.

Immagine che contiene schizzo, disegno, diagramma, Line art

Descrizione generata automaticamenteSono stati identificati cinque stati potenziali, chiamati come **S0**, **S1**, **S2**, **S3** e **S4**; tuttavia, il significato attribuito a ciascuno di questi stati, così come il numero effettivamente utilizzato, è strettamente dipendente dalla modalità M. Successivamente, vengono presentati i diagrammi a stati del progetto per entrambe le possibili tecniche di riconoscimento.

Implementazione

**riconoscitore.vhd**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity riconoscitore is

    port( i: in std\_logic;

          RST, CLK: in std\_logic;

          M : in std\_logic;

          Y: out std\_logic

    );

end riconoscitore;

architecture Behavioral of riconoscitore is

type stato is (S0, S1, S2, S3, S4);

signal stato\_corrente : stato := S0;

signal stato\_prossimo : stato;

begin

stato\_uscita: process(stato\_corrente, i, M)

    begin

       if(M'event) then

            stato\_prossimo <= S0;

            Y <= '0';

       elsif(M='1') then

       case stato\_corrente is

            when S0 =>

                if( i = '0' ) then

                    stato\_prossimo <= S0;

                    Y <= '0';

                else

                    stato\_prossimo <= S1;

                    Y <= '0';

                end if;

            when S1 =>

                if( i = '0' ) then

                    stato\_prossimo <= S2;

                    Y <= '0';

                else

                    stato\_prossimo <= S1;

                    Y <= '0';

                end if;

            when S2 =>

                if( i = '0' ) then

                    stato\_prossimo <= S0;

                    Y <= '0';

                else

                    stato\_prossimo <= S0;

                    Y <= '1';

                end if;

            when others =>

                    stato\_prossimo <= S0;

                    Y <= '0';

        end case;

        elsif(M='0') then

        case stato\_corrente is

            when S0 =>

                if( i = '0' ) then

                    stato\_prossimo <= S1;

                    Y <= '0';

                else

                    stato\_prossimo <= S2;

                    Y <= '0';

                end if;

            when S1 =>

                stato\_prossimo <= S3;

                Y <= '0';

            when S3 =>

                stato\_prossimo <= S0;

                Y <= '0';

            when S2 =>

                if( i = '0' ) then

                    stato\_prossimo <= S4;

                    Y <= '0';

                else

                    stato\_prossimo <= S3;

                    Y <= '0';

                end if;

            when S4 =>

                if( i = '0' ) then

                    stato\_prossimo <= S0;

                    Y <= '0';

                else

                    stato\_prossimo <= S0;

                    Y <= '1';

                end if;

            when others => stato\_prossimo <= S0;

                    Y <= '0';

        end case;

        end if;

end process;

mem: process (CLK)

begin

    if( CLK'event and CLK = '1' ) then

        if( RST = '1') then

           stato\_corrente <= S0;

        else

           stato\_corrente <= stato\_prossimo;

        end if;

   end if;

end process;

end Behavioral;

Simulazione

Il testbench è stato progettato con l’obiettivo di simulare il comportamento del file "riconoscitore.vhd"; esso include un process che genera il clock (**CLK\_process**) e uno che fornisce valori in ingresso, stimoli, a un’istanza del “design under test” (**stim\_proc**).

Durante la prima fase M è basso e viene fatto variare il valore del segnale i al fine di testare le transizioni di stato e osservare l'output del sistema.

Immagine che contiene schermata, Software multimediale

Descrizione generata automaticamente

Successivamente, il segnale di controllo M viene impostato su '1' e vengono forniti nuovi stimoli a i per studiare il circuito anche in questa modalità.

Immagine che contiene schermata, Software multimediale, software

Descrizione generata automaticamente

Come si può vedere, in entrambi i casi l’uscita si alza solo quando viene riconosciuta una sequenza “101” secondo l’approccio scelto e i passaggi di stato sono coerenti con la descrizione dell’automa precedentemente presentato.

**riconoscitore\_tb.vhd**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity riconoscitore\_tb is

end riconoscitore\_tb;

architecture Behavioral of riconoscitore\_tb is

component riconoscitore

    port (

         i: in std\_logic;

          RST, CLK: in std\_logic;

          M : in std\_logic;

          Y: out std\_logic

        );

    end component;

   signal i : std\_logic := '0';

   signal CLK : std\_logic := '0';

   signal RST : std\_logic := '0';

   signal M : std\_logic := '0';

   signal Y : std\_logic;

   constant CLK\_period : time := 10 ns;

BEGIN

   dut: riconoscitore port map(

          i => i,

          CLK => CLK,

          RST => RST,

          M => M,

          Y => Y

        );

   CLK\_process :process

   begin

        CLK <= '0';

        wait for CLK\_period/2;

        CLK <= '1';

        wait for CLK\_period/2;

   end process;

   stim\_proc: process

   begin

        wait for 100 ns;

        M<='0';

        i<='0';

        wait for 10 ns;

        i<='0';

        wait for 10 ns;

        i<='1';

        wait for 10 ns;

        i<='0';

        wait for 10 ns;

        i<='1';

        wait for 10 ns;

        i<='0';

        wait for 10 ns;

        i<='1';

        wait for 10 ns;

        i<='0';

        wait for 100 ns;

        M<='1';

        i<='0';

        wait for 50 ns;

        i<='0';

        wait for 10 ns;

        i<='1';

        wait for 10 ns;

        i<='0';

        wait for 10 ns;

        i<='1';

        wait for 10 ns;

        i<='1';

        wait for 10 ns;

        i<='0';

        wait for 10 ns;

        i<='1';

        wait;

   end process;

end;

Sintesi su board di sviluppo

Premendo il pulsante B2 collegato al pin P17 si seleziona la modalità M a partire dallo switch S2 associato al pin L16 e si mette a ‘0’ l’ingresso; altrimenti se si preme il bottone B1 del pin N17 si campiona il valore dell’input dallo switch J15. Quando viene riconosciuto l’inserimento di una sequenza del tipo “101” si accende il led H17. Per resettare il sistema è sufficiente premere il pulsante al pin M18.

**riconoscitore\_board.vhd**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity riconoscitore\_board is

    Port (

        S1, S2, B1, B2 : in STD\_LOGIC;

        RST, CLK : in STD\_LOGIC;

        LED : out STD\_LOGIC

    );

end riconoscitore\_board;

architecture Behavioral of riconoscitore\_board is

    signal i, M : STD\_LOGIC := '0';

    signal Y : STD\_LOGIC := '0';

    COMPONENT riconoscitore

        Port (

            i : in STD\_LOGIC;

            RST, CLK : in STD\_LOGIC;

            M : in STD\_LOGIC;

            Y : out STD\_LOGIC

        );

    end COMPONENT;

begin

    riconoscitore\_inst : riconoscitore

        port map (

            i=>i,

            RST=>RST,

            CLK=>CLK,

            M=>M,

            Y=>Y

    );

    process(CLK)

    begin

    if rising\_edge(CLK) then

        if B2='1' then

            i<='0';

            M<=S2;

        elsif B1='1' then

            i<=S1;

        end if;

    end if;

    end process;

    LED <= Y;

end Behavioral;

## Clock signal

set\_property -dict { PACKAGE\_PIN E3 IOSTANDARD LVCMOS33 } [get\_ports { CLK }]; #IO\_L12P\_T1\_MRCC\_35 Sch=clk100mhz

create\_clock -add -name sys\_clk\_pin -period 10.00 -waveform {0 5} [get\_ports {CLK}];

##Switches

set\_property -dict { PACKAGE\_PIN J15 IOSTANDARD LVCMOS33 } [get\_ports { S1 }]; #IO\_L24N\_T3\_RS0\_15 Sch=sw[0]

set\_property -dict { PACKAGE\_PIN L16 IOSTANDARD LVCMOS33 } [get\_ports { S2 }]; #IO\_L3N\_T0\_DQS\_EMCCLK\_14 Sch=sw[1]

## LEDs

set\_property -dict { PACKAGE\_PIN H17 IOSTANDARD LVCMOS33 } [get\_ports { LED }]; #IO\_L18P\_T2\_A24\_15 Sch=led[0]

##Buttons

set\_property -dict { PACKAGE\_PIN N17 IOSTANDARD LVCMOS33 } [get\_ports { B1 }]; #IO\_L9P\_T1\_DQS\_14 Sch=btnc

set\_property -dict { PACKAGE\_PIN M18 IOSTANDARD LVCMOS33 } [get\_ports { RST }]; #IO\_L4N\_T0\_D05\_14 Sch=btnu

set\_property -dict { PACKAGE\_PIN P17 IOSTANDARD LVCMOS33 } [get\_ports { B2 }]; #IO\_L12P\_T1\_MRCC\_14 Sch=btnl