Progetto di Reti Logiche

Prof. Fornaciari, Prof. Palermo e Prof. Salice
Anno Accademico 2023 - 2024

SPECIFICA per lo svolgimento del progetto
(Aggiornamento del 22 Dicembre 2023)

In questa pagina potete trovare le modifiche fatte alla specifica del progetto dal suo primo rilascio. Tutti i cambiamenti con data annessa saranno riportati qui sotto e sono mantenuti in rosso nel testo.

Errata Corrige:

- 18 dicembre 2023: corretto il numero di parole K dell'esempio 1.
- 22 dicembre 2023: nell'esempio 1 è stato sostituito il valore PRIMA all'indirizzo 006E con il valore 00
- 22 dicembre 2023: Nella descrizione dell'interfaccia è stato sostituita la lettera W con K, quando si chiariva il segnale i_k
- 22 dicembre 2023: Negli esempi il valore K è stato portato su 3 cifre esadecimali (prima erano 4) coerenti con i 10 bit della sua lughezza.

Progetto di Reti Logiche

Prof. Fornaciari, Prof. Palermo e Prof. Salice
Anno Accademico 2023 - 2024

SPECIFICA per lo svolgimento del progetto
(Aggiornamento del 22 Dicembre 2023)

Descrizione generale

La specifica della "*Prova Finale (Progetto di Reti Logiche)*" per l'Anno Accademico 2023/2024 chiede di implementare un modulo HW (descritto in VHDL) che si interfacci con una memoria e che rispetti le indicazioni riportate nella seguente specifica.

A livello generale, il sistema legge un messaggio costituito da una sequenza di K parole W il cui valore è compreso tra 0 e 255. Il valore 0 all'interno della sequenza deve essere considerato non come valore ma come informazione "*il valore non è specificato*". La sequenza di K parole W da elaborare è memorizzata a partire da un indirizzo specificato (ADD), ogni 2 byte (e.g. ADD, ADD+2, ADD+4, …, ADD+2*(K-1)). Il byte mancante dovrà essere completato come descritto in seguito. Il modulo da progettare ha il compito di completare la sequenza, sostituendo gli zero laddove presenti con l'ultimo valore letto diverso da zero, ed inserendo un valore di "credibilità" C, nel byte mancante, per ogni valore della sequenza. La sostituzione degli zero avviene copiando l'ultimo valore valido (non zero) letto precedente e appartenente alla sequenza. Il valore di credibilità C è pari a 31 ogni volta che il valore W della sequenza è non zero, mentre viene decrementato rispetto al valore precedente ogni volta che si incontra uno zero in W. Il valore C associato ad ogni parola W viene memorizzato in memoria nel byte subito successivo (i.e. ADD+1 per W in ADD, ADD+3 per W in ADD+2,…). Il valore C è sempre maggiore o uguale a 0 ed è reinizializzato a 31 ogni volta che si incontra un valore W diverso da zero. Quando C raggiunge il valore 0 non viene ulteriormente decrementato.

ESEMPIO (valori in decimale):

- Sequenza di partenza (in grassetto i valori W):

128 0 **64** 0 **0** 0 **0** 0 **0** 0 **0** 0 **0** 0 **0** 0 **100** 0 **1** 0 **0** 0 **5** 0 **23** 0 **200** 0 **0** 0

- Sequenza finale

128 31 **64** 31 **64** 30 **64** 29 **64** 28 **64** 27 **64** 26 **100** 31 **1** 31 **1** 30 **5** 31 **23** 31 **200** 31 **200** 30

Funzionamento

Il modulo da implementare ha **3 ingressi primari, uno ad 1 bit (START), uno a 16 bit** (**ADD**) **e uno da 10 bit** (**K**), e **una uscita primaria da 1 bit (DONE).** Inoltre, il modulo ha un segnale di clock **CLK**, unico per tutto il sistema, e un segnale di reset **RESET** anch'esso unico per tutto il sistema. Tutti i segnali sono sincroni e devono essere interpretati sul fronte di salita del clock. L'unica eccezione è RESET che, invece, è asincrono.

All'istante iniziale, quello relativo al **reset** del sistema, l'uscita DONE deve essere 0. Quando RESET torna a zero, il modulo partirà nella elaborazione quando un segnale START in ingresso verrà portato a 1. Il segnale di START rimarrà alto fino a che il segnale di DONE non verrà portato alto; al termine della computazione (e una volta scritto il risultato in memoria), il modulo da progettare deve alzare (portare a 1) il segnale DONE che notifica la fine dell'elaborazione. Il segnale DONE deve rimanere alto fino a che il segnale di START non è riportato a 0. Un nuovo segnale START non può essere dato fin tanto che DONE non è stato riportato a zero.

Il modulo deve essere progettato considerando che prima del primo START=1 (e prima di richiedere il corretto funzionamento del modulo) verrà **sempre** dato il RESET (RESET=1). Prima del primo reset del sistema, il funzionamento del modulo da implementare è non specificato. Una seconda (o successiva) elaborazione con START=1 non dovrà invece attendere il reset del modulo. Ogni qual volta viene dato il segnale di RESET (RESET=1), il modulo viene re-inizializzato.

Quando il segnale di START viene posto ad 1 (e per tutto il periodo in cui esso rimane alto) sugli ingressi ADD e K vengono posti il primo indirizzo e la dimensione della sequenza da elaborare. Il modulo prima di alzare il segnale di DONE deve aggiornare la sequenza ed i relativi valori di credibilità al valore opportuno seguendo la descrizione generale del modulo.

Nota: Se il primo dato della sequenza è pari a zero, il suo valore rimane tale e il valore di credibilità deve essere posto a 0 (zero). Lo stesso succede fino al raggiungimento del primo dato della sequenza con valore diverso da zero.

Interfaccia del Componente

Il componente da descrivere deve avere la seguente interfaccia.

```
entity project_reti_logiche is
    port (
        i_clk   : in std_logic;
        i_rst   : in std_logic;
        i_start : in std_logic;
        i_add   : in std_logic_vector(15 downto 0);
        i_k     : in std_logic_vector(9 downto 0);

        o_done  : out std_logic;

        o_mem_addr : out std_logic_vector(15 downto 0);
        i_mem_data : in std_logic_vector(7 downto 0);
        o_mem_data : out std_logic_vector(7 downto 0);
        o_mem_we   : out std_logic;
        o_mem_en   : out std_logic
    );
end project_reti_logiche;
```

In particolare:

- il nome del modulo **deve essere** `project_reti_logiche` e deve essere presente **una sola architettura** per ogni entità; la violazione di queste indicazioni comporta l'impossibilità di eseguire il Test Bench e una conseguente valutazione di zero;
- i_clk è il segnale di CLOCK in ingresso generato dal Test Bench;
- i_rst è il segnale di RESET che inizializza la macchina pronta per ricevere il primo segnale di START;
- i_start è il segnale di START generato dal Test Bench;
- i_k è il segnale (vettore) K generato dal Test Bench rappresentante la lunghezza della sequenza;
- i_add è il segnale (vettore) ADD generato dal Test Bench che rappresenta l'indirizzo dal quale parte la sequenza da elaborare;
- o_done è il segnale DONE di uscita che comunica la fine dell'elaborazione;
- o_mem_addr è il segnale (vettore) di uscita che manda l'indirizzo alla memoria;
- i_mem_data è il segnale (vettore) che arriva dalla memoria e contiene il dato in seguito ad una richiesta di lettura;
- o_mem_data è il segnale (vettore) che va verso la memoria e contiene il dato che verrà successivamente scritto;
- o_mem_en è il segnale di ENABLE da dover mandare alla memoria per poter comunicare (sia in lettura che in scrittura);
- o_mem_we è il segnale di WRITE ENABLE da dover mandare alla memoria (=1) per poter scriverci. Per leggere da memoria esso deve essere 0.

APPENDICE: Descrizione Memoria
NOTA: **La memoria è già istanziata all'interno del Test Bench e non va sintetizzata**

La memoria e il suo protocollo può essere estratto dalla seguente descrizione VHDL che fa parte del test bench e che è derivata dalla User guide di VIVADO disponibile al seguente link:
https://www.xilinx.com/support/documentation/sw_manuals/xilinx2017_3/ug901-vivado-synthesis.pdf

```
-- Single-Port Block RAM Write-First Mode (recommended template)
--
-- File: rams_02.vhd
--
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity rams_sp_wf is
port(
  clk  : in  std_logic;
  we   : in  std_logic;
  en   : in  std_logic;
  addr : in  std_logic_vector(15 downto 0);
  di   : in  std_logic_vector(7 downto 0);
  do   : out std_logic_vector(7 downto 0)
);
end rams_sp_wf;

architecture syn of rams_sp_wf is
type ram_type is array (65535 downto 0) of std_logic_vector(7 downto 0);
signal RAM : ram_type;
begin
  process(clk)
    begin
    if clk'event and clk = '1' then
      if en = '1' then
        if we = '1' then
          RAM(conv_integer(addr)) <= di;
          do                      <= di after 2 ns;
        else
          do <= RAM(conv_integer(addr)) after 2 ns;
        end if;
      end if;
    end if;
  end process;
end syn;
```

ESEMPI

Gli esempi qui di seguito (valori in esadecimale) hanno lo scopo di esemplificare la relazione della sequenza di partenza (indicata come “prima”), cioè durante il periodo nel quale START assume valore 1, la configurazione finale (indicata come “dopo”) nel solo momento utile, cioè durante il periodo nel quale DONE assume valore 1. Il valore C è scritto dopo la sua parola W. Inizialmente C è sempre 0. K è il numero di parole W della sequenza mentre ADD è l'indirizzo nel quale si trova la prima parola W.

Esempio 1

K 00A

ADD 0064

Prima		Dopo	
ADD	W	ADD	W
0064	33	0064	33
0065	00	0065	1F
0066	00	0066	33
0067	00	0067	1E
0068	39	0068	39
0069	00	0069	1F
006A	18	006A	18
006B	00	006B	1F
006C	00	006C	18
006D	00	006D	1E
006E	00	006E	18
006F	00	006F	1D
0070	7E	0070	7E
0071	00	0071	1F
0072	00	0072	7E
0073	00	0073	1E

0074	C0	0074	C0
0075	00	0075	1F
0076	00	0076	C0
0077	00	0077	1E

Esempio 2

K	023
ADD	0064

Prima		Dopo	
ADD	W	ADD	W
0064	33	0064	33
0065	00	0065	1F
0066	00	0066	33
0067	00	0067	1E
0068	00	0068	33
0069	00	0069	1D
006A	00	006A	33
006B	00	006B	1C
006C	00	006C	33
006D	00	006D	1B
006E	00	006E	33
006F	00	006F	1A
0070	00	0070	33
0071	00	0071	19
0072	00	0072	33
0073	00	0073	18
0074	00	0074	33
0075	00	0075	17

0076	00	0076	33
0077	00	0077	16
0078	00	0078	33
0079	00	0079	15
007A	00	007A	33
007B	00	007B	14
007C	00	007C	33
007D	00	007D	13
007E	00	007E	33
007F	00	007F	12
0080	00	0080	33
0081	00	0081	11
0082	00	0082	33
0083	00	0083	10
0084	00	0084	33
0085	00	0085	0F
0086	00	0086	33
0087	00	0087	0E
0088	00	0088	33
0089	00	0089	0D
008A	00	008A	33
008B	00	008B	0C
008C	00	008C	33
008D	00	008D	0B
008E	00	008E	33
008F	00	008F	0A
0090	00	0090	33
0091	00	0091	09
0092	00	0092	33

0093	00	0093	08
0094	00	0094	33
0095	00	0095	07
0096	00	0096	33
0097	00	0097	06
0098	00	0098	33
0099	00	0099	05
009A	00	009A	33
009B	00	009B	04
009C	00	009C	33
009D	00	009D	03
009E	00	009E	33
009F	00	009F	02
00A0	00	00A0	33
00A1	00	00A1	01
00A2	00	00A2	33
00A3	00	00A3	00
00A4	00	00A4	33
00A5	00	00A5	00
00A6	00	00A6	33
00A7	00	00A7	00
00A8	00	00A8	33
00A9	00	00A9	00

Esempio 3

K	021		
ADD	0214		
Prima		Dopo	
ADD	W	ADD	W

0214	00	0214	00
0215	00	0215	00
0216	FF	0216	FF
0217	00	0217	1F
0218	00	0218	FF
0219	00	0219	1E
021A	00	021A	FF
021B	00	021B	1D
021C	89	021C	89
021D	00	021D	1F
021E	00	021E	89
021F	00	021F	1E
0220	00	0220	89
0221	00	0221	1D
0222	00	0222	89
0223	00	0223	1C
0224	00	0224	89
0225	00	0225	1B
0226	00	0226	89
0227	00	0227	1A
0228	00	0228	89
0229	00	0229	19
022A	00	022A	89
022B	00	022B	18
022C	00	022C	89
022D	00	022D	17
022E	6F	022E	6F
022F	00	022F	1F
0230	00	0230	6F

0231	00	0231	1E
0232	00	0232	6F
0233	00	0233	1D
0234	00	0234	6F
0235	00	0235	1C
0236	B5	0236	B5
0237	00	0237	1F
0238	00	0238	B5
0239	00	0239	1E
023A	00	023A	B5
023B	00	023B	1D
023C	00	023C	B5
023D	00	023D	1C
023E	00	023E	B5
023F	00	023F	1B
0240	00	0240	B5
0241	00	0241	1A
0242	F6	0242	F6
0243	00	0243	1F
0244	7C	0244	7C
0245	00	0245	1F
0246	00	0246	7C
0247	00	0247	1E
0248	5D	0248	5D
0249	00	0249	1F
024A	00	024A	5D
024B	00	024B	1E
024C	00	024C	5D
024D	00	024D	1D

024E	00	024E	5D
024F	00	024F	1C
0250	00	0250	5D
0251	00	0251	1B
0252	00	0252	5D
0253	00	0253	1A
0254	00	0254	5D
0255	00	0255	19

Esempio 4

K	00A		
ADD	0013		
Prima		Dopo	
ADD	W	ADD	W
0013	FF	0013	FF
0014	00	0014	1F
0015	00	0015	FF
0016	00	0016	1E
0017	00	0017	FF
0018	00	0018	1D
0019	8F	0019	8F
001A	00	001A	1F
001B	00	001B	8F
001C	00	001C	1E
001D	00	001D	8F
001E	00	001E	1D
001F	00	001F	8F
0020	00	0020	1C
0021	00	0021	8F

0022	00	0022	1B
0023	00	0023	8F
0024	00	0024	1A
0025	00	0025	8F
0026	00	0026	19

Esempio 5

K	010
ADD	03B0

Prima		Dopo	
ADDW	W	ADDW	W
03B0	FF	03B0	FF
03B1	00	03B1	1F
03B2	5B	03B2	5B
03B3	00	03B3	1F
03B4	A1	03B4	A1
03B5	00	03B5	1F
03B6	9B	03B6	9B
03B7	00	03B7	1F
03B8	B2	03B8	B2
03B9	00	03B9	1F
03BA	0B	03BA	0B
03BB	00	03BB	1F
03BC	53	03BC	53
03BD	00	03BD	1F
03BE	1B	03BE	1B
03BF	00	03BF	1F
03C0	39	03C0	39
03C1	00	03C1	1F

03C2	81	03C2	81
03C3	00	03C3	1F
03C4	27	03C4	27
03C5	00	03C5	1F
03C6	F3	03C6	F3
03C7	00	03C7	1F
03C8	9E	03C8	9E
03C9	00	03C9	1F
03CA	AD	03CA	AD
03CB	00	03CB	1F
03CC	86	03CC	86
03CD	00	03CD	1F
03CE	3A	03CE	3A
03CF	00	03CF	1F