

# Progetto di Reti Logiche

Prof. Fornaciari, Prof. Palermo e Prof. Salice  
Anno Accademico 2024 - 2025

## SPECIFICA per lo svolgimento del progetto Revisione del 12 febbraio 2025

### 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.

Il sistema legge, un byte alla volta, un messaggio in memoria costituito da una sequenza di K parole W, il cui valore è compreso tra -128 e +127 rappresentato in complemento a 2, applica un filtro differenziale e salva la sequenza di K parole R in memoria. NOTA: i valori di R al di fuori dell'intervallo -128 : +127 sono saturati al limite definito dall'intervallo (per esempio, se R=-196, R viene sostituito con -128)

Tutti i dati da utilizzare, di partenza, sono memorizzati sequenzialmente a partire da un indirizzo specificato (ADD). I dati iniziali sono i seguenti:

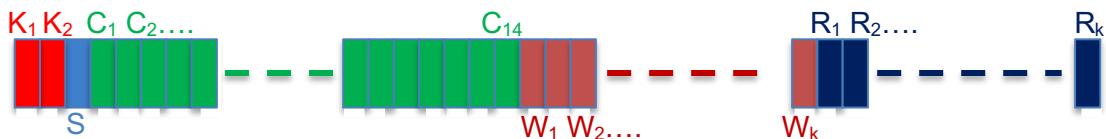
- **17 (2+1+14) byte** che indicano la lunghezza della sequenza, l'ordine del filtro differenziale e i coefficienti. In particolare:
  - **K1** e **K2**, due byte, che indicano la lunghezza della sequenza K dei dati W; K1 è il byte più significativo; **la lunghezza minima è di 7 byte**. La lunghezza del vettore da elaborare è tale da essere sicuri di rimanere all'interno della memoria data all'interno del Test Bench. In pratica, non verrà mai dato un TB non valido, se questo avvenisse un qualsiasi comportamento del modulo da progettare sarebbe considerato valido.
  - **S**, un byte, che indica quale filtro selezionare tra quello di ordine 3 e quello di ordine 5; Il valore consente di selezionare le aree di memoria nelle quali sono caricati i coefficienti. Se il bit meno significativo di S è pari a **0** il filtro da utilizzare è quello di **ordine 3**, se il bit meno significativo di S è pari a **1** il filtro da utilizzare è quello di **ordine 5**.
  - **Da C1 a C14**, quattordici byte, che memorizzano i coefficienti dei due filtri. In particolare:
    - Ordine 3: c=[**0, -1, 8, 0, -8, 1, 0**] normalizzazione n=12
    - Ordine 5: c=[**1, -9, 45, 0, -45, 9, -1**] normalizzazione n=60
    - **ATTENZIONE: il valore dei coefficienti dei filtri può cambiare ad ogni richiesta di elaborazione (altrimenti non sarebbero in memoria).**
- **K byte** da elaborare che contengono valori da -128 a +127. E' il vettore numerico da processare e da riscrivere secondo la specifica del filtro.

Il filtro ha la seguente funzione  $f'(i) = (1/n) * \sum C_j * f[j+i]$  con j che va da -l a +l e con l che vale 2 per il filtro di ordine 3 e 3 per il filtro di ordine 5; n è il valore di normalizzazione. Si osservi

che il risultato del filtraggio prima della normalizzazione ( $\sum C_j * f[j+i]$ ) richiede un numero di bit adeguato per la sua rappresentazione.

Per esempio, per il filtro di ordine 3 la  $f'(x) \approx (1/12)*(-f[i-2]+8*f[i-1]-8*f[i+1]+f[i+2])$ .

La sequenza K del risultato R viene riscritta a partire dall'indirizzo ADD+17+K cioè alla fine dei valori da elaborare.



I valori ad inizio e fine sequenza sono da considerare pari a 0. Per esempio, considerando un filtro di ordine 3, i valori iniziali ADD+17-2, ADD+17-1 non esistono e il filtro utilizza degli 0. Allo stesso modo, a fine sequenza ADD+17+K e ADD+17+K+1 non esistono e il filtro utilizza degli 0.

Per esempio, si consideri la sequenza +101 -86 -12 +41 +87 -123 -127 +87 +108. Utilizzando un filtro di ordine 3, il primo valore ottenuto è  $0^* -1 +0^* 8 +101^* 0 -86^* -8 -12^* 1$  cioè -40.

Ogni valore ottenuto deve essere normalizzato dividendolo per il coefficiente n; per esempio, 60 quando si utilizza il filtro di ordine 5. La divisione deve essere approssimata nel seguente modo:

- $1/12$  è approssimato con  $1/16 + 1/64 + 1/256 + 1/1024$
- $1/60$  è approssimato con  $1/64 + 1/1024$

**Importante:** si osservi che ogni shift a destra equivale a una divisione per due e produce un troncamento verso -1 per numeri negativi e verso 0 per i numeri positivi. Quindi, nel caso di numeri negativi, l'errore si amplifica perché ogni termine viene troncato indipendentemente, come avviene nel caso degli shift multipli, e può fornire un contributo di -1. Per esempio, se il numero (in decimale) è -1, ogni shift destro (per esempio, 1/16, 1/64, 1/256, 1/1024) restituisce -1 e la somma è -4. Per ridurre questo errore, si aggiunge +1 al risultato del singolo shift quando il valore da shiftare è negativo. Riassumendo, se  $val >> m$  è maggiore di zero si aggiunge 0, se  $val >> m$  è minore di zero si aggiunge 1.  $>>n$ : shift destro di n posizioni

Per esempio,  $-1^* 1/12$  è circa uguale a  $-1^*(1/16 + 1/64 + 1/256 + 1/1024) = (-1 >> 4 + 1) + (-1 >> 6 + 1) + (-1 >> 8 + 1) + (-1 >> 10 + 1) = 0$ .

Per l'esempio precedente, invece,  $-40^*(1/12)$  può essere approssimato con  $(-40 >> 4 + 1) + (-40 >> 6 + 1) + (-40 >> 8 + 1) + (-40 >> 10 + 1)$  cioè, in Complemento a 2 1111 1101 1000 si ottiene  $(1111 1111 1101 + 0000 0000 0001) + (1111 1111 1111 1111 + 0000 0000 0001) + (1111 1111 1111 1111 + 0000 0000 0001) + (1111 1111 1111 1111 + 0000 0000 0001) = 1111 1111 1110$ . Il risultato è poi riportato su 8 bit cioè 1111 1110 (-2 decimale al posto del valore corretto -3 ma meglio -5, ottenuto se non applicassimo la correzione).

#### ESEMPIO(valori in decimale)

- Sequenza di partenza

32 -24 -35 0 46 -54 -39 -22 -53 -47 12 11 11 45 -30 -14 -35 -25 -19 -35 -41 -61 -24 -62

- Dati filtrati con filtro di ordine 3

157 536 -178 -678 428 658 -355 119 251 -487 -400 100 -314 303 426 ...

- Sequenza finale

11 43 -13 -54 33 53 -28 8 18 -38 -31 7 -24 23 33

## Funzionamento

Il modulo da implementare ha **3 ingressi primari, uno ad 1 bit (START), uno a 16 bit (ADD) 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** vengono posti il primo indirizzo da elaborare. Il modulo prima di alzare il segnale di **DONE** deve aggiornare i dati in memoria seguendo la descrizione generale del modulo.

## 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);

        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\_ADD è il segnale (vettore) ADD generato dal Test Bench che rappresenta l'indirizzo dal quale parte la sequenza da elaborare (**primo byte del preambolo di 17Byte al vettore numerico da processare**);
- 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.

NOTA! Non verrà mai dato un TB non valido, e.g. che non rispetti il protocollo di start/done, o che non dia il reset all'inizio della sequenza, o che dia un K troppo grande tale per cui si va oltre il limite della memoria. Se questo avvenisse (e non avverrà) un qualsiasi comportamento del modulo da progettare sarebbe considerato valido.

## 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](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 tra la dei valori della sequenza di partenza (indicata come "prima"), cioè durante il periodo nel quale START assume valore 1, il valore ottenuto dal filtraggio (indicata come " $\sum C_i * f[j+i]$ ") e la configurazione finale (indicata come "dopo") nel solo momento utile, cioè durante il periodo nel quale DONE assume valore 1. Il valore R è scritto dopo la sequenza delle parole W. Inizialmente R è 0. K è il numero di parole W della sequenza mentre ADD è l'indirizzo nel quale si trova la prima parola W. I valori sono scritti sia in esadecimale che decimale.

### ESEMPIO 1 Filtro ordine 3

K1	00						
K2	07						
S	02						
<b>ADD 00C3 Nota ADDW = ADD+17 (in decimale)</b>							
<b>Prima</b> $\sum C_i * f[j+i]$ <b>Dopo</b>							
ADDW	W	Wdec	exa	dec	ADDR	R	Rdec
0004	2F	47	0037	55	00CA	03	3
0005	FE	-2	FFF9	-7	00CB	00	0
0006	27	39	018D	397	00CC	1F	31
0007	B9	-71	042B	1067	00CD	57	87
0008	94	-108	013E	318	00CE	18	24
0009	91	-111	FBBF	-1089	00CF	A6	-90
000A	25	37	FCF4	-780	00D0	C1	-63

### ESEMPIO 2 Filtro ordine 5

K1	00						
K2	11						
S	11						
<b>ADD 002C Nota ADDW = ADD+17 (in decimale)</b>							
<b>Prima</b> $\sum C_i * f[j+i]$ <b>Dopo</b>							
ADDW	W	Wdec	exa	dec	ADDR	R	Rdec
003D	26	38	F888	-1912	003D	E2	-30
003E	14	20	1B7A	7034	003E	73	115
003F	38	-104	F5BC	-2628	003F	D5	-43
0040	4C	76	EA7F	-5505	0040	A5	-91
0041	28	40	FB1B	-1253	0041	EC	-20
0042	7E	126	0559	1369	0042	16	22
0043	11	17	FD9E	-610	0043	F7	-9
0044	70	112	1098	4248	0044	46	70
0045	8E	-114	1D0F	7439	0045	7B	123
0046	BA	-70	F65F	-2465	0046	D8	-40
0047	B3	-77	F5E3	-2589	0047	D6	-42
0048	1F	31	E41D	-7139	0048	8B	-117
0049	58	88	0BFC	3068	0049	31	49
004A	E6	-26	1173	4467	004A	49	73
004B	FC	-4	EC42	-5054	004B	AE	-82
004C	54	84	F3B9	-3143	004C	CC	-52
004D	49	73	0ECE	3790	004D	3E	62

### ESEMPIO 3 Filtro ordine 3

K1	00							
K2	20							
S	FE							
<b>ADD 00C3 Nota ADDW = ADD+17 (in decimale)</b>								
Prima		$\sum C_i \cdot f[j+i]$		Dopo				
ADDW	W	Wdec	exa	dec	ADDR	R	Rdec	
00D4	C9	-55		01C0	448	00E3	24	36
00D5	B9	-71		0248	584	00E4	2F	47
00D6	88	-120		FC49	-951	00E5	B4	-76
00D7	40	64		FAA7	-1369	00E6	90	-112
00D8	4A	74		FF05	-251	00E7	EE	-18
00D9	70	112		01F1	497	00E8	27	39
00DA	0D	13		0127	295	00E9	17	23
00DB	49	73		F0CD	-563	00EA	D3	-45
00DC	39	57		05C8	1480	00EB	79	121
00DD	9D	-99		FDB6	-586	00EC	D1	-47
00DE	75	117		FD70	-656	00ED	CC	-52
00DF	0F	-33		05E0	1504	00EE	7B	123
00E0	B9	-71		0200	512	00EF	2A	42
00E1	9D	-99		FA67	-1433	00F0	8B	-117
00E2	65	101		0003	3	00F1	00	0
00E3	A6	-90		03D6	982	00F2	4F	79
00E4	04	4		F8F3	-1805	00F3	80	-128
00E5	6B	107		045A	1114	00F4	5B	91
00E6	80	-128		046B	1131	00F5	5C	92
00E7	E0	-32		FB02	-1278	00F6	99	-103
00E8	17	23		FE06	-506	00F7	D9	-39
00E9	25	37		03B8	952	00F8	4C	76
00EA	AE	-82		FE86	-378	00F9	E3	-29
00EB	50	80		FD7C	-644	00FA	CC	-52
00EC	F5	-11		041B	1051	00FB	56	86
00ED	D9	-39		FEB4	-332	00FC	E6	-26
00EE	11	17		FFF1	-15	00FD	00	0
00EF	E4	-28		FEAE	-338	00FE	E5	-27
00F0	3E	62		FFE2	-30	00FF	FF	-1
00F1	EF	-17		FF70	-144	0100	F6	-10
00F2	4B	75		015A	346	0101	1B	27
00F3	BC	-68		0269	617	0102	31	49

#### ESEMPIO 4 Filtro ordine 5

K1	00						
K2	2F						
S	7F						
ADD	01A6	<b>Nota ADDW = ADD+17 (in decimale)</b>					
Prima		$\sum C_i \cdot f[j+i]$		Dopo			
ADDW	W	Wdec	exa	dec	ADDR	R	Rdec
01B7	8D	-115	F1D0	-3632	01D5	C5	-59
01B8	3C	60	FEA5	-347	01D6	FB	-5
01B9	9A	-102	082A	2090	01D7	22	34
01BA	0E	14	FFB6	-74	01D8	FF	-1
01BB	90	-112	0323	803	01D9	0C	12
01BC	07	7	F376	-3210	01DA	CB	-53
01BD	BC	-68	16D3	5843	01DB	60	96
01BE	86	-122	086D	2157	01DC	23	35
01BF	A0	-96	088E	-10098	01DD	80	-128
01C0	78	120	EE9D	-4451	01DE	B7	-73
01C1	08	8	2599	9625	01DF	7F	127
01C2	9B	-101	1299	4761	01E0	4E	78
01C3	94	-108	E211	-7663	01E1	82	-126
01C4	5A	90	DE56	-8618	01E2	80	-128
01C5	5D	93	1E54	7764	01E3	7F	127
01C6	D1	-47	FCF1	-783	01E4	F4	-12
01C7	43	67	078A	1930	01E5	1F	31
01C8	83	-125	1EC4	7876	01E6	7F	127
01C9	A6	-90	E07A	-8070	01E7	80	-128
01CA	17	23	0385	901	01E8	0E	14
01CB	A8	-88	091B	2331	01E9	26	38
01CC	E0	-32	00AE	174	01EA	02	2
01CD	A9	-87	F1ED	-3603	01EB	C5	-59
01CE	2E	46	02BE	702	01EC	0A	10
01CF	9A	-102	0FFF	4095	01ED	42	66
01D0	F6	-10	DB13	-9453	01EE	80	-128
01D1	53	83	0DA2	3490	01EF	39	57
01D2	B3	-77	16B2	5810	01F0	5F	95
01D3	BE	-66	FF4B	-181	01F1	FE	-2
01D4	97	-105	06CC	1740	01F2	1C	28
01D5	C2	-62	D893	-10093	01F3	80	-128
01D6	7E	126	FE7A	-390	01F4	FA	-6
01D7	F6	-10	03D8	384	01F5	0F	15
01D8	7D	125	EFC0	-4160	01F6	BC	-68
01D9	36	54	1A01	6657	01F7	6E	110
01DA	00	0	F47D	-2947	01F8	D0	-48
01DB	54	84	0631	1585	01F9	19	25
01DC	C4	-60	1E76	7798	01FA	7F	127
01DD	B2	-78	E9EF	-5649	01FB	A3	-93
01DE	34	52	F198	-3688	01FC	C4	-60
01DF	0E	14	0F1F	3871	01FD	3F	63
01E0	EB	-21	002A	42	01FE	00	0
01E1	EF	-17	0B28	2856	01FF	2E	46
01E2	9F	-97	0604	1540	0200	19	25
01E3	BE	-66	FF5B	-165	0201	FE	-2
01E4	98	-104	03B2	946	0202	0E	14
01E5	BC	-68	EFA9	-4183	0203	BB	-69
		-		--			--

### ESEMPIO 5 Filtro ordine 3

K1	00						
K2	5A						
S	FE						
<b>ADD</b>	<b>00C3</b>	<b>Nota ADDW = ADD+17 (in decimale)</b>					
<b>Prima</b>		<b><math>\sum C_i \cdot f[i+i]</math></b>			<b>Dopo</b>		
ADDW	W	Wdec	exa	dec	ADDR	R	Rdec
00D4	FC	-4	FED4	-300	011D	E9	-23
00D5	16	22	0417	1047	011E	56	86
00D6	84	-124	FE58	-424	011F	DF	-33
00D7	57	87	F988	-1656	0120	80	-128
00D8	5C	92	008B	139	0121	0A	10
00D9	5E	94	002D	45	0122	02	2
00DA	47	71	03C2	362	0123	4E	78
00DB	DC	-36	01D8	472	0124	25	37
00DC	0E	14	FB7F	-1153	0125	A1	-95
00DD	6E	110	FDB8	-584	0126	01	-47
00DE	56	86	0500	1280	0127	6A	106
00DF	04	-44	00D1	209	0128	10	16
00E0	3E	62	FA3B	-1477	0129	87	-121
00E1	7F	127	0298	664	012A	35	53
00E2	E9	-23	0549	1353	012B	6F	111
00E3	C4	-60	013D	317	012C	18	24
00E4	AF	-81	FED1	-303	012D	E9	-23
00E5	EC	-20	FE57	-425	012E	DF	-33
00E6	FA	-6	FBF8	-1032	012F	AB	-85
00E7	73	115	0096	150	0130	0B	11
00E8	DF	-33	06B7	1719	0131	7F	127
00E9	AA	-86	FB81	-1151	0132	A3	-93
00EA	69	105	FB0F	-1265	0133	99	-103
00EB	44	68	0545	1349	0134	6F	111
00EC	BE	-66	0546	1350	0135	6F	111
00ED	97	-105	FB05	-1067	0136	A9	-87
00EE	47	71	FA0F	-1521	0137	84	-124
00EF	61	97	014F	335	0138	1A	26
00F0	1D	29	0611	1553	0139	7F	127
00F1	36	-106	008C	140	013A	0A	10
00F2	00	0	FC0B	-1013	013B	AF	-81
00F3	05	5	032A	810	013C	41	65
00F4	A0	-96	01B6	438	013D	22	34
00F5	C0	-64	0085	133	013E	0A	10
00F6	8E	-114	FEE8	-280	013F	EA	-22
00F7	FA	-6	FA5F	-1441	0140	8A	-118
00F8	58	88	FD23	-733	0141	C6	-58
00F9	6F	111	0008	8	0142	00	0
00FA	59	89	02DA	730	0143	3A	58
00FB	0A	10	01D7	471	0144	25	37
00FC	0A	10	014C	332	0145	1A	26
00FD	CE	-50	01E2	482	0146	26	38
00FE	C5	-59	00A8	168	0147	0C	12
00FF	C4	-60	FAD3	-1325	0148	94	-108
0100	62	98	0283	643	0149	34	52
0101	89	-119	002E	46	014A	02	2
0102	70	112	F8B5	-1867	014B	80	-128
0103	C2	99	04D3	122F	014C	FF	101

0103	62	98	04D3	1235	014C	65	101
0104	DF	-33	0385	901	014D	49	73
0105	D4	-44	0225	549	014E	2C	44
0106	85	-123	0138	312	014F	18	24
0107	B7	-73	FAAC	-1364	0150	90	-112
0108	2F	47	FFD5	-43	0151	FE	-2
0109	D0	-48	0086	134	0152	0A	10
010A	22	34	FFBA	-70	0153	FB	-5
010B	D5	-43	009C	156	0154	0B	11
010C	11	17	FF31	-207	0155	F1	-15
010D	E4	-28	01F2	498	0156	27	39
010E	CB	-53	01F4	500	0157	27	39
010F	37	-105	01EA	490	0158	26	38
0110	9D	-93	F9D5	-1573	0159	80	-128
0111	5E	94	FEC8	-312	015A	E8	-24
0112	D8	-40	01EC	492	015B	26	38
0113	37	55	FB88	-1144	015C	A3	-93
0114	51	81	04A2	1186	015D	61	97
0115	AE	-82	0117	279	015E	16	22
0116	32	50	FA02	-1534	015F	84	-124
0117	56	86	0568	1384	0160	71	113
0118	33	-109	0158	344	0161	1B	27
0119	1E	30	FDB9	-583	0162	D1	-47
011A	CA	-54	0322	802	0163	41	65
011B	C7	-57	FE1D	-483	0164	DA	-38
011C	FD	-3	0006	6	0165	00	0
011D	D3	-45	FF1E	-226	0166	EF	-17
011E	30	48	FB22	-1246	0167	9B	-101
011F	7D	125	FE1E	-482	0168	DA	-38
0120	6F	111	0456	1110	0169	5B	31
0121	E9	-23	0368	872	016A	46	70
0122	E6	-26	01E0	480	016B	26	38
0123	9D	-93	FF72	-142	016C	F6	-10
0124	EF	-17	FFB4	-76	016D	FB	-5
0125	A3	-93	018B	395	016E	1F	31
0126	CA	-54	FCAD	-851	016F	BB	-63
0127	00	0	0308	776	0170	3F	63
0128	84	-124	FC0B	-1013	0171	AF	-81
0129	7B	123	FE6E	-402	0172	E0	-32
012A	AD	-83	0682	1666	0173	7F	127
012B	B6	-74	FE1B	-485	0174	DA	-38
012C	DE	-34	FD13	-749	0175	C5	-59
012D	1E	30	FF3A	-198	0176	F1	-15

## ESEMPIO 6 Filtro ordine 5

K1	00						
K2	81						
S	81						
<b>ADD 01AD Nota ADDW = ADD+17 (in decimale)</b>							
Prima		$\Sigma C_i \cdot f[i+i]$	Dopo				
ADDW	W	Wdec	exa	dec	ADDR	R	Rdec
01BE	59	89	0A5F	2655	022E	2B	43
01BF	AC	-84	2530	9520	022F	7F	127
01C0	83	-125	F204	-3580	0230	C6	-58
01C1	00	0	DA09	-9719	0231	80	-128
01C2	6E	110	0296	662	0232	0A	10
01C3	F5	-11	2268	8808	0233	7F	127
01C4	92	-110	0BF5	3061	0234	31	49
01C5	8C	-116	FA46	-1466	0235	E9	-23
01C6	B3	-77	F32D	-3283	0236	CA	-54
01C7	ED	-19	F4CA	-2870	0237	D2	-46
01C8	13	19	F629	-2519	0238	D7	-41
01C9	44	68	F423	-3037	0239	CF	-49
01CA	60	96	05A1	1441	023A	17	23
01CB	2D	45	0327	807	023B	0C	12
01CC	47	71	011E	286	023C	04	4
01CD	2B	43	F7B9	-2119	023D	DD	-35
01CE	70	112	05CD	1485	023E	18	24
01CF	10	16	0343	841	023F	0D	13
01D0	60	103	EBC9	-5175	0240	AB	-85
01D1	75	117	0BB1	2993	0241	30	48
01D2	1F	31	1A2C	6700	0242	6E	110
01D3	C8	-56	0382	898	0243	0E	14
01D4	ED	-19	FD32	-718	0244	F5	-11
01D5	CD	-51	02E8	744	0245	0B	11
01D6	CD	-51	10C1	4289	0246	47	71
01D7	80	-128	EAFD	-5379	0247	A7	-89
01D8	4F	79	EA60	-5536	0248	A5	-91
01D9	02	2	13D5	5077	0249	53	83
01DA	F3	-13	FF69	-151	024A	FE	-2
01DB	E9	-23	075A	1882	024B	1E	30
01DC	D5	-43	F276	-3466	024C	C7	-57
01DD	32	50	00E2	226	024D	03	3
01DE	EB	-21	F50F	-2593	024E	D6	-42
01DF	6F	111	FE6C	-404	024F	FA	-6
01E0	E9	-23	1535	5429	0250	59	89
01E1	05	5	EFF8	-4104	0251	BC	-68
01E2	2E	46	00E2	226	0252	03	3
01E3	FA	-6	117E	4478	0253	49	73
01E4	B5	-75	10F5	4341	0254	47	71
01E5	A8	-88	DCFE	-8962	0255	80	-128
01E6	6B	107	03CA	970	0256	0F	15
01E7	9F	-97	199D	6557	0257	6C	108
01E8	FF	-1	D55F	-10913	0258	80	-128
01E9	5F	95	1D30	7472	0259	7B	123
01EA	83	-125	FE32	-462	025A	F9	-7
01EB	61	97	EB58	-5288	025B	A9	-87
01EC	D9	-39	1FF7	8183	025C	7F	127
01ED	80	67	FF1A	496	025D	59	7

01E0	09	-39	11F7	8183	025C	1F	12F
01ED	BD	-67	FE1A	-486	025D	F9	-7
01EE	D7	-41	ED7C	-4740	025E	B2	-78
01EF	30	48	FCA0	-864	025F	F3	-13
01F0	0A	10	F987	-1657	0260	E6	-26
01F1	67	103	F62A	-2518	0261	D7	-41
01F2	3C	60	0E01	3585	0262	3B	59
01F3	18	24	060C	1548	0263	19	25
01F4	0B	11	FCE9	-791	0264	F4	-12
01F5	1B	27	069E	1694	0265	1B	27
01F6	EF	-17	F911	-1775	0266	E4	-28
01F7	42	66	FA6E	-1426	0267	E9	-23
01F8	04	4	12AF	4783	0268	4E	78
01F9	ED	-19	EDAO	-4704	0269	B3	-77
01FA	4E	78	0CBD	3261	026A	35	53
01FB	B4	-76	01EC	492	026B	07	7
01FC	4F	79	E971	-5775	026C	A1	-95
01FD	20	45	08C3	2243	026D	25	37
01FE	2B	43	093F	2367	026E	26	38
01FF	FC	-4	F3DE	-3106	026F	CD	-51
0200	67	103	F05B	-677	0270	F6	-10
0201	EA	-22	29E6	10726	0271	7F	127
0202	8A	-118	EBF7	-5129	0272	AB	-85
0203	53	83	E03E	-8130	0273	80	-128
0204	20	45	285F	10335	0274	7F	127
0205	86	-122	0163	355	0275	05	5
0206	FE	-2	FC16	-1002	0276	F1	-15
0207	A8	-88	F52B	-2773	0277	D3	-45
0208	67	103	DE2F	-8657	0278	80	-128
0209	4D	77	2C4C	11340	0279	7F	127
020A	8C	-116	FCB0	-848	027A	F3	-13
020B	54	84	E0EF	-7953	027B	80	-128
020C	3D	61	0741	1857	027C	1E	30
020D	43	67	08E4	2276	027D	25	37
020E	01	1	FF9B	-101	027E	FF	-1
020F	26	38	1042	4162	027F	45	69
0210	36	-106	092C	2348	0280	26	38
0211	F3	-13	EB8E	-5234	0281	AA	-86
0212	FA	-6	082A	2090	0282	22	34
0213	CF	-49	0A3F	2623	0283	2A	42
0214	D0	-48	EA56	-5546	0284	A5	-91
0215	5A	90	EE77	-4489	0285	B6	-74
0216	58	88	F8B7	-1865	0286	E2	-30
0217	7C	124	1A94	6804	0287	70	112
0218	B0	-80	0F20	3872	0288	3F	63
0219	FE	-2	03DC	988	0289	0F	15
021A	88	-120	FD2F	-721	028A	F5	-11
021B	10	16	FAB4	-1356	028B	EA	-22
021C	B8	-72	F779	-2183	028C	DC	-36
021D	6B	107	E2DE	-7458	028D	85	-123
021E	61	97	0C12	3090	028E	33	51
021F	27	39	1AB0	6832	028F	70	112
0220	C4	-60	F52B	-2773	0290	D3	-45
0221	5E	94	EA47	-5561	0291	A5	-91
0222	2C	44	1EDF	7903	0292	7F	127
0223	BB	-72	0607	1751	0293	1C	28

0222	4L	44	ICUR	ICUS	0234	rr	121
0223	B8	-72	0607	1751	0293	1C	28
0224	EB	-21	F5A1	-2655	0294	D5	-43
0225	D5	-43	12F8	4856	0295	4F	79
0226	89	-119	FECB	-309	0296	FC	-4
0227	E9	-23	E1D1	-7727	0297	81	-127
0228	27	39	10E7	4327	0298	47	71
0229	85	-123	1F82	8066	0299	7F	127
022A	84	-124	0DF6	-8714	029A	80	-128
022B	55	85	DC27	-9177	029B	80	-128
022C	7A	122	0339	825	029C	0C	12
022D	4E	78	1CFA	7418	029D	7A	122
022E	D3	-45	F8EE	-1810	029E	E3	-29
022F	4E	78	051D	1309	029F	15	21
0230	A6	-90	12A4	4772	02A0	4E	78
0231	F3	-13	E976	-5770	02A1	A1	-35
0232	13	19	01B4	436	02A2	06	6
0233	E9	-23	16CE	5838	02A3	60	96
0234	A7	-89	E7B7	-6217	02A4	99	-103
0235	65	101	FAC5	-1339	02A5	EB	-21
0236	D6	-42	0BF7	3063	02A6	31	49
0237	4B	75	DE9B	-8549	02A7	80	-128
0238	7D	125	1204	4612	02A8	4C	76
0239	0A	10	FA35	-1483	02A9	E8	-24
023A	7E	126	0CA6	3238	02AA	35	53
023B	B6	-74	0C3F	3135	02AB	33	51
023C	37	55	F01D	-4067	02AC	BE	-66
023D	F3	-13	10A1	4257	02AD	46	70
023E	EA	-22	FB7E	-1154	02AE	ED	-19