UNIVERSITÀ DEGLI STUDI DI
NAPOLI FEDERICO II

Scuola Politecnica e delle Scienze di Base
Corso di Laurea in Ingegneria Informatica

Elaborato in **Architettura dei Sistemi Digitali**

Elaborato finale

Anno Accademico 2024/25

Studenti
Filomena Vigliotti
matr. M63001734
Ciro Scognamilgio
matr. M6300
Antonio Sirignano
matr. M63001732

Indice

1 Esercizio 1 **1**

1.1 Multiplexer 16:1 1

1.1.1 Progetto e architettura 3

1.1.2 Implementazione 8

1.1.3 Simulazione 12

1.2 Rete di interconnessione a 16 ingressi e 4 uscite 15

1.2.1 Progettazione 16

1.2.2 Implementazione 20

1.2.3 Simulazione 20

1.3 Implementazione su board del punto precedente 20

2 Esercizio 2 - Sistema ROM+M **21**

2.1 Progettazione 22

2.2 Implementazione 22

2.3 Simulazione 26

3 Title **30**

3.1 Section 30

4 Title **34**

4.1 Section 34

5 Conclusion **38**

Bibliografia **39**

Chapter 1

Esercizio 1

1.1 Multiplexer 16:1

Un multiplexer è una **macchina combinatoria**, ovvero una macchina la cui uscita in un determinato istante di tempo dipende solo dall'ingresso nel medesimo istante, e quindi realizza una funzione del tipo:

$$U = f(I)$$

dove I e U rappresentano rispettivamente gli insiemi limitati dei valori di ingresso e di uscita.
Il Multiplexer realizza una connessione *n:1*, ovvero connette n sorgenti a un'unica destinazione sulla base di segnali di selezione.
Un **Multiplexer lineare** è composto da n segnali in ingresso e n segnali di selezione. Tale dispositivo convoglia uno specifico segnale in ingresso verso l'uscita solo se il corrispondente segnale di selezione è

alto. Uno svantaggio di un dispositivo di questo tipo è il numero eccessivo di fili per i segnali di selezione. Per risolvere ciò si può aggiungere un **Decoder**, un altro dispositivo notevole, che riceve in ingresso una parola codice di n bit e presenta in uscita la sua rappresentazione decodificata di 2^n bit.

Unendo un Multiplexer lineare a un Decoder, l'architettura diventa quella in figura, e si ottiene un componente definito **Multiplexer indirizzabile**, che diversamente da quello lineare, prende solo 2 segnali di selezione in ingresso. Un MUX indirizzabile è a sua volta una macchina notevole, caratterizzata da 2^n ingressi, n ingressi di selezione e un'unica uscita.

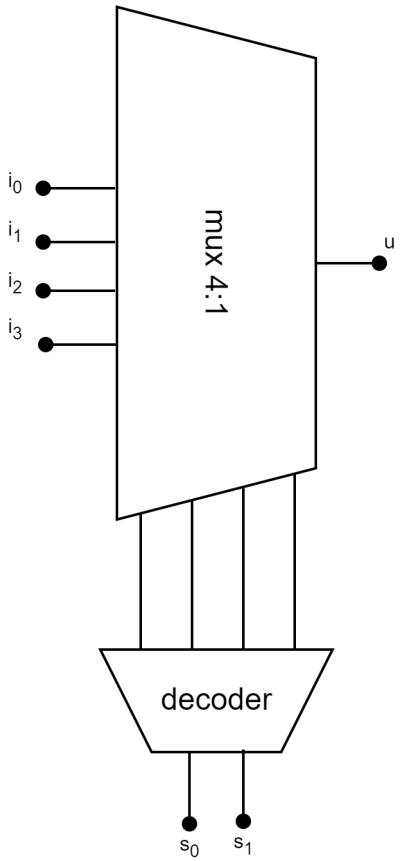

Figure 1.1: multiplexer indirizzabile

Si vuole ora progettare un multiplexer indirizzabile 16:1, utilizzando un approccio per composizione, a partire da multiplexer 4:1. Tale multiplexer è rappresentato di seguito.

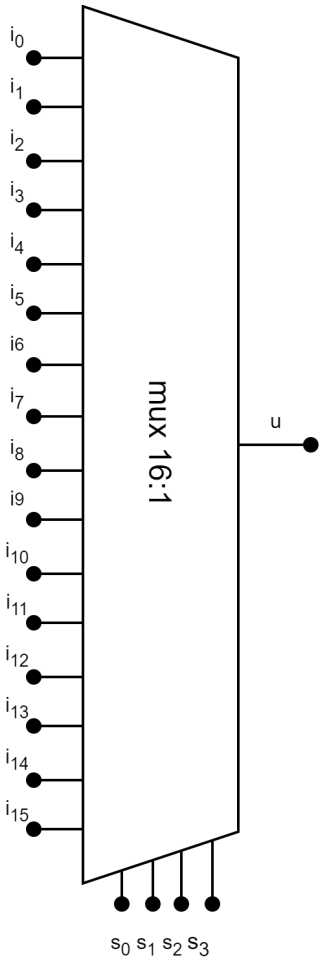

Figure 1.2: multiplexer 16:1

1.1.1 Progetto e architettura

Dapprima si utilizza un approccio per composizione per realizzare un multiplexer 4:1 con multiplexer 2:1.

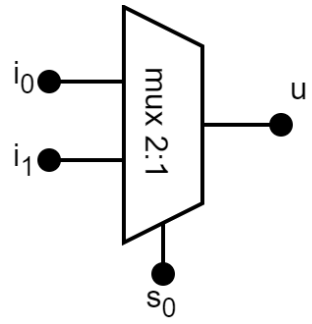

Figure 1.3: multiplexer 2:1

Il primo componente che si realizza è un multiplexer 2:1, caratterizzato dalla seguente tabella di verità:

$\mathbf{s_0}$	$\mathbf{i_1}$	$\mathbf{i_0}$	$\mathbf{u}$
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Table 1.1: Tabella di verità di un Mux 2:1

da cui si ottiene l'equazione:

$$u = (i_0 \text{ AND } \bar{s_0}) \text{ OR } (i_1 \text{ AND } s_0)$$

Il successivo componente da costruire è un multiplexer 4:1.

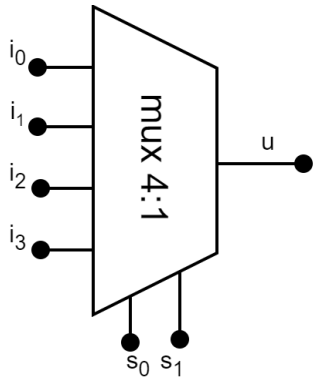

Figure 1.4: multiplexer 4:1

Per composizione, a partire da 3 multiplexer 2:1, si può ottenere un multiplexer 4:1

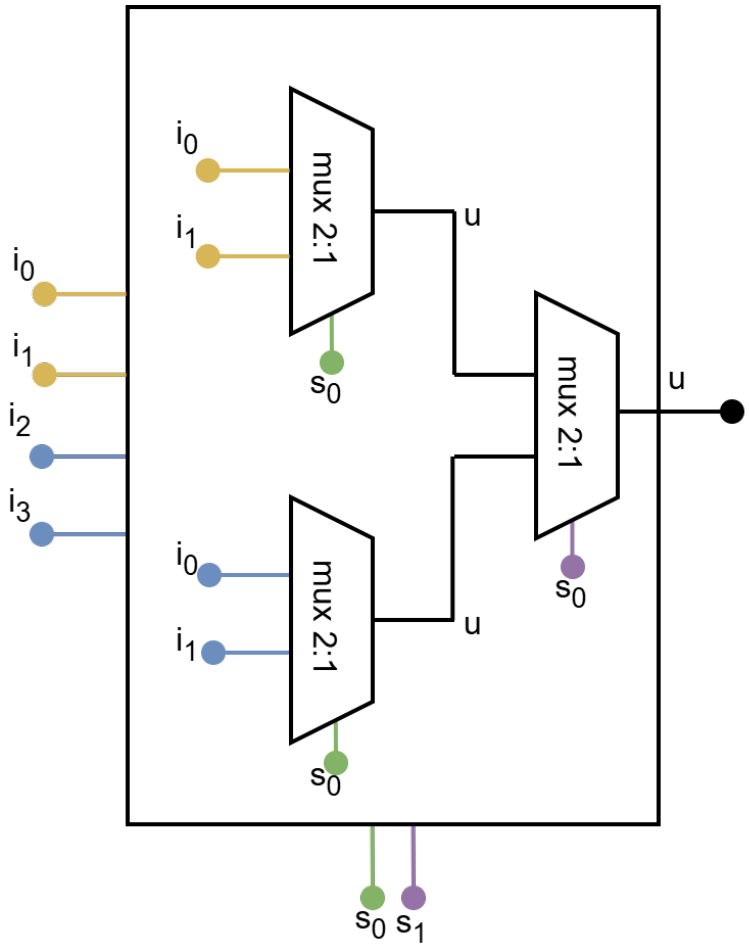

Figure 1.5: multiplexer 4:1 per composizione di multiplexer 2:1

I 4 ingressi entrano in due multiplexer 2:1, che prendono due ingressi e producono un'uscita ciascuno; tali uscite vengono immesse nel terzo multiplexer, che produrrà l'unico output finale. Concettualmente, si divide la selezione in ingresso al multiplexer esterno in due parti:

- la parte meno significativa (indicata dal colore verde) s_0, viene posta in ingresso ai multiplexer del primo stadio e seleziona per ciascuno un filo in uscita;

- la parte più significativa (indicata dal colore viola) s_1 entra nel multiplexer del secondo stadio e decide quale dei due fili, provenienti dai due blocchi precedenti, sarà immessa in uscita.

In maniera analoga si procede con la progettazione del multiplexer 16:1.
Anche in questo caso, sono stati usati dei colori per identificare i collegamenti tra le componenti.

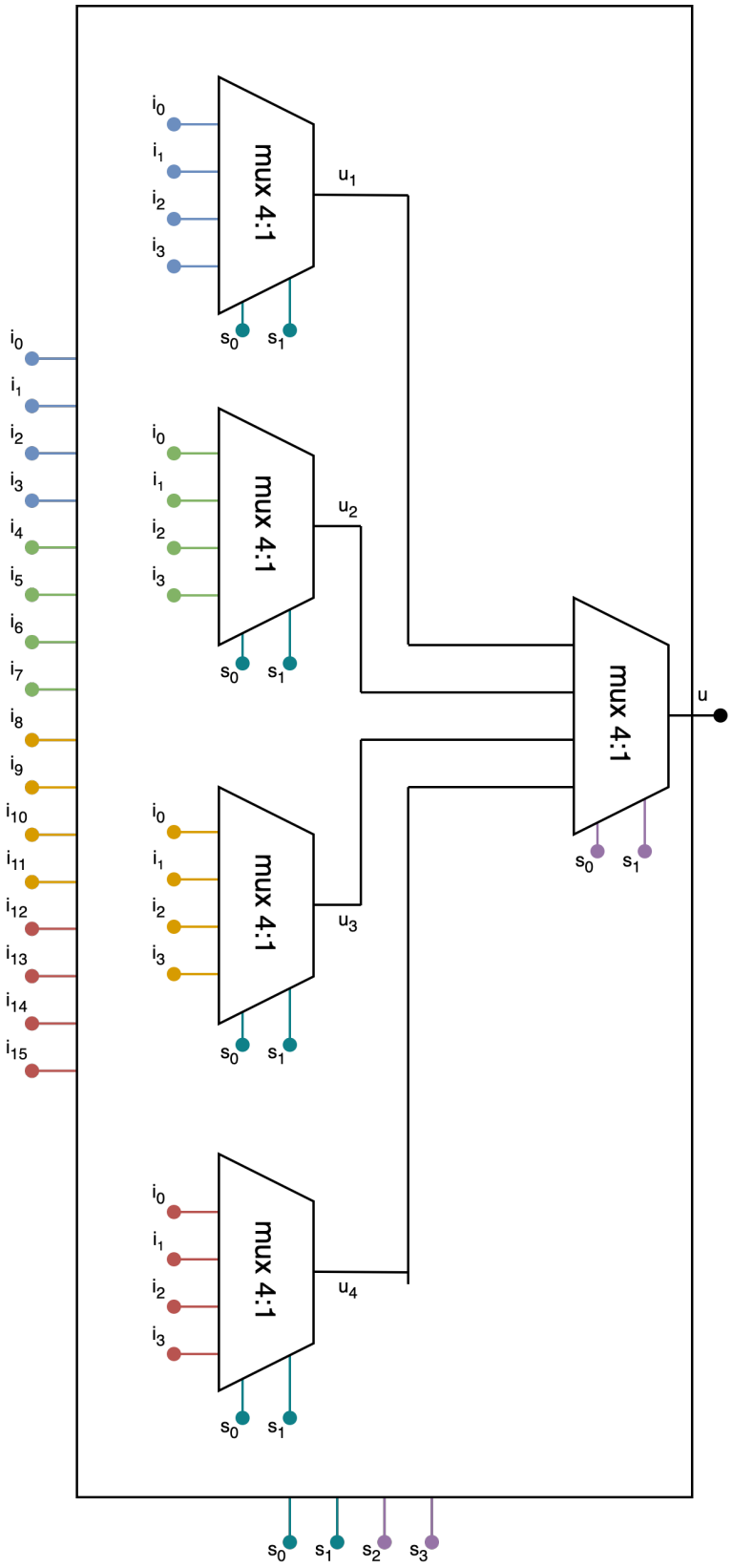

Figure 1.6: multiplexer 16:1 per composizione di multiplexer 4:1

1.1.2 Implementazione

Per l'implementazione si procede con un approccio di tipo strutturale, iniziando quindi dalla codifica del multiplexer 2:1, e, a partire da questo si compongnono dispositivi sempre più complessi fino ad arrivare all'obiettivo del multiplexer 16:1.

Mux 2:1 Di seguito il codice riguardante il Mux 2:1.

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.all;
3
4  entity mux_21 is
5      port
6      (
7          i0, i1: in std_logic;
8          s0: in std_logic;
9          u: out std_logic
10     );
11 end mux_21;
12
13 architecture rtl of mux_21 is
14 begin
15     u <= i0 when s0='0' else
16         i1 when s0='1' else
17         '-';
18 end rtl;
```

Code 1.1: Multiplexer 2:1 in VHDL

L'interfaccia del componente ha come ingressi `i0` ed `i1`, come selezione `s0` e come uscita `u`.

Al seguito della definizione dell'interfaccia, si definisce il comportamento dell'entità, che risponde alla tabella della verità 1.1.

Mux 4:1 Si prosegue con il Mux 4:1.

Come anticipato, viene costruito a partire da tre mux 2:1.

```
1  library IEEE;
2  use IEEE.std_logic_1164.all;
3
4  entity mux_41 is
5      port
6      (
7          i: in std_logic_vector(0 to 3);
8          s: in std_logic_vector(1 downto 0);
9          u: out std_logic
10     );
11 end mux_41;
12
13 architecture structural of mux_41 is
14     signal u_mid: std_logic_vector(0 to 1);
15
16     component mux_21 is
17         port (
18                 i0, i1: in std_logic;
19                 s0: in std_logic;
20                 u: out std_logic
21             );
22     end component;
23
24     begin
25         mux0to1: FOR k IN 0 TO 1 GENERATE
26             m: mux_21
27             port map
28             (
29                 i(k*2),
30                 i(k*2 + 1),
31                 s(0),
32                 u_mid(k)
33             );
34         end GENERATE;
35
36         mux_2: mux_21
37             port map
38             (
39                 u_mid(0),
40                 u_mid(1),
41                 s(1),
```

```
42                  u
43              );
44
45  end structural;
46
```

Code 1.2: Multiplexer 4:1 in VHDL

In quest'entità, l'interfaccia è dichiarata come segue:

- Il parametro `i` vettore di 4 elementi, ognuno corrispondente ad un ingresso del mux 4:1.
- Il parametro `s` vettore di 2 elementi, ognuno corrispondente ad un ingresso di selezione.
- Il parametro `u` corrispondente all'uscita del multiplexer.

A seguire si definisce la struttura del mux 4:1, utilizzando mux 2:1 come componenti.
Con il ciclo `for`, vengono stanziati i primi due mux 2:1, i quali riceveranno in ingresso rispettivamente, gli ingressi del mux 4:1 e la loro uscita è il vettore d'appoggio `u_mid`, il quale è talvolta l'ingresso del terzo mux 2:1.

Mux 16:1 In maniera analoga si procede con la costruzione del mux 16:1. Il codice è il seguente:

```
1  library IEEE;
2  use IEEE.std_logic_1164.all;
3
4  entity mux_161 is
```

```
5       port
6       (
7           i: in std_logic_vector(0 to 15);
8           s: in std_logic_vector(3 downto 0);
9           u: out std_logic
10      );
11  end mux_161;
12
13  architecture structural of mux_161 is
14
15      signal u_mid: std_logic_vector(0 to 3);
16
17      component mux_41 is
18          port
19          (
20              i: in std_logic_vector(0 to 3);
21              s: in std_logic_vector(0 to 1);
22              u: out std_logic
23          );
24      end component;
25
26      begin
27          mux0to3: FOR k IN 0 TO 3 GENERATE
28              m: mux_41
29              port map
30              (
31                  i => i((k*4) to (k*4 + 3)),
32                  s => s(1 downto 0),
33                  u => u_mid(k)
34              );
35          end GENERATE;
36
37          mux_2: mux_41
38              port map
39              (
40                  i => u_mid,
41                  s => s(3 downto 2),
42                  u => u
43              );
44
45  end structural;
```

Code 1.3: Multiplexer 16:1 in VHDL

1.1.3 Simulazione

Per la simulazione, vi è la necessità di un testbench, il quale generiamo in maniera automatica tramite software appositi.

In tale progetto la generazione viene effettuata tramite ChatGPT ed il codice è il seguente:

```
1  library IEEE;
2  use IEEE.std_logic_1164.all;
3  use IEEE.numeric_std.all; -- Libreria necessaria per `to_unsigned`
4
5  entity tb_mux_161 is
6  end tb_mux_161;
7
8  architecture behavior of tb_mux_161 is
9      -- Component declaration
10     component mux_161
11         port (
12             i: in std_logic_vector(0 to 15);
13             s: in std_logic_vector(3 downto 0);
14             u: out std_logic
15         );
16     end component;
17
18     -- Signals for testing
19     signal i: std_logic_vector(0 to 15);
20     signal s: std_logic_vector(3 downto 0);
21     signal u: std_logic;
22
23 begin
24     -- Instantiate the unit under test (UUT)
25     uut: mux_161
26         port map (
27             i => i,
28             s => s,
29             u => u
30         );
31
32     -- Test process
33     stim_proc: process
34         variable expected_output: std_logic; -- Variabile per il
            ↪ controllo
```

```
35      begin
36          -- Initialize inputs
37          i <= (others => '0');
38          s <= "0000";
39          wait for 10 ns;
40
41          -- Apply test cases
42          for sel in 0 to 15 loop
43              -- Set the ith bit of i to '1'
44              i <= (others => '0');
45              i(sel) <= '1';
46
47              -- Set the selector
48              s <= std_logic_vector(to_unsigned(sel, 4));
49
50              -- Aspetta che l'uscita si stabilizzi
51              wait for 10 ns;
52
53              -- Calcola l'uscita attesa
54              expected_output := i(sel);
55
56              -- Controlla se l'uscita è corretta
57              if u = expected_output then
58                  report "Test passed for s = " & integer'image(sel) &
59                         ", u = " & std_logic'image(u);
60              else
61                  report "Test failed for s = " & integer'image(sel) &
62                         ": expected = " &
                           ↪  std_logic'image(expected_output) &
63                         ", got = " & std_logic'image(u)
64                         severity error;
65              end if;
66          end loop;
67
68          -- Fine simulazione
69          report "All tests completed";
70          wait;
71      end process;
72
73  end behavior;
```

Code 1.4: Testbench multiplexer 16:1 in VHDL

Una volta generato ciò, utilizzando i software GHDL e GTKWAVE,

vengono eseguiti i seguenti comandi:

```
[antoniosirignano@Antonios-MacBook-Pro Problema_1 % ghdl -a mux_2_1.vhdl
[antoniosirignano@Antonios-MacBook-Pro Problema_1 % ghdl -a mux_4_1.vhdl
[antoniosirignano@Antonios-MacBook-Pro Problema_1 % ghdl -a mux_16_1.vhdl
[antoniosirignano@Antonios-MacBook-Pro Problema_1 % ghdl -a tb_mux_16_1.vhdl
[antoniosirignano@Antonios-MacBook-Pro Problema_1 % ghdl -e tb_mux_161
 antoniosirignano@Antonios-MacBook-Pro Problema_1 % ghdl -r tb_mux_161 --wave=mux_16_1.ghw

tb_mux_16_1.vhdl:58:17:@20ns:(report note): Test passed for s = 0, u = '1'
tb_mux_16_1.vhdl:58:17:@30ns:(report note): Test passed for s = 1, u = '1'
tb_mux_16_1.vhdl:58:17:@40ns:(report note): Test passed for s = 2, u = '1'
tb_mux_16_1.vhdl:58:17:@50ns:(report note): Test passed for s = 3, u = '1'
tb_mux_16_1.vhdl:58:17:@60ns:(report note): Test passed for s = 4, u = '1'
tb_mux_16_1.vhdl:58:17:@70ns:(report note): Test passed for s = 5, u = '1'
tb_mux_16_1.vhdl:58:17:@80ns:(report note): Test passed for s = 6, u = '1'
tb_mux_16_1.vhdl:58:17:@90ns:(report note): Test passed for s = 7, u = '1'
tb_mux_16_1.vhdl:58:17:@100ns:(report note): Test passed for s = 8, u = '1'
tb_mux_16_1.vhdl:58:17:@110ns:(report note): Test passed for s = 9, u = '1'
tb_mux_16_1.vhdl:58:17:@120ns:(report note): Test passed for s = 10, u = '1'
tb_mux_16_1.vhdl:58:17:@130ns:(report note): Test passed for s = 11, u = '1'
tb_mux_16_1.vhdl:58:17:@140ns:(report note): Test passed for s = 12, u = '1'
tb_mux_16_1.vhdl:58:17:@150ns:(report note): Test passed for s = 13, u = '1'
tb_mux_16_1.vhdl:58:17:@160ns:(report note): Test passed for s = 14, u = '1'
tb_mux_16_1.vhdl:58:17:@170ns:(report note): Test passed for s = 15, u = '1'
tb_mux_16_1.vhdl:69:9:@170ns:(report note): All tests completed
[antoniosirignano@Antonios-MacBook-Pro Problema_1 % gtkwave mux_16_1.ghw

GTKWave Analyzer v3.4.0 (w)1999-2022 BSI

[0] start time.
[170000000] end time.
2024-11-25 18:54:06.970 gtkwave[78165:3049537] +[IMKClient subclass]: chose IMKClient_Modern
2024-11-25 18:54:06.970 gtkwave[78165:3049537] +[IMKInputSession subclass]: chose IMKInputSession_Modern
```

Figure 1.7: Comandi per la simulazione

Con l'esecuzione dell'ultimo comando, vi si apre una nuova finestra che permette la visualizzazione delle onde:

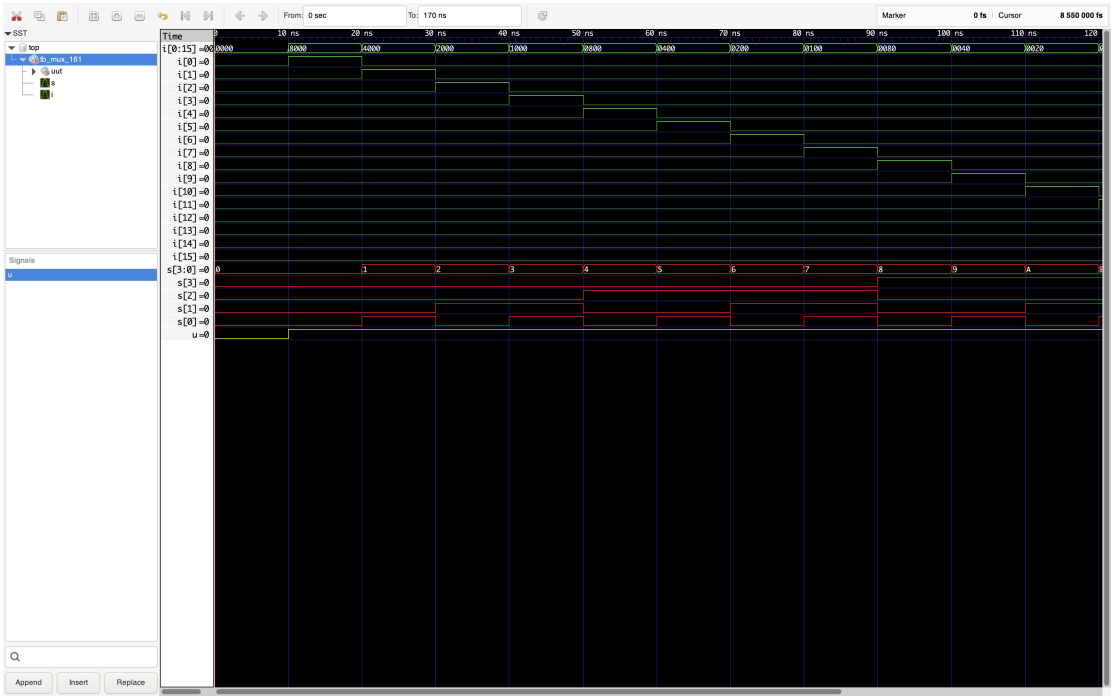

Figure 1.8: Risultati della simulazione: waveform

1.2 Rete di interconnessione a 16 ingressi e 4 uscite

Una rete di interconnessione è è un tipo di rete di commutazione che permette di instradare i segnali da un insieme di ingressi a un insieme più ridotto di uscite. Tale rete può essere progettata attraverso un adeguato utilizzo di Multiplexer e Demultiplexer.

Nel caso in esame, si vuole progettare una rete che prenda 16 ingressi e restituisca 4 uscite. Si utilizza anche in questo caso un approccio per composizione, a partire dal Multiplexer 16:1 implementato nell'esercizio precedente, la cui uscita sarà posta in ingresso a un Demultiplexer 1:4.

La rete complessiva sarà fatta in questo modo:

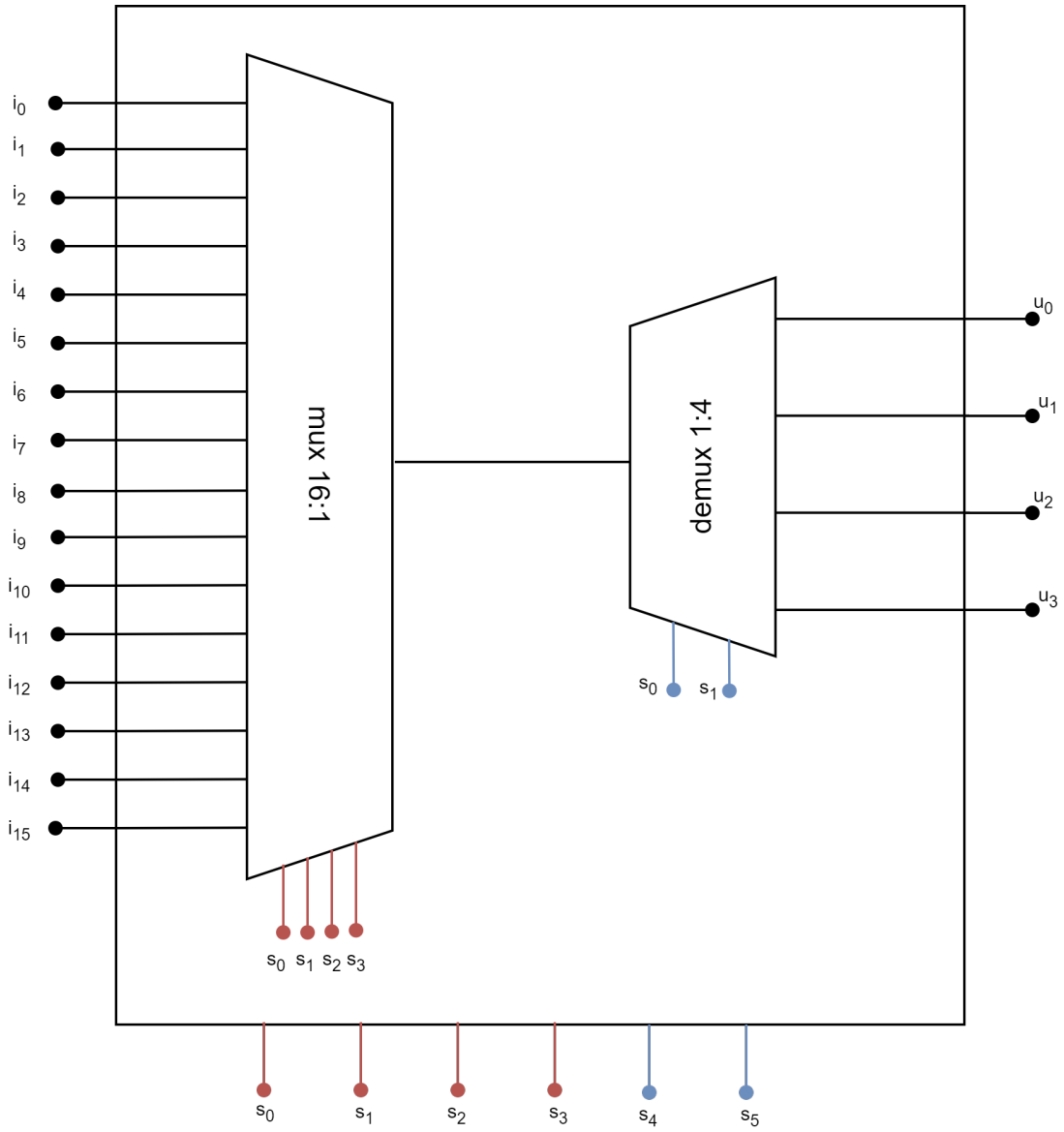

Figure 1.9: Rete di interconnesione

1.2.1 Progettazione

Anche in questo caso, prima di procedere all'implementazione della rete nel complesso, si costruisce il Demultiplexer 4:1 a partire da Demultiplexer 2:1.

Un Demultiplexer 1 : u è un dispositivo che prende un solo segnale di ingresso, due segnali di selezione e a partire da essi restituisce u uscite. Un Demultiplexer 2:1 è un dispositivo fatto in questo modo:

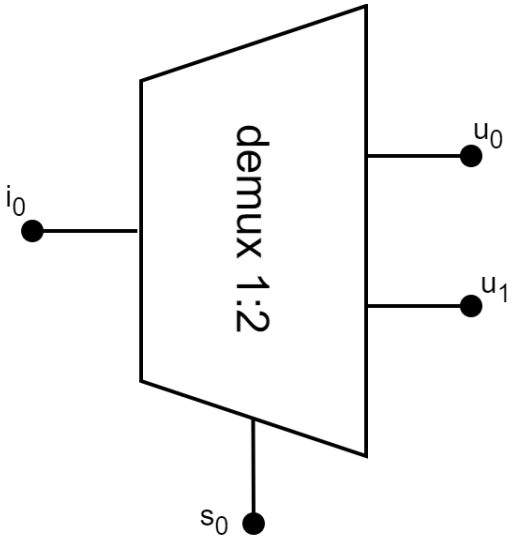

Figure 1.10: Demux 1:2

Tale componente è caratterizzato dalla seguente tabella di verità:

$\mathbf{s_0}$	$\mathbf{i_0}$	$\mathbf{u_0}$	$\mathbf{u_1}$
0	0	0	0
0	1	1	0
1	0	0	0
1	1	0	1

Table 1.2: Tabella di verità di un Demux 2:1

Da cui si ricavano le seguenti equazioni relative alle uscite:

$$u_0 = (i_0 \text{ AND } \bar{s_0})$$

$$u_1 = (i_0 \text{ AND } s_0)$$

A partire dalla composizione di dispositivi di questo tipo, si può real-

izzare un Demultiplexer 1:4, come rappresentato in figura.

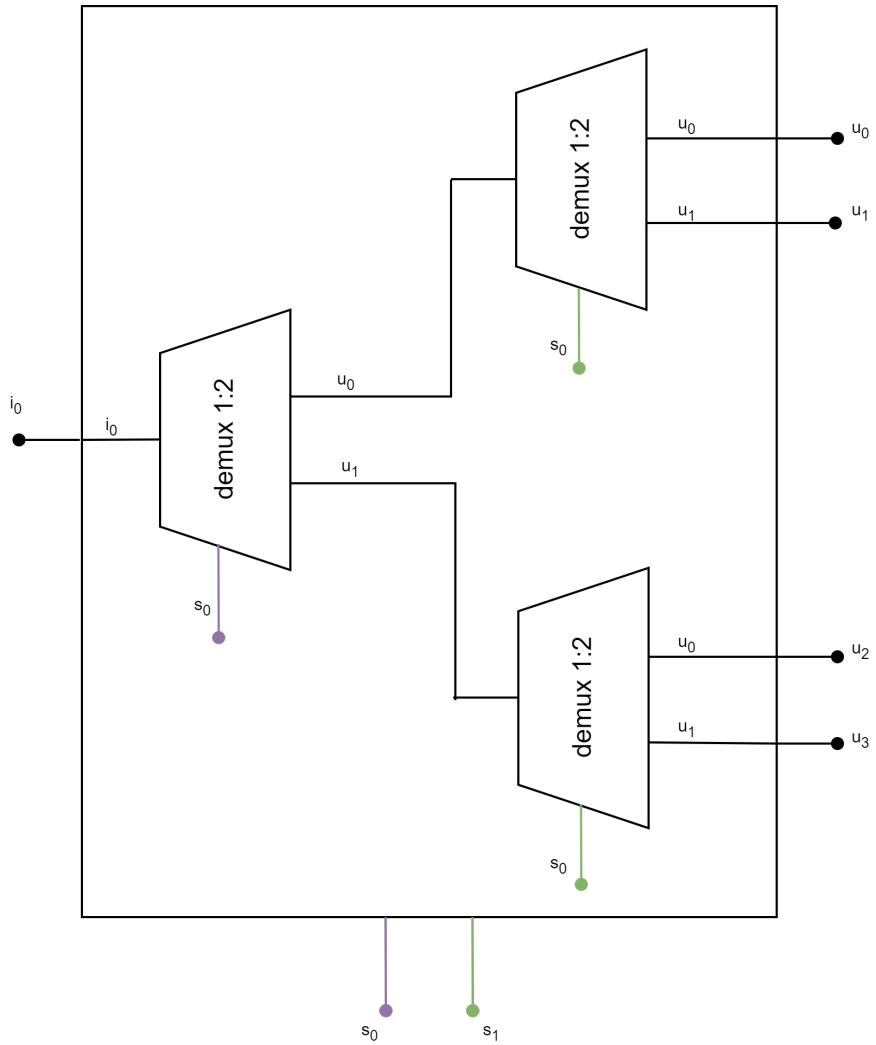

Figure 1.11: Demux 1:4 composto a partire da Demux 1:2

Utilizzando il Demultiplexer appena progettato, al cui ingresso si fa corrispondere l'uscita del Multiplexer 16:1, progettato nell'esercizio precedente, si ottiene la rete di interconnessione, così formata:

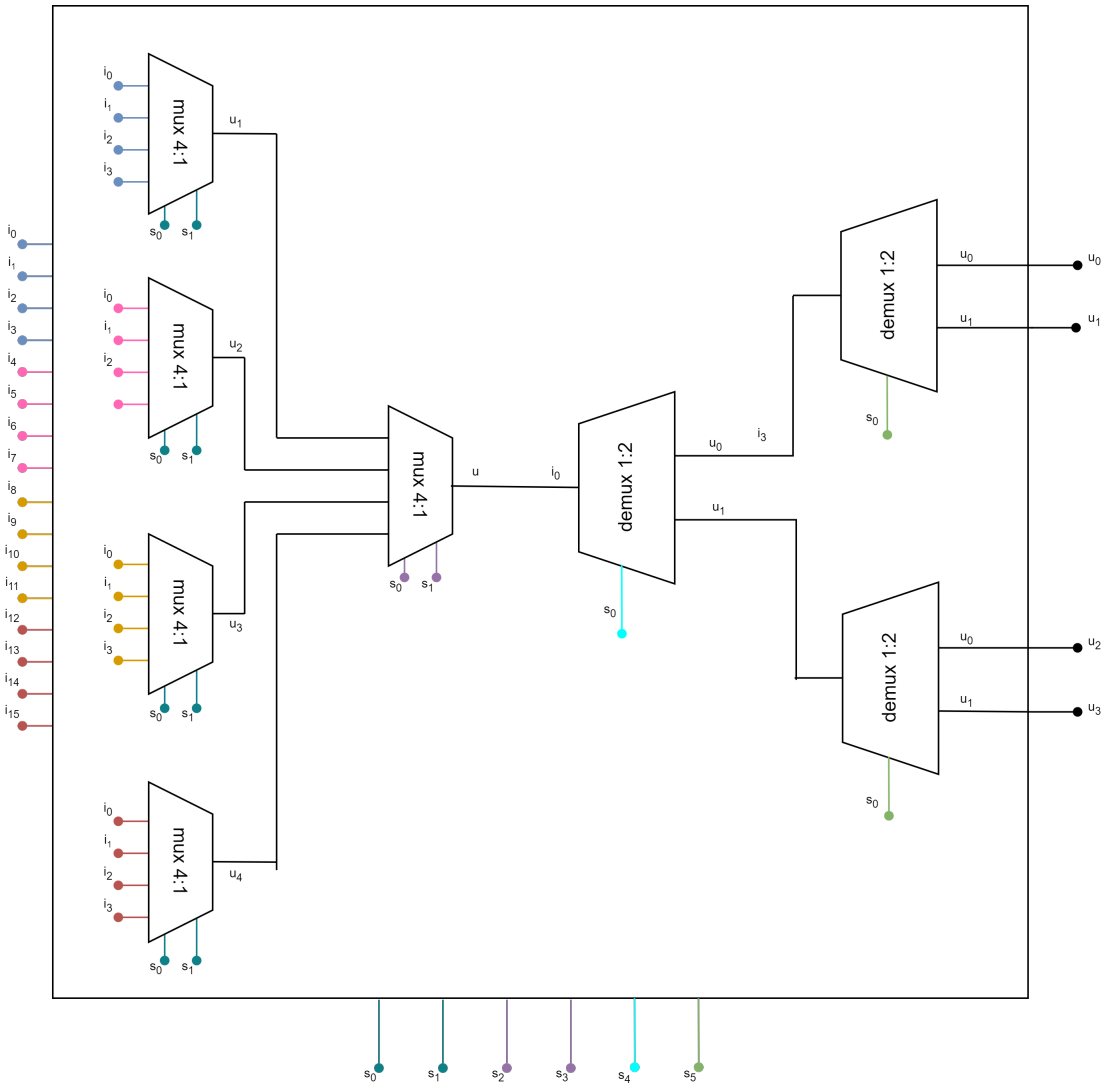

Figure 1.12: Rete di interconnessione: funzionamento interno

Nell'immagine, i colori sono stati usati per rendere più chiari i collegamenti tra segnali.

1.2.2 Implementazione

1.2.3 Simulazione

1.3 Implementazione su board del punto precedente

Chapter 2

Esercizio 2 - Sistema ROM+M

Il sistema che si vuole costruire consiste in due elementi principali: una ROM (Read-Only-Memory) puramente combinatoria e una macchina combinatoria M, che esegue una trasformazione sui dati letti da M e li pone in uscita. La ROM si compone di 16 locazioni di memoria, ciascuna contenente una stringa di 8 bit. Il sistema prende in ingresso un indirizzo di 4 bit, che permetterà di accedere a una delle locazioni della ROM; il dato in tale locazione viene posto in uscita alla ROM, e quindi in ingresso alla macchina M. La macchina M deve effettuare una trasformazione sulla stringa di 8 bit, in modo da restituire in uscita una stringa di 4 bit. La trasformazione scelta consiste nel sommare i 4 bit più significativi della stringa con i 4 bit meno significativi, la stringa di 4 bit risultante sarà restituita come uscita all'intero sistema.

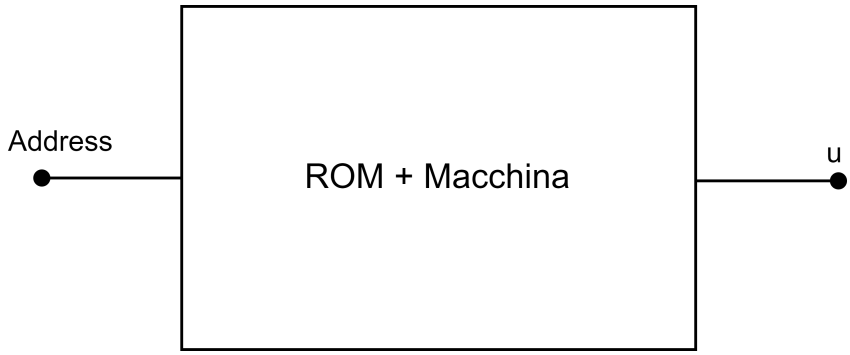

Figure 2.1: ROM + M

2.1 Progettazione

La progettazione consiste nella realizzazione dei due componenti fondamentali del sistema: ROM e M.

2.2 Implementazione

Dapprima si implementa la ROM, in cui sono memorizzati 16 elementi, ciascuno da 8 bit. Il codice sottostante crea l'entità ROM, al cui ingresso è presente un vettore da 4 bit di *std_logic* che rappresenta l'indirizzo, e in uscita restituisce un vettore di 8 bit. Vengono poi definite le stringhe di bit contenute nella ROM. Nel processo main, si pone in uscita l'elemento corrispondente alla locazione address.

```
1
2 library IEEE;
3 use IEEE.STD_LOGIC_1164.ALL;
4 use IEEE.NUMERIC_STD.ALL;
5
```

```
6
7  entity ROM is
8      port(
9          address: in STD_LOGIC_VECTOR(3 downto 0);
10         dout: out STD_LOGIC_VECTOR(7 downto 0)
11          );
12 end entity ROM;
13
14 architecture RTL of ROM is
15
16 type MEMORY is array(15 downto 0) of STD_LOGIC_VECTOR(7 downto 0);
   ↪  --memoria da N locazioni che contengono 8 bit
17 constant ROM_N: MEMORY := (
18     "01000000",  -- in locazione 15
19     "01000001",
20     "01000010",
21     "01000011",
22     "00010100",
23     "01000101",
24     "00000110",
25     "01000111",
26     "00001000",
27     "00001001",
28     "01001010",
29     "00001011",
30     "00001100",
31     "00001101",
32     "10001010",
33     "00001001"  --in locazione 0
34 );
35
36
37 begin
38 main: process(address)
39 begin
40 dout<= ROM_N(TO_INTEGER(unsigned(address))); --lettura dalla rom
41 end process main;
42 end architecture RTL;
```

Code 2.1: Implementazione ROM in VHDL

Si procede poi con l'implementazione del componente M, che effettua la trasformazione.

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.NUMERIC_STD.ALL;
4
5  entity M is
6      port(
7      ingresso: in std_logic_vector(7 downto 0);
8      uscita: out std_logic_vector(3 downto 0)
9      );
10
11 end entity M;
12
13 architecture Behavioral of M is
14 begin
15     process(ingresso)
16     begin
17         -- Somma dei 4 bit pi significativi e dei 4 meno
           ↪  significativi
18      uscita <= std_logic_vector(unsigned(ingresso(7 downto 4)) +
         ↪  unsigned(ingresso(3 downto 0)));
19     end process;
20 end Behavioral;
21
22
```

Code 2.2: Macchina M

Nel processo si pone come uscita della macchina la somma tra i bit più significativi dell'ingresso (dal bit 7 al 4) e dei bit meno significativi (dal bit 3 allo 0).

Le due componenti sono parte del sistema S che è così implementato:

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.NUMERIC_STD.ALL;
4
5  entity ROMplusM is
6        Port (
7          A: in std_logic_vector(3 downto 0); --indirizzo in ingresso
           ↪  al sistema
```

```
8         bout: out std_logic_vector(3 downto 0) --uscita complessiva
          ↪  del sistema
9        );
10 end ROMplusM;
11
12 architecture structural of ROMplusM is
13 signal u0 : std_logic_vector(7 downto 0) := "00000000";
14
15 component ROM
16             port(
17                 address: in STD_LOGIC_VECTOR(3 downto 0);
18                 dout: out STD_LOGIC_VECTOR(7 downto 0)
19             );
20             end component;
21 component M
22            port(
23             ingresso: in std_logic_vector(7 downto 0);
24             uscita: out std_logic_vector(3 downto 0)
25            );
26            end component;
27
28 begin
29 -- Istanza della ROM
30 rom_instance: ROM
31     port map(
32         address => A,
33         dout => u0
34     );
35 -- Istanza della macchina combinatoria M
36     transform: M
37         port map(
38             ingresso => u0,
39             uscita => bout
40         );
41 end structural;
```

Code 2.3: Sistema S

Tale sistema è stato costruito come structural: sono stati dichiarati i componenti, e ne sono state definite le istanze. Si è utilizzato un segnale di supporto $u0$, che funge da segnale intermedio tra l'uscita della ROM e l'ingresso della macchina.

Si osserva lo schematic fornito dall'ambiente di sviluppo Vivado:

Figure 2.2: Schematic di S

2.3 Simulazione

Per procedere alla simulazione si realizza un testbench, con diversi casi di test, che permettano di osservare il comportamento del sistema.

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.NUMERIC_STD.ALL;
4
5  entity ROMplusM_tb is
6  -- Un testbench non ha porte, n'entituota.
7  end ROMplusM_tb;
8
9  architecture behavior of ROMplusM_tb is
10
11     -- Component declaration for the unit under test (UUT)
12     component ROMplusM
13         Port (
14             A: in std_logic_vector(3 downto 0);
15             bout: out std_logic_vector(3 downto 0)
16         );
17     end component;
18
19     -- Signals to connect to the UUT
20     signal A_tb: std_logic_vector(3 downto 0) := (others => '0'); --
       ↪  Ingresso inizializzato a 0
21     signal bout_tb: std_logic_vector(3 downto 0); -- Uscita
22
```

```
23 begin
24
25     -- Instantiation of the UUT (Unit Under Test)
26     uut: ROMplusM
27         Port map (
28             A => A_tb,
29             bout => bout_tb
30         );
31
32     -- Stimulus process to provide inputs and check outputs
33     stimulus_process: process
34     begin
35         -- Test 1: Indirizzo A = 0
36         A_tb <= "0000";
37         wait for 10 ns;
38
39         -- Test 2: Indirizzo A = 1
40         A_tb <= "0001";
41         wait for 10 ns;
42
43         -- Test 3: Indirizzo A = 2
44         A_tb <= "0010";
45         wait for 10 ns;
46
47          -- Test 4: Indirizzo A = 3
48         A_tb <= "0011";
49         wait for 10 ns;
50
51          -- Test 5: Indirizzo A = 5
52         A_tb <= "0101";
53         wait for 10 ns;
54
55          -- Test 6: Indirizzo A = 7
56         A_tb <= "0111";
57         wait for 10 ns;
58
59          -- Test 7: Indirizzo A = 10
60         A_tb <= "1010";
61         wait for 10 ns;
62
63         -- Test 8: Indirizzo A = 255
64         A_tb <= "1111";
65         wait for 10 ns;
66
67         -- Fine simulazione
68         wait;
```

```
69      end process;
70
71  end behavior;
```

Code 2.4: Testbench

La seguente figura permette la visualizzazione delle waveform.

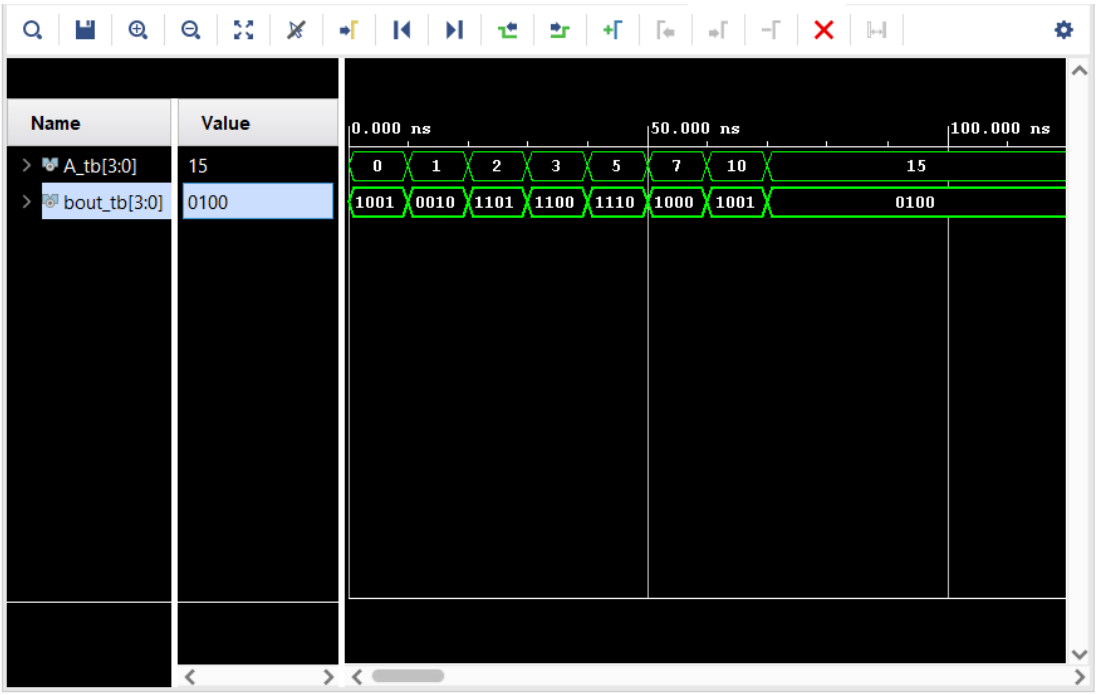

Figure 2.3: Waveform della simulazione di S

Si procede con dei test effettuati manualmente per mostrare la correttezza nel funzionamento del sistema S. Per consentire una maggiore leggibilità, si è scelto di visualizzare gli indirizzi come *Unsigned Decimal*.

Nel caso $A = 0$, si accede alla stringa 00001001, sommando i bit meno significativi con quelli più significativi si ottiene $0000 + 1001 = 1001$; nel caso $A = 5$, si accede alla stringa 01001010, e procedendo come sopra si ottiene $0100 + 1010 = 1110$.

Come si può vedere, i risultati di questi test coincidono con il comportamento atteso dal sistema e che sono mostrati nella waveform relativa alla simulazione.

Chapter 3

Title

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris.

3.1 Section ...

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis,

viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate

metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros,

malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

Chapter 4

Title

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris.

4.1 Section ...

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis,

viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate

metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros,

malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

Chapter 5

Conclusion

Donec et nisl id sapien blandit mattis. Aenean dictum odio sit amet risus. Morbi purus. Nulla a est sit amet purus venenatis iaculis. Vivamus viverra purus vel magna. Donec in justo sed odio malesuada dapibus. Nunc ultrices aliquam nunc. Vivamus facilisis pellentesque velit. Nulla nunc velit, vulputate dapibus, vulputate id, mattis ac, justo. Nam mattis elit dapibus purus. Quisque enim risus, congue non, elementum ut, mattis quis, sem. Quisque elit.

Bibliografia