

CORSO DI TECNOLOGIE DIGITALI

PROGETTO IN VHDL

Relazione scritta dallo studente Alessandro Ferrentino

Sommario

1. Introduzione

- Scopo del progetto
- Scelte progettuali effettuate

2. Schemi

- Diagramma a blocchi
- Automa a stati finiti

3. Realizzazione

- Controller in VHDL
- Test bench del controller
- Design e Reports

4. Simulazione

5. Conclusioni e progetto alternativo

1. Introduzione

- **Scopo del progetto**

Realizzare il controllore di un sistema di inscatolamento. Su un nastro trasportatore circolare arrivano casualmente oggetti da 1Kg e da 2Kg. Il sistema deve prendere gli oggetti e riempire totalmente delle scatole che possono contenere oggetti per 4Kg. Gli oggetti vengono messi dentro le scatole, se c'è posto, da un braccio meccanico. Quando la scatola è piena viene rimossa dal sistema attraverso un altro braccio meccanico differente dal precedente, ponendole su un pallet. L'automa ogni volta che si presenta un oggetto sul nastro deve decidere se metterlo nella scatola perché c'è ancora posto ed anche se rimuovere la scatola perché piena (dopo aver inserito l'oggetto). Dopo che il secondo braccio ha spostato 10 scatole, attiva una sirena che indica all'operatore di dover rimuovere il pallet. Fintanto che l'operatore non ha rimosso il pallet contenente le scatole, i bracci non possono muovere altri oggetti e scatole.

- **Scelte progettuali effettuate**

Si suppone sia possibile avviare il sistema solo se la scatola è presente e in posizione, dal momento che, in seguito all'azione del secondo braccio, potrebbe non esserlo. A tal scopo è necessario l'utilizzo di una fotocellula.

Si suppone l'esistenza di un sensore in grado di fornire il peso di un oggetto presente in una certa posizione del nastro. In particolare, il sensore fornisce l'uscita su due bit, il cui significato verrà meglio specificato nella terza sezione ([3. Realizzazione](#)) di questa relazione.

Il braccio è in grado di prendere un oggetto in movimento sul nastro ed impiega un certo tempo per prenderlo, inserirlo all'interno della scatola e ritornare nella sua posizione iniziale. Finché il braccio è occupato nel caricamento di un oggetto nella scatola, non è possibile inserire altri oggetti. Pertanto, si è scelto di usare un contatore in modo da sapere quando il braccio risulta nuovamente disponibile. Si precisa inoltre che, per semplicità di progetto, il tempo che il braccio impiega per effettuare le operazioni è di 10 colpi di clock.

In seguito alla sostituzione del pallet l'operatore comunica il corretto esito dell'operazione premendo un bottone, il quale permette al sistema di riprendere la sua normale esecuzione.

2. Schemi

- **Diagramma a blocchi**

In [Figura 2.1](#) si riporta una possibile realizzazione, mediante schema a blocchi, del sistema.

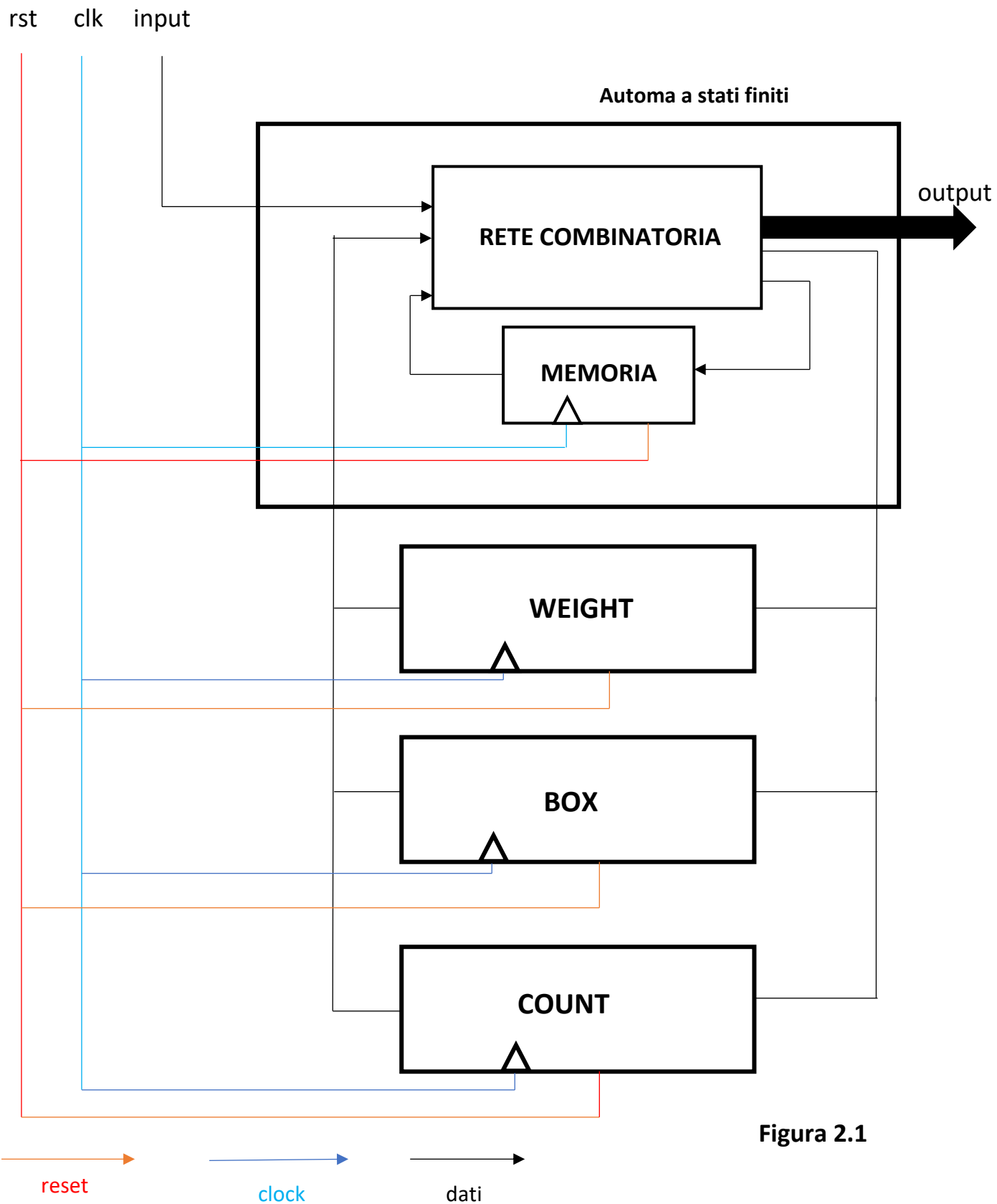


Figura 2.1

Nel diagramma sopra riportato, i blocchi “weight”, “box” e “count” sono dei contatori usati per memorizzare rispettivamente: il peso della scatola, il numero di scatole presenti sul pallet e quanti colpi di clock sono trascorsi dall’avvio del braccio. Il blocco “memoria” è necessario a memorizzare lo stato dell’automa.

- **Automa a stati finiti**

In [Figura 2.2](#) si riporta l’automa a stati finiti del sistema.

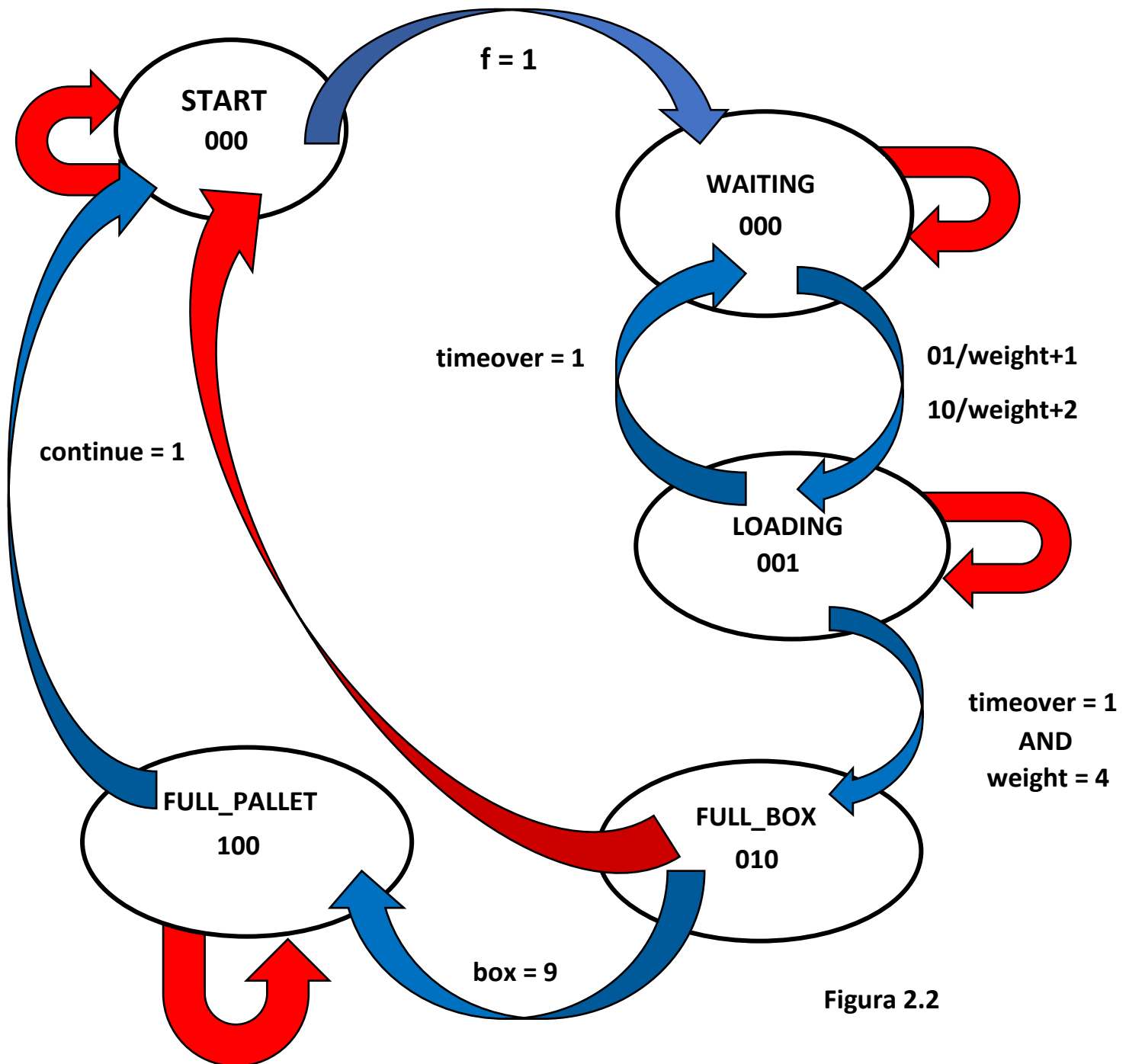



Figura 2.2

Le frecce rosse  sono state usate per indicare “others”, ossia una qualsiasi altra configurazione di ingressi.

Le uscite presenti negli stati dell'automa sono rispettivamente i segnali **sirena**, **B2** e **B1**, il cui significato è presentato nella sezione [3. Realizzazione](#).

In seguito, si riporta una breve descrizione degli stati che compongono l'automa.

Nello stato *start*, si attende che una scatola sia in posizione.

Nello stato *waiting*, si attende che un oggetto arrivi sul nastro e si sceglie se caricarlo o meno all'interno della scatola.

Nello stato *loading*, si attende che il braccio completi le operazioni per inserire l'oggetto.

Nello stato *full_box*, la scatola è piena (contiene 4kg) e si comunica al braccio di spostarla sul pallet.

Nello stato *full_pallet*, il pallet è pieno (contiene 10 scatole) e si attende che l'operatore lo sostituisca.

3. Realizzazione

Prima di riportare i codici in VHDL si presenta una descrizione delle variabili e segnali utilizzati.

Con **p1** e **p0** sono stati indicati rispettivamente il bit più significativo e quello meno significativo del sensore di peso. Nella tabella in [Figura 3.1](#) si presenta il significato da attribuire ai possibili valori.

p1 p0	Significato
00	Nessun oggetto presente
01	Oggetto di 1kg
10	Oggetto di 2kg
11	Valore non previsto

Figura 3.1

Con **f** si è indicato il segnale generato dalla fotocellula per rilevare la presenza della scatola. Questo assume il valore 1 se l'oggetto è presente, il valore 0 altrimenti.

I segnali di **clk** e **rst** per indicare rispettivamente clock e reset.

Il segnale **continue**, il quale assume il valore 1 in seguito alla pressione del bottone da parte dell'operatore. Si suppone inoltre che il segnale assuma nuovamente il valore 0 dopo un tempo sufficientemente piccolo.

I segnali **B1** e **B2**, utilizzati per attivare rispettivamente il braccio 1 (per spostare gli oggetti nelle scatole) ed il braccio 2 (per spostare le scatole sul pallet).

Si suppone che i bracci funzionino sui fronti di salita del segnale, è quindi sufficiente che il valore cambi da 0 ad 1 affinché il braccio esegua le operazioni previste.

Il segnale **sirena**, usato per attivare la sirena quando questo assume il valore 1.

In segnale **timer**, utilizzato per avviare il contatore *count*, il quale, al termine del conteggio, porta il valore del segnale **timeover** ad 1.

- **Controller in VHDL**

```
-- Company:
-- Engineer:
--
-- Create Date: 12:38:58 12/04/2018
-- Design Name:
-- Module Name: controller - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
-- use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity controller is
    Port ( p1, p0 : in STD_LOGIC;
           f : in STD_LOGIC;
           clk, rst : in STD_LOGIC;
           continue : in STD_LOGIC;
           B1, B2 : out STD_LOGIC;
           sirena : out STD_LOGIC);
end controller;
```

architecture Behavioral of controller is

type state is (start, waiting, loading, full_box, full_pallet);

signal cstate, nstate : state;

signal weight, nweight : integer range 0 to 4;

signal box, nbox : integer range 0 to 9;

signal count, ncount : integer range 0 to 9;

signal timeover : std_logic := '0';

signal timer : std_logic := '0';

begin

-- Salvo lo stato corrente e i valori dei contatori.

up_state: process(clk)

begin

if rising_edge(clk) then

if (rst='1') then

cstate <= start;

weight <= 0;

box <= 0;

count <= 0;

else

cstate <= nstate;

weight <= nweight;

box <= nbox;

count <= ncount;

end if;

end if;

end process;

-- Aggiorno lo stato prossimo e le uscite.

Finite_State_Machine: process(cstate, p1, p0, f, continue, timeover, weight, box)

-- tmp_out = sirena & timer & B2 & B1

variable tmp_out : std_logic_vector(3 downto 0);

begin

case (cstate) is

-- Aspetto che una scatola venga messa in posizione

when start =>

tmp_out := "0000";

nweight <= weight;

nbox <= box;

if f = '1' then

nstate <= waiting;

else

nstate <= start;

end if;

-- Aspetto che arrivi un oggetto sul nastro e scelgo se metterlo nella scatola o meno

when waiting =>

```
tmp_out := "0000";
nbox <= box;
-- Arriva un oggetto di 1kg
if (p1 = '0' and p0 = '1') then
    nstate <= loading;
    nweight <= weight + 1;
-- Arriva un oggetto di 2kg
elsif (p1 = '1' and p0 = '0' and weight < 3) then
    nstate <= loading;
    nweight <= weight + 2;
-- Non arriva nessun oggetto
else
    nstate <= waiting;
    nweight <= weight;
end if;
```

-- Aspetto che la macchina finisca di caricare l'oggetto nella scatola

when loading =>

```
tmp_out := "0101";
nbox <= box;
if timeover = '1' then
    if weight = 4 then
        nstate <= full_box;
        nweight <= 0;
    else
        nstate <= waiting;
        nweight <= weight;
    end if;
else
    nstate <= loading;
    nweight <= weight;
end if;
```

-- Sposto la scatola sul pallet

when full_box =>

tmp_out := "0010";

nweight <= weight;

-- La prossima scatola riempira' il pallet

if box = 9 then

nstate <= full_pallet;

nbox <= 0;

else

nstate <= start;

nbox <= box + 1;

end if;

-- Attivo la sirena ed aspetto il segnale da parte dell'operatore

when full_pallet =>

tmp_out := "1000";

nweight <= weight;

nbox <= box;

if continue = '1' then

nstate <= start;

else

nstate <= full_pallet;

end if;

end case;

sirena <= tmp_out(3);

timer <= tmp_out(2);

B2 <= tmp_out(1);

B1 <= tmp_out(0);

end process;

-- Contatore per il caricamento di un oggetto nella scatola

counter_timer: process(timer, count)

begin

if timer = '1' then

if count = 9 then

ncount <= 0;

timeover <= '1';

else

ncount <= count + 1;

timeover <= '0';

end if;

else

ncount <= count;

timeover <= '0';

end if;

end process;

end Behavioral;

- **Test bench del controller**

```
-----  
-- Company:  
-- Engineer:  
--  
-- Create Date: 21:45:12 12/04/2018  
-- Design Name:  
-- Module Name: /home/ise/Desktop/Xilinx_Projects/Inscatolamento/controller_tb.vhd  
-- Project Name: Inscatolamento  
-- Target Device:  
-- Tool versions:  
-- Description:  
--  
-- VHDL Test Bench Created by ISE for module: controller  
--  
-- Dependencies:  
--  
-- Revision:  
-- Revision 0.01 - File Created  
-- Additional Comments:  
--  
-- Notes:  
-- This testbench has been automatically generated using types std_logic and  
-- std_logic_vector for the ports of the unit under test. Xilinx recommends  
-- that these types always be used for the top-level I/O of a design in order  
-- to guarantee that the testbench will bind correctly to the post-implementation  
-- simulation model.
```

```
-----  
LIBRARY ieee;  
USE ieee.std_logic_1164.ALL;  
  
-- Uncomment the following library declaration if using  
-- arithmetic functions with Signed or Unsigned values  
--USE ieee.numeric_std.ALL;  
  
ENTITY controller_tb IS  
END controller_tb;
```

ARCHITECTURE behavior OF controller_tb IS

-- Component Declaration for the Unit Under Test (UUT)

```
COMPONENT controller
PORT(
    p1 : IN std_logic;
    p0 : IN std_logic;
    f : IN std_logic;
    clk : IN std_logic;
    rst : IN std_logic;
    continue : IN std_logic;
    B1 : OUT std_logic;
    B2 : OUT std_logic;
    sirena : OUT std_logic
);
END COMPONENT;
```

--Inputs

```
signal p1 : std_logic := '0';
signal p0 : std_logic := '0';
signal f : std_logic := '0';
signal clk : std_logic := '0';
signal rst : std_logic := '0';
signal continue : std_logic := '0';
```

--Outputs

```
signal B1 : std_logic;
signal B2 : std_logic;
signal sirena : std_logic;
```

-- Clock period definitions

```
constant clk_period : time := 10 ns;
```

BEGIN

-- Instantiate the Unit Under Test (UUT)

uut: controller PORT MAP (

p1 => p1,

p0 => p0,

f => f,

clk => clk,

rst => rst,

continue => continue,

B1 => B1,

B2 => B2,

sirena => sirena

);

-- Clock process definitions

clk_process :process

begin

clk <= '0';

wait for clk_period/2;

clk <= '1';

wait for clk_period/2;

end process;

-- Stimulus process

stim_proc: process

begin

-- Sequenza di reset

rst <= '1';

wait for clk_period;

rst <= '0';

wait for clk_period;

```

-- Carico 10 scatole per verificare che lo stato Full_pallet venga attivato
for i in 1 to 10 loop
    -- Al prossimo colpo di clock l'automa evolvera' in waiting
    f <= '1';
    wait for clk_period;

    -- Carico due oggetti da 2kg per riempire velocemente la scatola
    for j in 1 to 2 loop
        -- Al prossimo colpo di clock l'automa passera' allo stato di loading
        p1 <= '1';
        p0 <= '0';
        wait for clk_period;

        -- Attendo 10 colpi di clock (tempo impiegato dal braccio per spostare un
oggetto)
        wait for clk_period*10;
    end loop;

    -- Mi trovo in full_box, dopo questo colpo di clock l'automa evolve in:
    -- start, se il numero di scatole posizionate e' minore di 9,
    -- full_pallet, se il numero di scatole posizionate pari a 9.
    -- Abbasso f perche' ho tolto la scatola
    f <= '0';
    wait for clk_period;
end loop;

-- L'automa e' in full_pallet
wait for clk_period;

-- Al prossimo colpo di clock l'automa passera' nello stato start
continue <= '1';
wait for clk_period;
continue <= '0';
wait for clk_period;

-- Alzo il valore della fotocellula, al prossimo colpo di clock l'automa passera' nello stato
waiting
f <= '1';
wait for clk_period;

-- Verifico che con gli ingressi 00 ed 11 (p1 p0) l'automa non cambia stato
p1 <= '0';
p0 <= '0';
wait for clk_period*5;
p1 <= '1';
p0 <= '1';
wait for clk_period*5;

```

```
-- Ora voglio verificare che il sistema non carica un oggetto di 2kg
-- quando sono presenti gia' 3kg nella scatola
-- Inizio col caricare 1 solo kg
p1 <= '0';
p0 <= '1';
wait for clk_period;
-- L'automa si trova nello stato loading, ne approfitto per verificare
-- che l'automa non cambia stato con le possibili configurazioni di p1 p0
p1 <= '0';
p0 <= '0';
wait for clk_period;
p1 <= '0';
p0 <= '1';
wait for clk_period;
p1 <= '1';
p0 <= '0';
wait for clk_period;
p1 <= '1';
p0 <= '1';
wait for clk_period*7;

-- Carico 2kg
p1 <= '1';
p0 <= '0';
wait for clk_period;
p1 <= '0';
wait for clk_period*10;
-- dopo 10 colpi di clock l'automa si trova nello stato waiting,
-- verifico che il sistema non carica gli oggetti di 2kg
p1 <= '1';
p0 <= '0';
wait for clk_period*5;
-- carico 1kg
p1 <= '0';
p0 <= '1';
wait for clk_period;

-- Non arrivano piu' oggetti
p1 <= '0';
p0 <= '0';
```

```
wait;
end process;
```

```
END;
```

● Design e Reports

controller Project Status					
Project File:	Inscatolamento.xise	Parser Errors:	No Errors		
Module Name:	controller	Implementation State:	Synthesized		
Target Device:	xc6s1x4-3csg225	• Errors:	No Errors		
Product Version:	ISE 14.7	• Warnings:	No Warnings		
Design Goal:	Balanced	• Routing Results:			
Design Strategy:	Xilinx Default (unlocked)	• Timing Constraints:			
Environment:	System Settings	• Final Timing Score:			

Device Utilization Summary (estimated values)					
Logic Utilization	Used	Available	Utilization		
Number of Slice Registers	14	4800			0%
Number of Slice LUTs	26	2400			1%
Number of fully used LUT-FF pairs	14	26			53%
Number of bonded IOBs	9	132			6%
Number of BUFG/BUFGCTRLs	1	16			6%

Detailed Reports						
Report Name	Status	Generated	Errors	Warnings	Infos	
Synthesis Report	Current	Wed Dec 5 02:09:05 2018	0	0	1 Info (1 new)	
Translation Report						
Map Report						
Place and Route Report						
Power Report						
Post-PAR Static Timing Report						
Bitgen Report						


```

=====
*                               HDL Synthesis                               *
=====

Synthesizing Unit <controller>.
  Related source file is "/home/ise/Desktop/Xilinx_Projects/Inscatolamento/controller.vhd".
  Found 3-bit register for signal <weight>.
  Found 4-bit register for signal <box>.
  Found 4-bit register for signal <count>.
  Found 3-bit register for signal <cstate>.
  Found finite state machine <FSM_0> for signal <cstate>.

-----
| States           | 5 |
| Transitions      | 14 |
| Inputs           | 8 |
| Outputs          | 7 |
| Clock            | clk (rising_edge) |
| Reset            | rst (positive) |
| Reset type       | synchronous |
| Reset State      | start |
| Power Up State   | start |
| Encoding         | auto |
| Implementation   | LUT |
-----

Found 3-bit adder for signal <weight[2]_GND_3_o_add_9_OUT> created at line 94.
Found 3-bit adder for signal <weight[2]_GND_3_o_add_11_OUT> created at line 98.
Found 4-bit adder for signal <box[3]_GND_3_o_add_22_OUT> created at line 132.
Found 4-bit adder for signal <count[3]_GND_3_o_add_31_OUT> created at line 160.
Found 3-bit comparator greater for signal <weight[2]_GND_3_o_LessThan_11_o> created at line 96
Summary:
  inferred 3 Adder/Subtractor(s).
  inferred 11 D-type flip-flop(s).
  inferred 1 Comparator(s).
  inferred 5 Multiplexer(s).
  inferred 1 Finite State Machine(s).
Unit <controller> synthesized.

```

```

=====
*                               Low Level Synthesis                               *
=====

Analyzing FSM <MFsm> for best encoding.
Optimizing FSM <FSM_0> on signal <cstate[1:3]> with user encoding.

-----
| State           | Encoding |
-----
| start           | 000 |
| waiting         | 001 |
| loading         | 010 |
| full_box        | 011 |
| full_pallet     | 100 |
-----

Optimizing unit <controller> ...

Mapping all equations...
Building and optimizing final netlist ...
Found area constraint ratio of 100 (+ 5) on block controller, actual ratio is 1.

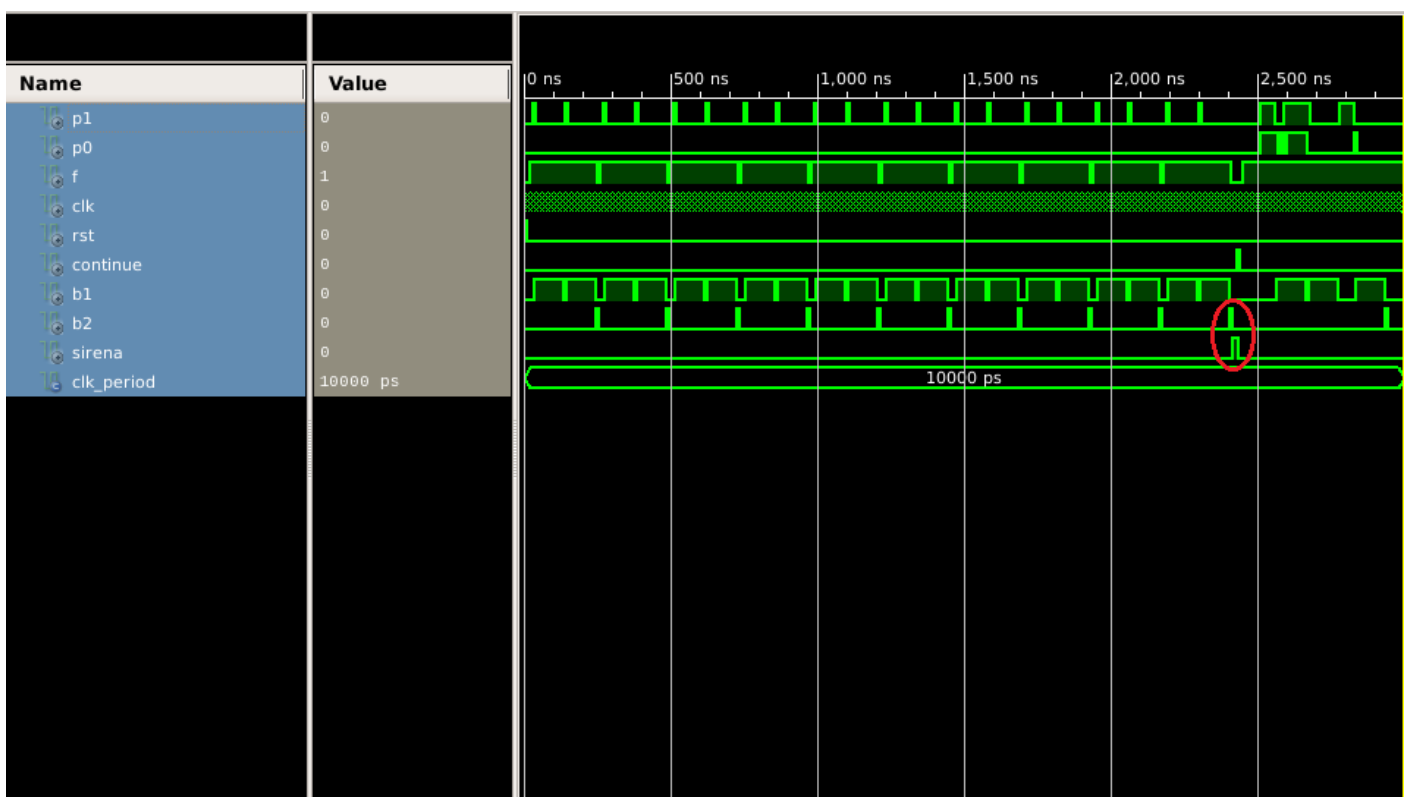
Final Macro Processing ...

```

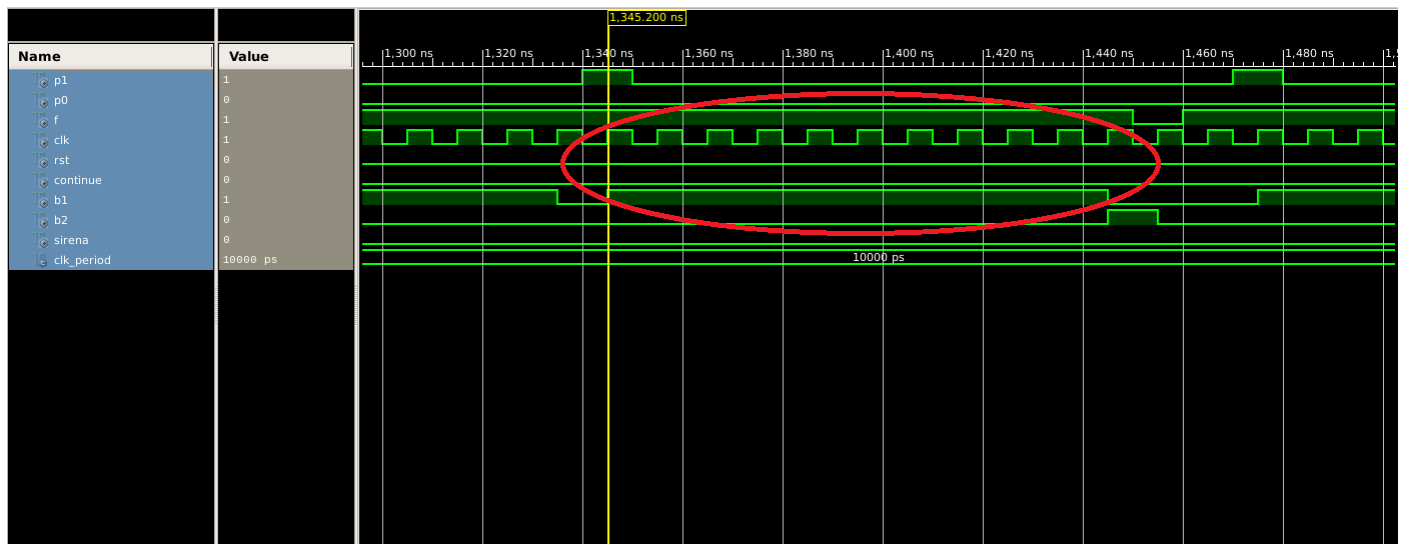
4. Simulazione

In questa sezione vengono riportati e commentati i risultati ottenuti dal simulatore ISim.

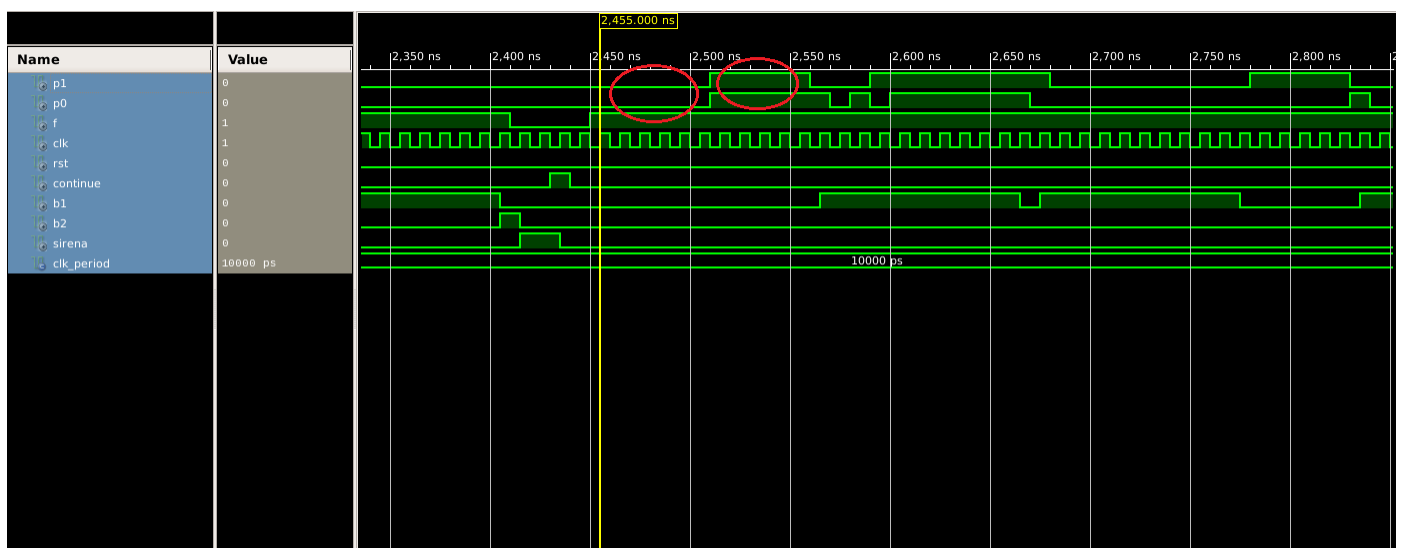
Si fa notare al lettore che alcune uscite assumono il valore alto soltanto in determinati stati (ad esempio il valore **B1** è alto solo nello stato di *loading*). Questa considerazione risulta utile, in alcuni casi, per verificare il corretto funzionamento del sistema.



Osserviamo che, dopo aver inserito 10 scatole sul pallet (il valore di **B2** passa da basso ad alto), la sirena viene attivata, come da specifica. Si può vedere inoltre che prima di caricare una scatola sul pallet, questa viene riempita con due oggetti di 2kg (il valore di **B1** passa da basso ad alto e **p1p0** = 10).

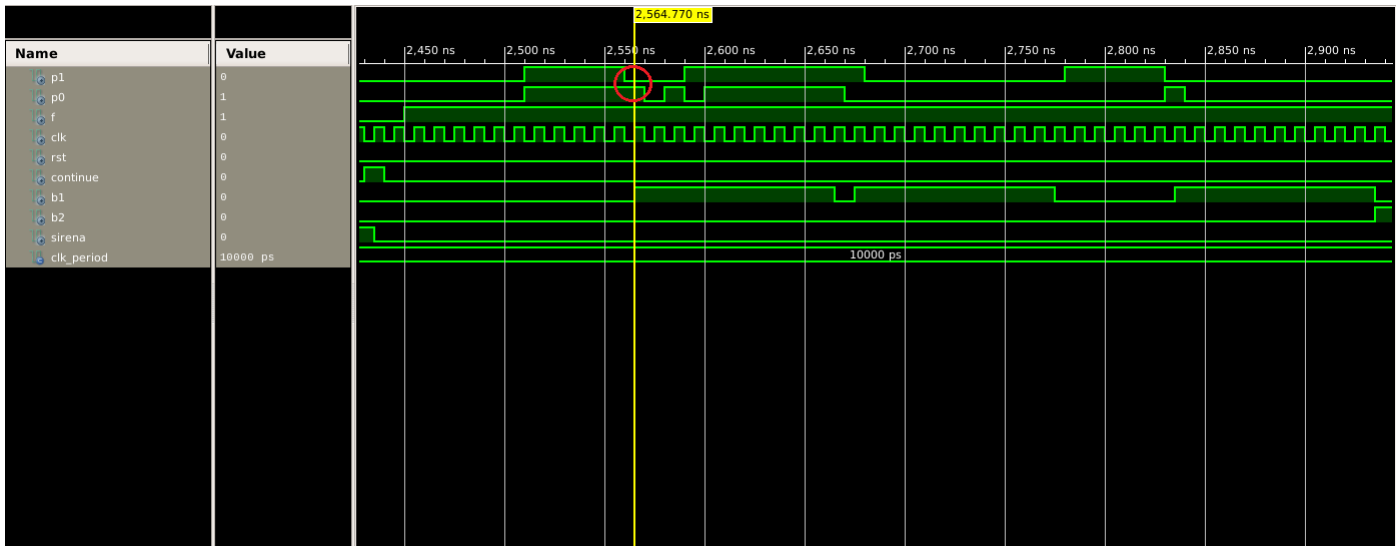


Il sistema resta per 10 colpi di clock nello stato di *loading*.



In seguito alla commutazione del segnale **continue**, il sistema passa dallo stato *full_pallet* a quello di *start*.

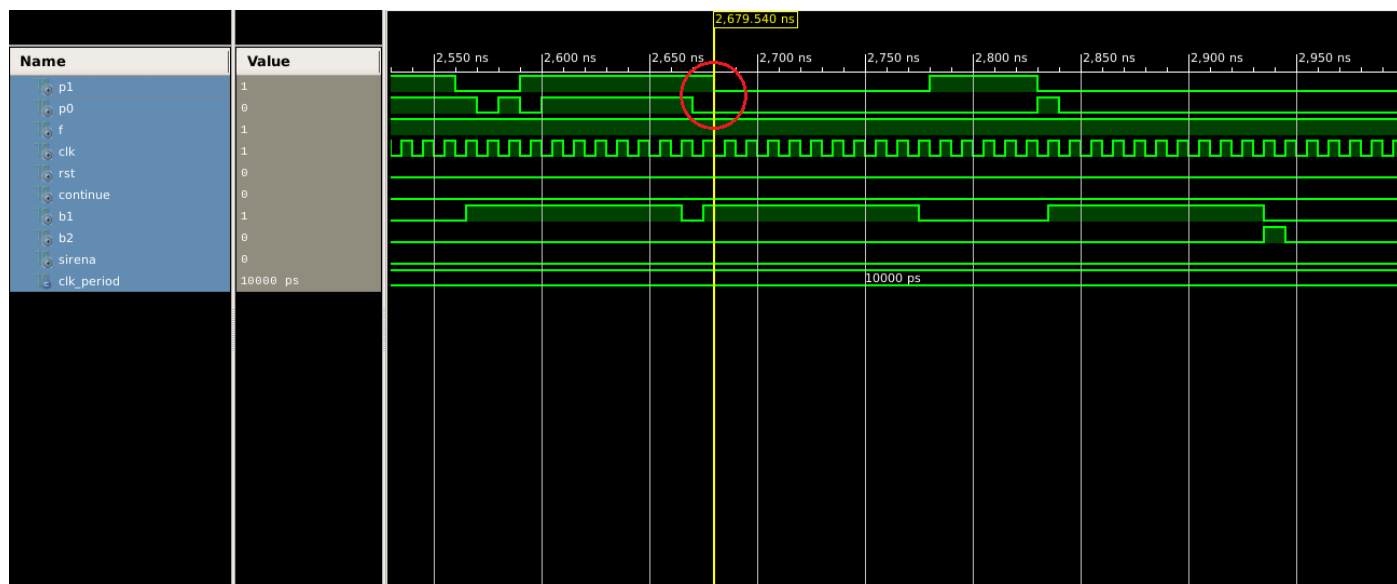
Successivamente, dopo che il segnale **f** assume il valore alto, il sistema passa dallo stato di *start* a quello di *waiting*, nel quale le configurazioni di ingressi 00 ed 11 dei segnali **p1p0** non ne alterano lo stato.



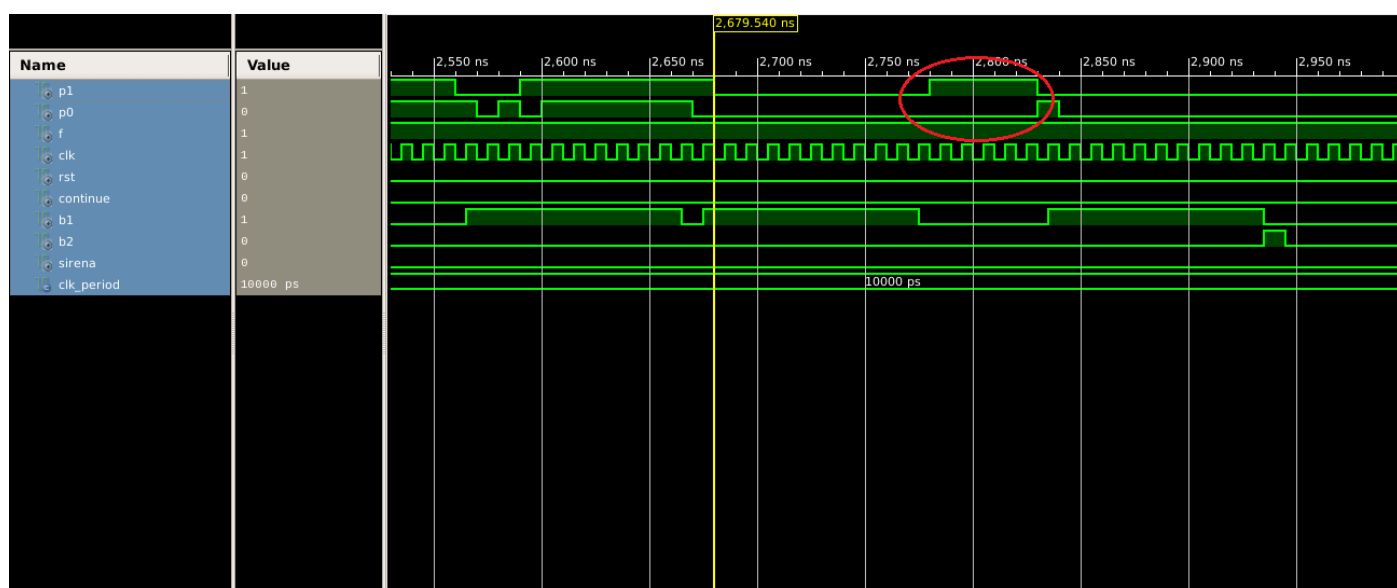
In seguito all'arrivo di un oggetto di 1kg, questo viene caricato nella scatola. La scatola contiene dunque 1kg.



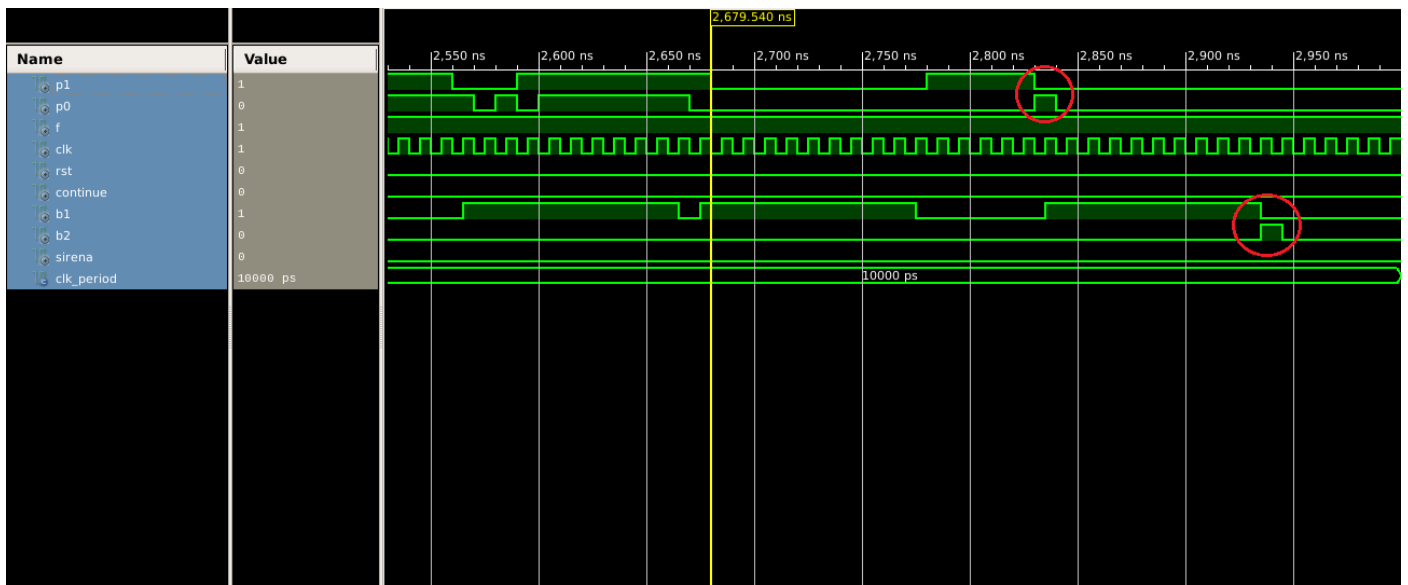
Notiamo che qualsiasi configurazione di valori assunti da **p1p0** durante lo stato di *loading*, non alterano lo stato del sistema.



In seguito all'arrivo di un oggetto di 2kg, questo viene caricato nella scatola. La scatola contiene ora 3kg.



Notiamo che nonostante sia presente un oggetto di 2kg sul nastro, questo non viene caricato dal braccio, in quanto violerebbe le specifiche: infatti sono presenti già 3kg nella scatola, la cui capienza massima è di 4kg.



Solo in seguito all'arrivo di un oggetto di 1kg, il braccio si mobilita e lo carica all'interno della scatola, riempiendola.

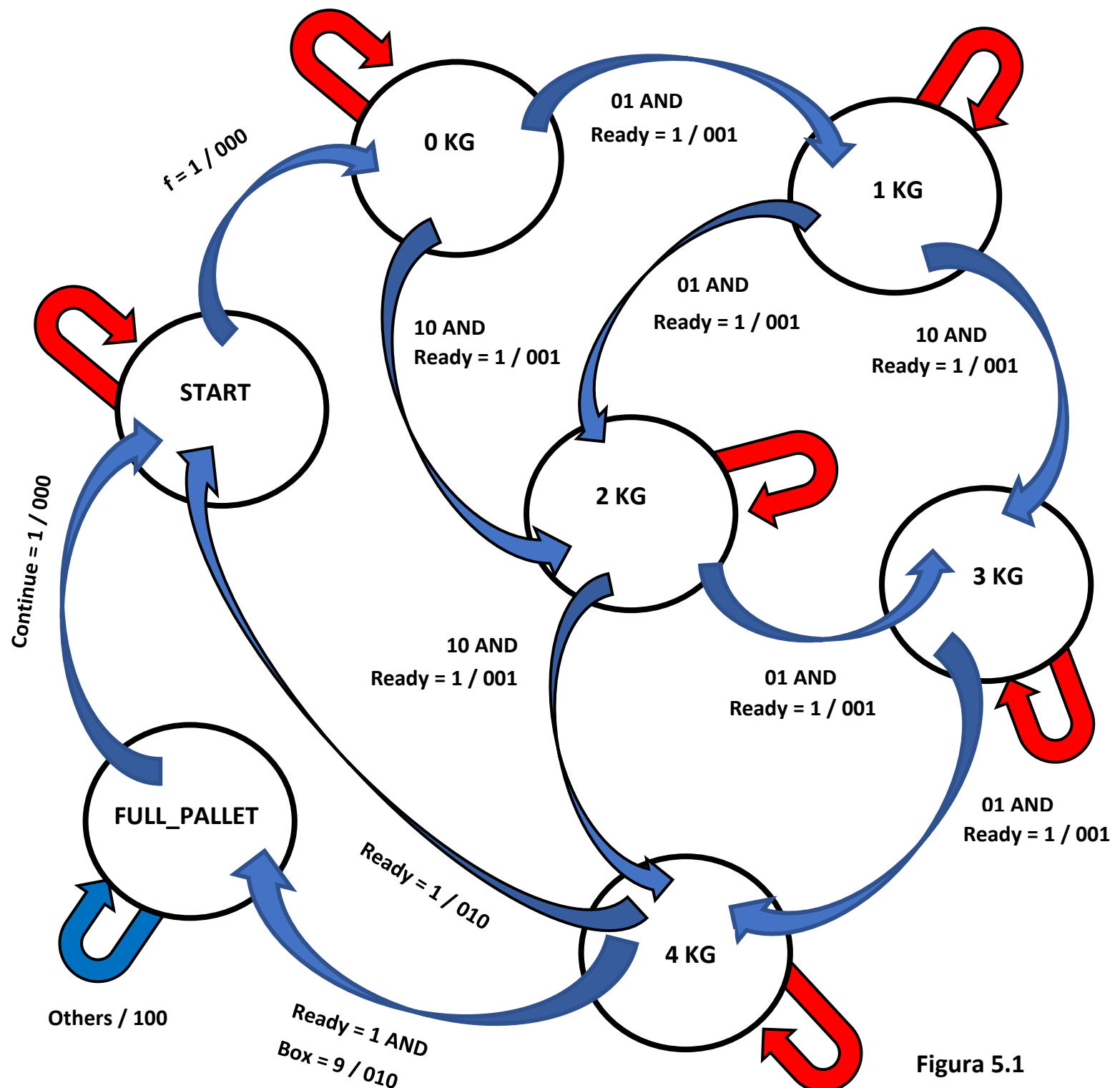
Infatti, possiamo notare che, una volta effettuato il caricamento, la scatola viene raccolta dal secondo braccio e posizionata sul pallet.


5. Conclusioni e progetto alternativo

Dalle osservazioni effettuate sulle uscite, possiamo concludere, dunque, che il sistema rispetta appieno le specifiche.

Si nota, inoltre, che la soluzione offerta non è l'unica e che si sarebbero potute considerare anche scelte progettuali differenti.

A titolo di esempio, avremmo potuto realizzare il sistema seguendo la logica dell'automa a stati finiti presentato in [Figura 5.1](#).



Le frecce rosse  sono state usate per indicare la configurazione “others / 000” .
Le uscite riportate sono le stesse del progetto recente, ossia: **sirena, B2, B1.**

Nel seguente progetto l'informazione del peso della scatola non è più associata ad un contatore, bensì è codificata negli stati dell'automa, rendendolo, tuttavia, più complesso.

Un'ulteriore differenza rispetto al primo progetto presentato riguarda il braccio robotico: è presente adesso un segnale di ingresso chiamato “**ready**”, il quale ci informa se il braccio è disponibile o meno. La presenza di questo segnale di ingresso ci permette di rimuovere uno stato (quello di *loading*) ed un contatore, dunque semplifica il progetto del controllore impiegato per l'inscatolamento, tuttavia complica, seppure leggermente, il controllore del braccio robotico, il quale dovrà adesso gestire tale segnale.

In ultima analisi possiamo dunque osservare che il seguente progetto permette di eliminare i contatori associati al braccio e al peso della scatola al costo di complicare leggermente l'automa e di aggiungere un ulteriore segnale di ingresso.