Progetto Reti Logiche

Francesco Gianchino: 869944
Gabriele Gallotti: 867634
Anno 2018-19

1 Specifica

1.1 : DESCRIZIONE INTERFACCIA COMPONENTE

Il progetto si basa sul progettare e implementare un componente VHDL che si interfaccia con una RAM per risolvere un problema di natura matematica.
Il componente da descrivere ha la seguente interfaccia:

```
entity project_reti_logiche is
port (
i_clk : in std_logic;
i_start : in std_logic;
i_rst : in std_logic;
i_data : in std_logic_vector(7 downto 0);
o_address : out std_logic_vector(15 downto 0);
o_done : out std_logic;
o_en : out std_logic;
o_we : out std_logic;
o_data : out std_logic_vector (7 downto 0)
);
end project_reti_logiche;
```

In particolare:

- i_clk è il segnale di CLOCK in ingresso generato dal TestBench;
- i_start è il segnale di START generato dal Test Bench;
- i_rst è il segnale di RESET che inizializza la macchina pronta per ricevere il primo segnale di START;
- i_data è il segnale (vettore) che arriva dalla memoria in seguito ad una richiesta di lettura;

• o_address è il segnale (vettore) di uscita che manda l'indirizzo alla memoria;
• o_done è il segnale di uscita che comunica la fine dell'elaborazione e il dato di uscita scritto in memoria;
• o_en è il segnale di ENABLE da dover mandare alla memoria.
• o_we è il segnale di WRITE ENABLE da dover mandare alla memoria.
• o_data è il segnale (vettore) di uscita dal componente verso la memoria.

1.2 : RAM

La RAM si interfaccia col componente attraverso i segnali:

• o_en
• o_we
• o_data
• o_address
• i_data

Il funzionamento per scrittura e lettura è il seguente:

LETTURA

o_en deve essere messo a 1 , o_we a 0 e nel bus o_address va indicato l'indirizzo di memoria dove si deve leggere una stringa di 8 bit.
La memoria entro un ciclo di clock risponderà con la stringa di 8 bit inviandoli sul bus i_data.

SCRITTURA

o_en deve essere messo a 1 , o_we a 1 e rispettivamente nel bus o_address e e o_data vanno indicati l'indirizzo di memoria dove scrivere e la stringa di 8 bit da scrivere.La scrittura viene fatta entro un ciclo di clock.

STRUTTURA MEMORIA PER IL PROGETTO

I dati ciascuno di dimensione 8 bit sono memorizzati in una memoria con indirizzamento al Byte partendo dalla posizione 0.

- L'indirizzo 0 è usata per memorizzare il numero di centroidi (Maschera di ingresso: definisce quale centroide deve essere esaminato);
- Gli indirizzi dal 1 al 16 sono usati per memorizzare le coordinate a coppie X e Y dei centroidi: -
- 1 - Coordinata X 1° Centroide
- 2 - Coordinata Y 1° Centroide
- 3 - Coordinata X 2° Centroide
- 4 - Coordinata Y 2° Centroide

…

- 15 - Coordinata X 8o Centroide
- 16 - Coordinata Y 8o Centroide
- Gli indirizzi 17 e 18 sono usati per memorizzare le Coordinate X e Y del punto da valutare
- L'indirizzo 19 è usato per scrivere, alla fine, il valore della maschera di uscita.

1.3 : SPECIFICA

Sia dato uno spazio bidimensionale definito in termini di dimensione orizzontale e verticale, e siano date le posizioni di N punti, detti "centroidi", appartenenti a tale spazio. Si vuole implementare un componente HW descritto in VHDL che, una volta fornite le coordinate di un punto appartenente a tale spazio, sia in grado di valutare a quale/i dei centroidi risulti più vicino (Manhattan distance). Lo spazio in questione è un quadrato (256x256) e le coordinate nello spazio dei centroidi e del punto da valutare sono memorizzati in una memoria (la cui implementazione non è parte del progetto). La vicinanza al centroide viene espressa tramite una maschera di bit (maschera di uscita) dove ogni suo bit corrisponde ad un centroide: il bit viene posto a 1 se il centroide è il più vicino

al punto fornito, 0 negli altri casi. Nel caso il punto considerato risulti equidistante da 2 (o più) centroidi, i bit della maschera d'uscita relativi a tali centroidi saranno tutti impostati ad 1. Degli N centroidi K<=N sono quelli su cui calcolare la distanza dal punto dato. I K centroidi sono indicati da una maschera di ingresso a N bit: il bit a 1 indica che il centroide è valido (punto dal quale calcolare la distanza) mentre il bit a 0 indica che il centroide non deve essere esaminato. Si noti che la maschera di uscita è sempre a N bit e che i bit a 1 saranno non più di K.

Per il progetto, si consideri che il numero di centroidi sia pari a N=8.

NOTE ULTERIORI SULLA SPECIFICA

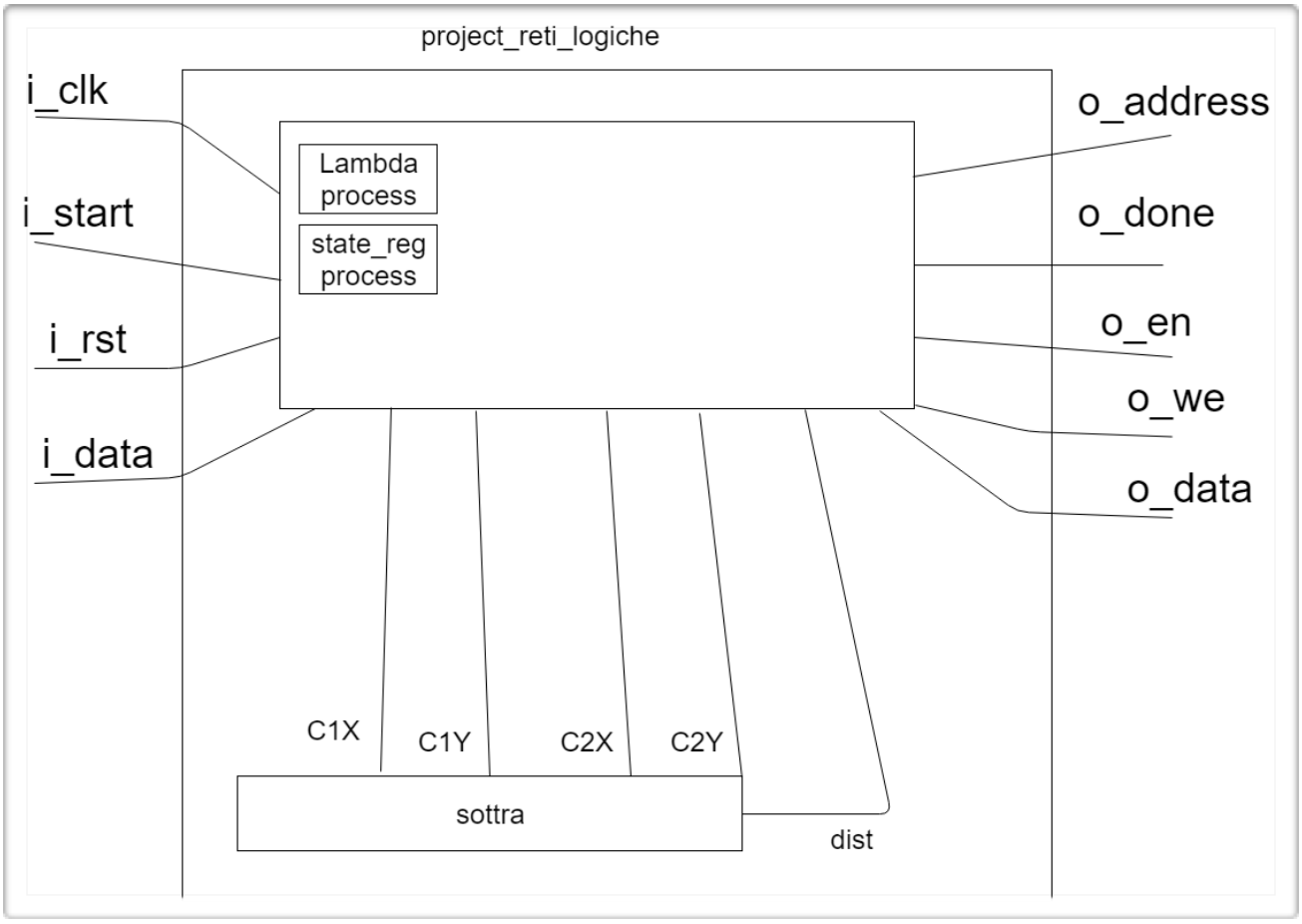
1. I centroidi nella maschera vengono elencati dal bit meno significativo - posizione più a destra - (Centroide 1) al più significativo (Centroide 8);

2. Il valore della maschera risultante dall'identificazione del/dei centroide/i più vicino/i segue la stessa regola vista al punto precedente. Il bit meno significativo rappresenta il Centroide 1 mentre quello più significativo il Centroide 8;

3. 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.

2 Design

2.1 : SCHEMA FUNZIONALE



2.2 : COMPONENTE DIFFERENZA

```
entity sottra is
    Port ( c1x : in std_logic_vector (7 downto 0);
           c2x : in STD_LOGIC_VECTOR (7 downto 0);
           c1y : in STD_LOGIC_VECTOR (7 downto 0);
           c2y : in STD_LOGIC_VECTOR (7 downto 0);
           dist : out STD_LOGIC_VECTOR (8 downto 0)
           );
end sottra;
```

Questo componente effettua la distanza di Manhattan prendendo in input le coordinate di due punti bidimensionali e restituisce il valore della distanza in binario naturale.

-c1x e c1y sono le coordinate del primo punto ascissa e ordinata rispettivamente.
-c2x e c2y sono le coordinate del secondo punto.
-dist è il risultato della distanza di manhattan.

In formule l'operazione effettuata è la seguente:
dist = |c1x - c2x| + |c1y - c2y|

2.3 : DESCRIZIONE GENERALE COMPORTAMENTO COMPONENTE PRINCIPALE

Il componente principale ha i seguenti process:

-state_reg : che gestisce la commutazione dello stato e di altri segnali sul fronte di salita del clock e gestisce anche il reset asincrono della macchina.

-lambda : che esegue un comportamento diverso della macchina a seconda dello stato in cui si trova e modifica lo stato una volta eseguite tutte le operazioni necessarie.

Il componente ha all'interno una macchina a stati finiti che ne cambia il comportamento a seconda dello stato in cui si trova la macchina.
I diversi comportamenti implementano i vari passi dell'algoritmo risolutivo.
I passi dell'algoritmo sono i seguenti:
1) La macchina legge la maschera d'input dalla memoria e anche le coordinate del punto da considerare per il calcolo dei centroidi.
2) Viene preso un bit della maschera(Si parte da quello meno significativo)
3) Se il bit è a 1 la macchina legge dalla memoria le coordinate del centroide che corrispondono a quel bit in quella posizione della maschera d'input.
4) Il componente differenza calcola la distanza di manhattan.
5) La distanza viene confrontata con il valore minimo della distanza calcolato fino a quel momento (All'inizio questo valore viene messo a 511) e ci sono 3 casi possibili:

-1 Se distanza < valore minimo : distanza diventa il nuovo valore minimo la maschera d'output viene azzerata e si mette a 1 il bit che corrisponde al centroide con cui si è calcolata la distanza.

-2 distanza > valore minimo : valore minimo e maschera d'output non cambiano.

-3 distanza = valore minimo : valore minimo non cambia ma il bit della maschera d'output che corrisponde al centroide con cui si è calcolata la distanza viene messo a 1.

6 Si passa al prossimo bit della maschera d'input e tutto il procedimento nei punti 3,4,5 viene ripetuto per tutti i bit che sono 1.
7 Alla fine dopo che si è considerato anche l'ultimo bit della maschera d'input

la macchina scrive la maschera d'output in memoria e pone il segnale o_done a 1.

2.4 : DESCRIZIONE MACCHINA A STATI

Qua sotto segue una descrizione abbastanza esaustiva sulla macchina a stati finiti all'interno del componente con tutti i suoi stati e come evolve a seconda degli ingressi.

La macchina a stati finiti parte dallo stato SR nel quale se i_start è uguale ad 1 assegna 0 alla porta o_done e assegna S0 come stato successivo .

La macchina si posizione quindi nello stato S0 nel quale se i_start è uguale ad 1 avviene la preparazione della lettura della maschera assegnando 1 alla porta o_en, 0 alla porta o_we, std_logic_vector(to_unsigned(0 , 16)) alla porta o_address per indicare la maschera di input e salviamo il valore di i_data nel signal maschera,infine assegna SK0 come stato successivo.

La macchina si posizione quindi nello stato SK0 nel quale se i_start è uguale ad 1 ripete le operazioni effettuate in S0 e infine assegna S1 come stato successivo.

La macchina si posizione quindi nello stato S1 nel quale se i_start è uguale ad 1 avviene la lettura maschera e preparazione lettura x0 assegnando std_logic_vector(to_unsigned(17 , 16)) alla porta o_address per indicare l'ascissa del punto dal quale calcolare la distanza e salviamo il valore di i_data nel signal x0,infine assegna SK1 come stato successivo.

La macchina si posizione quindi nello stato SK1 nel quale se i_start è uguale ad 1 ripete le operazioni effettuate in S1 e infine assegna S2 come stato successivo.

La macchina si posizione quindi nello stato S2 nel quale se i_start è uguale ad 1 avviene la lettura maschera e preparazione lettura y0 assegnando

std_logic_vector(to_unsigned(18 , 16)) alla porta o_address per indicare l'ordinata del punto dal quale calcolare la distanza e salviamo il valore di i_data nel signal y0,infine assegna SK2 come stato successivo.

La macchina si posizione quindi nello stato SK2 nel quale se i_start è uguale ad 1 ripete le operazioni effettuate in S2 e infine assegna S3 come stato successivo.

La macchina si posizione quindi nello stato S3 nel quale se i_start è uguale ad 1 e a seconda del valore del signal cont e della nmaschera decide se assegnare std_logic_vector(to_unsigned(valore , 16)) , con il valore che può essere 1,3,5,7,9,11,13 o 15 , alla porta o_address per indicare l'ascissa del centroide e salviamo il valore di i_data nel signal x1 ,infine assegna SK3 come stato successivo.

La macchina si posizione quindi nello stato SK3 nel quale se i_start è uguale ad 1 ripete le operazioni effettuate in S3 e infine assegna S4 come stato successivo.

La macchina si posizione quindi nello stato S4 nel quale se i_start è uguale ad 1 e a seconda del valore del signal cont e della nmaschera decide se assegnare std_logic_vector(to_unsigned(valore , 16)) , con il valore che può essere 2,4,6,8,10,12,14 o 16 , alla porta o_address per indicare l'ordinata del centroide e salviamo il valore di i_data nel signal y1 ,infine assegna SK4 come stato successivo.

La macchina si posizione quindi nello stato SK4 nel quale se i_start è uguale ad 1 ripete le operazioni effettuate in S4 e infine assegna S5 come stato successivo.

La macchina si posizione quindi nello stato S5 nel quale a seconda se del confronto tra i signal distanza e MinDistanza si modifica il next_state , se distanza>MinDistanza assegna S6 a next_state, se distanza=MinDistanza

assegna SA a next_state se distanza<MinDistanza assegna SB a next_state ,infine assegna S6 come stato successivo.

La macchina si posizione quindi nello stato SA nel quale a seconda del valore di cont cambia la posizione in cui aggiunge un bit sulla mascheraOut

La macchina si posizione quindi nello stato SB nel quale a seconda del valore di cont cambia il valore con cui resetta mascheraOut dopodichè assegna l'attuale distanza come nuova MinDistanza ,infine assegna S6 come stato successivo.

La macchina si posizione quindi nello stato S6 nel quale a seconda del valore di cont assegna uno stato tra SD , SE, SF, SG, SH, SI, SJ, S7 come stato successivo.

La macchina si posizione quindi nello stato SD nel quale assegna il valore 00000001 a cont e assegna S3 come stato successivo.

La macchina si posizione quindi nello stato SE nel quale assegna il valore 00000010 a cont e assegna S3 come stato successivo.

La macchina si posizione quindi nello stato SF nel quale assegna il valore 00000100 a cont e assegna S3 come stato successivo.

La macchina si posizione quindi nello stato SG nel quale assegna il valore 00001000 a cont e assegna S3 come stato successivo.

La macchina si posizione quindi nello stato SH nel quale assegna il valore 00010000 a cont e assegna S3 come stato successivo.

La macchina si posizione quindi nello stato SI nel quale assegna il valore 00100000 a cont e assegna S3 come stato successivo.

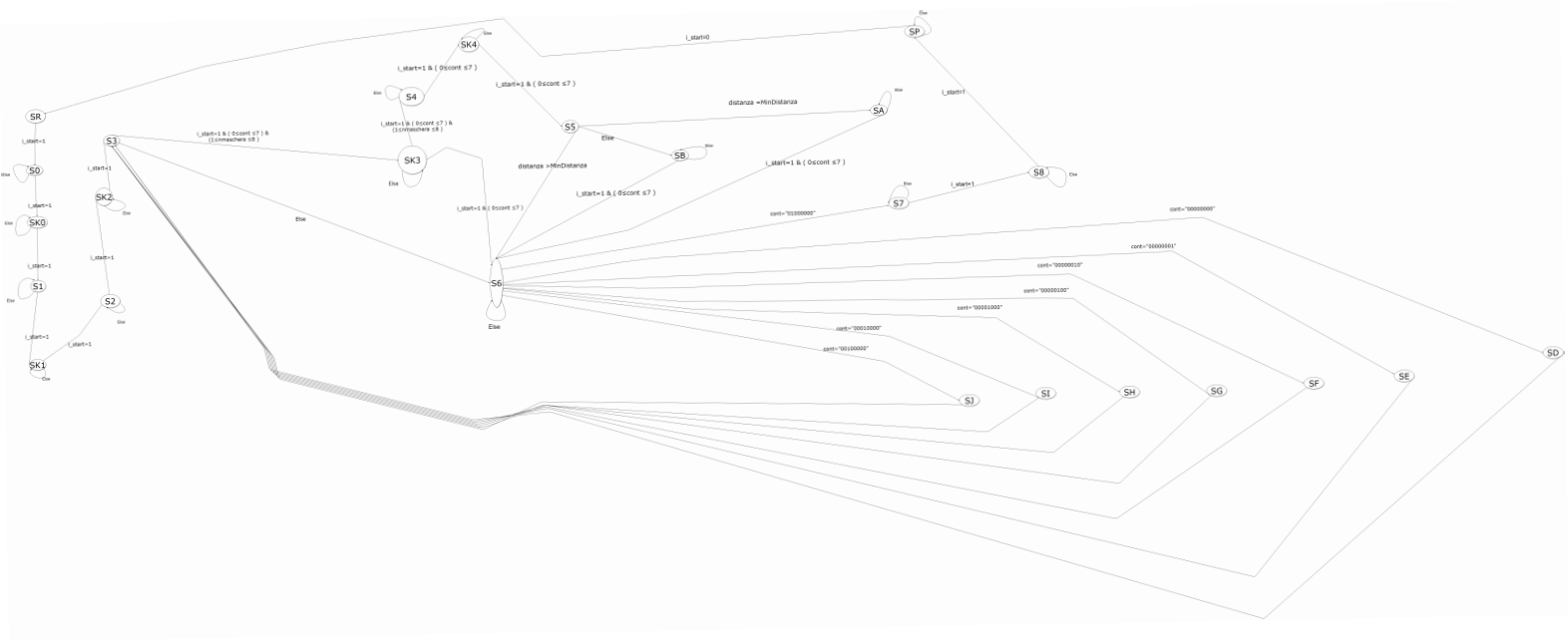
La macchina si posizione quindi nello stato SJ nel quale assegna il valore 01000000 a cont e assegna S3 come stato successivo.

La macchina si posizione quindi nello stato S7 nel quale se i_start è uguale ad 1 assegna il valore 1 alla porta o_we, std_logic_vector(to_unsigned(19 , 16)) a o_address e mascheraOut in o_data al fine di scrivere la maschera di output e infine assegna S8 come stato successivo.

La macchina si posizione quindi nello stato S8 nel quale se i_start è uguale ad 1 assegna il valore 1 a o_done e infine assegna SP come stato successivo

La macchina si posizione quindi nello stato SP nel quale assegna il valore 0 alla porta o_en e alla porta o_done e infine assegna SR come stato successivo.

2.5 : DIAGRAMMA MACCHINA A STATI



Current state	Next state														
	i_start=1	i_start=0	i_start=1 & (0≤cont ≤7) & (1≤nmaschera ≤8)	i_start=1 & (0≤cont ≤7)	Distanza = mindistanza	Distanza > mindistanza	Cont = 0	Cont = 1	Cont = 2	Cont = 3	Cont = 4	Cont = 5	Cont = 5	Cont = 7	else
SR	S0	X	X	X	X	X	X	X	X	X	X	X	X	X	SR
S0	SK0	X	X	X	X	X	X	X	X	X	X	X	X	X	S0
SK0	S1	X	X	X	X	X	X	X	X	X	X	X	X	X	SK0
S1	SK1	X	X	X	X	X	X	X	X	X	X	X	X	X	S1
SK1	S2	X	X	X	X	X	X	X	X	X	X	X	X	X	SK1
S2	SK2	X	X	X	X	X	X	X	X	X	X	X	X	X	S2
SK2	S4	X	X	X	X	X	X	X	X	X	X	X	X	X	SK2
S4	X	X	SK4	X	X	X	X	X	X	X	X	X	X	X	S8
SK4	X	X	S5	S8	X	X	X	X	X	X	X	X	X	X	SK4
S5	X	X	X	SK5	X	X	X	X	X	X	X	X	X	X	S5
SK5	X	X	X	S6	X	X	X	X	X	X	X	X	X	X	SK5
S6	X	X	X	X	SA	S8	X	X	X	X	X	X	X	X	SB
SA	X	X	X	S8	X	X	X	X	X	X	X	X	X	X	SA
SB	X	X	X	S8	X	X	X	X	X	X	X	X	X	X	SB
S8	X	X	X	X	X	X	SD	SE	SF	SG	SH	SI	SJ	S28	S8
SD	S4	S4	S4	S4	S4	S4	S4	S4	S4	S4	S4	S4	S4	S4	S4
SE	S4	S4	S4	S4	S4	S4	S4	S4	S4	S4	S4	S4	S4	S4	S4
SF	S4	S4	S4	S4	S4	S4	S4	S4	S4	S4	S4	S4	S4	S4	S4
SG	S4	S4	S4	S4	S4	S4	S4	S4	S4	S4	S4	S4	S4	S4	S4
SH	S4	S4	S4	S4	S4	S4	S4	S4	S4	S4	S4	S4	S4	S4	S4
SI	S4	S4	S4	S4	S4	S4	S4	S4	S4	S4	S4	S4	S4	S4	S4
SJ	S4	S4	S4	S4	S4	S4	S4	S4	S4	S4	S4	S4	S4	S4	S4
S28	S29	X	X	X	X	X	X	X	X	X	X	X	X	X	S28
S29	SP	X	X	X	X	X	X	X	X	X	X	X	X	X	S29
SP	X	SR	X	X	X	X	X	X	X	X	X	X	X	X	SP

2.6 : GIUSTIFICAZIONE SCELTE IMPLEMENTATIVE

Abbiamo deciso di strutturare la macchina a stati finiti in questo modo al fine di non dover calcolare anche le distanze di Manhattan dei centroidi che non andavano considerati in base all'osservazione della maschera di input.

Abbiamo inoltre scelto di effettuare la lettura del valore di i_data nello stesso stato della lettura di o_address perché tra tutte le diverse soluzioni che abbiamo considerato questa era l'unica che permetteva il funzionamento dei test legati alla timing.

3 Testing

3.1 : REPORT DI SINTESI

State	New Encoding	Previous Encoding
sr	00000000000000000000000001	00000
s0	00000000000000000000000010	00001
sk0	00000000000000000000000100	00010
s1	00000000000000000000001000	00011
sk1	00000000000000000000010000	00100
s2	00000000000000000000100000	00101
sk2	00000000000000000001000000	00110
s3	00000000000000000010000000	00111
sk3	00000000000000000100000000	01000
s4	00000000000000001000000000	01001
sk4	00000000000000010000000000	01010
s5	00000000000000100000000000	01011
sa	00000000000001000000000000	01100
sb	00000000000010000000000000	01101
s6	00000000000100000000000000	01110
sd	00000000001000000000000000	01111
se	00000000010000000000000000	10000
sf	00000001000000000000000000	10001
sg	00000010000000000000000000	10010
sh	00000100000000000000000000	10011
si	00001000000000000000000000	10100
sj	00010000000000000000000000	10101
s7	00100000000000000000000000	10110
s8	01000000000000000000000000	10111
sp	10000000000000000000000000	11000

```
Report Cell Usage:
+------+-------+------+
|      |Cell   |Count |
+------+-------+------+
|1     |BUFG   |     1|
|2     |CARRY4 |     8|
|3     |LUT1   |     8|
|4     |LUT2   |    45|
|5     |LUT3   |    53|
|6     |LUT4   |    61|
|7     |LUT5   |    58|
|8     |LUT6   |   154|
|9     |FDCE   |    24|
|10    |FDPE   |     1|
|11    |FDRE   |    64|
|12    |IBUF   |    11|
|13    |OBUF   |    27|
+------+-------+------+

Report Instance Areas:
+------+----------------------+-------+------+
|      |Instance              |Module |Cells |
+------+----------------------+-------+------+
|1     |top                   |       |   515|
|2     |  CalcolatoreDistanza |sottra |   148|
+------+----------------------+-------+------+
---------------------------------------------------------------------------------
Finished Writing Synthesis Report : Time (s): cpu = 00:00:28 ; elapsed = 00:00:30 . Memory (MB): peak = 696.250 ; gain = 435.551
```

3.2 : CASI DI TEST

Il componente è stato testato con diversi casi di test alcuni andavano a generare dei casi particolari.

In particolare i test vanno a testare i casi :

-Distanze molto grandi : Per testare resistenza a eventuali overflow.

-Centroidi uguali.

-Centroidi Equidistanti.

-Maschera d'input tutta a 0.

-Maschera d'input tutta a 1.

-Centroidi con coordinate identiche.

-Centroidi al bordo.

Il resto dei test riguardano casi differenti e sono stati generati casualmente.

3.3 : RISULTATI

Il componente supera correttamente tutti i casi di test con la simulazione Behavioral, la simulazione Post-Syntesis Functional e Timing (con periodo di clock di 100 ns).

Abbiamo effettuato ulteriori test in Post-Syntesis Timing con valori di clock diversi e abbiamo scoperto che il componente può raggiungere una velocità di 100MHz (clock = 10 ns).