

RELAZIONE SUL PROGETTO DI ELETTRONICA DIGITALE

A.A. 2022-2023

Jhenalyn Subol

mat. 213485



Table of Contents

01

Traccia del
progetto

02

Schema a
blocchi

03

VHDL Code : Design
Source (Circ.vhd,
supporto.vhd,
my_clock)

04

Elaborated
Design (RTL
Schematic)

05

Test Bench +
Behavioral
Simulation
 $n=8\text{bit}$

06

Post-Synthesis
Timing
Simulation
 $n=8\text{bit}$

07

Post-
Implementation
Timing
Simulation
 $n=8\text{bit}$

08

Caratterizzazione
della struttura
progettata $n=8\text{bit}$

09

Ulteriori Test:
 $n=16\text{ bit};$
 $n= 32\text{ bit}$

Traccia del Progetto

Si richiede di descrivere un circuito che riceve in ingresso due numeri in complemento a 2, A e B ad n-bit, e due segnali di controllo c1 e c0, in base ai quali si deve stabilire l'operazione da eseguire:

- se $c1=c0=0 \rightarrow A+B;$
- se $c1=0 \text{ e } c0=1 \rightarrow A-B;$
- se $c1=1 \text{ e } c0=0 \rightarrow -A+B;$
- se $c1=c0=1 \rightarrow -A-B.$

Il circuito dev'essere pipeline e deve impiegare registri sensibili ai fronti di salita del segnale di clock.

Si richiede, oltre alla verifica funzionale della struttura progettata, la sua caratterizzazione (valutazione di risorse occupate, massima frequenza di funzionamento, latenza e potenza dissipata) per n=8, 16 e 32.

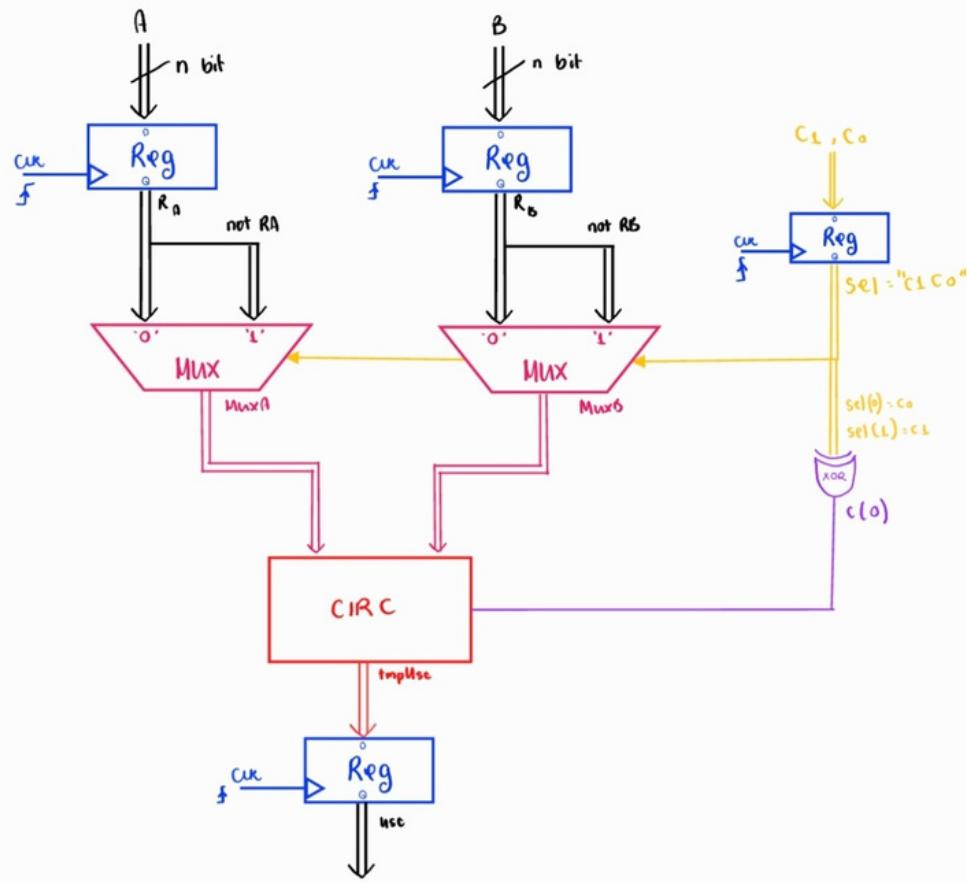
01

Schema a blocchi

Per poter comprendere facilmente la progettazione del circuito, andiamo a rappresentare la sua architettura in uno schema a blocchi costituito da seguenti componenti:

- 4 registri -> 2 dei quali servono per ricevere l'input A e l'input B (rappresentati in complimenti a 2), un registro riceve i 2 bit di controllo c1 e c0 e un registro serve per l'uscita. Un registro è realizzato mettendo insieme un gruppo di flipflop (1 per bit) sincronizzati mediante un ingresso di clock in comune. In questo circuito, il dato presente sull'ingresso del registro viene trasferito all'uscita ad ogni fronte di salita del segnale di clock (rising-edge).

Proposed Circuit implementation:



- *2 multiplexer a due ingressi, 1 selettore e 1 uscita* -> utilizzati per selezionare tra l'input diretto e l'input negato.
 - *Blocco Circ* - rappresenta l'unità che processa i dati ed esegue le operazioni necessari in base ai valori di c_0 e c_1 .
 - *Porta XOR* - questa porta opera tra i due bit di controllo c_1 e c_0 e il risultato sarà il valore del primo riporto da mandare dentro il blocco Circ. Questo valore quindi mi servirà per effettuare l'operazione di sottrazione.

VHDL Code - Design Source

Entity

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
library work;
use work.myconst.all;

entity Circ is
    Generic (n : integer := nbitin);
    Port ( A : in STD_LOGIC_VECTOR (n-1 downto 0);
            B : in STD_LOGIC_VECTOR (n-1 downto 0);
            c1 : in STD_LOGIC;
            c0 : in STD_LOGIC;
            clk : in STD_LOGIC;
            usc : out STD_LOGIC_VECTOR (n downto 0));
end Circ;
```

Architecture (...)

```
architecture Behavioral of Circ is
    signal RA, RB, tmpUsc : std_logic_vector(n downto 0);
    signal p, g, MuxA, MuxB : std_logic_vector(n downto 0);
    signal c : std_logic_vector (n+1 downto 0);
    signal one : std_logic_vector(n downto 0) := (0=>'1', others=> '0');
    signal sel : std_logic_vector (1 downto 0);

begin
    Pclk: process(clk) begin
        if rising_edge(clk) then
            RA <= A(n-1)&A;
            RB <= B(n-1)&B;
            sel <= c1&c0;
            usc <= tmpUsc;
        end if;
    end process;
```

Clock Logic : In questo pezzo del codice, tutti i processi cronometrati (clocked processes) vengono innescati simultaneamente e vengono letti una volta il segnale di clock è attivo. Nello stesso tempo è disponibile all'uscita il risultato dell'iterazione precedente. Pertanto, il risultato della corrente iterazione viene tenuto stabile fino al prossimo rising-edge signal del clock.

VHDL Code - Design Source

(...)Architecture

```
Pinputs: process(RA,RB,sel)
begin
    if sel = "00" then
        MuxA <= RA;
        MuxB <= RB;
    elsif sel = "01" then
        MuxA <= RA;
        MuxB <= not RB;
    elsif sel = "10" then
        MuxA <= not RA;
        MuxB <= RB;
    else --case '11'
        MuxA <= RA;
        MuxB <= RB;
    end if;
end process;
```

Process statement degli input : Per poter effettuare l'operazione di sottrazione, andremo a sfruttare questa logica: $A - B = A + \text{not}B + 1$ (*)
Nei primi 3 casi, quando il bit di controllo è 1, verrà scelto l'input negato, altrimenti l'input diretto.

Nell'ultimo caso in cui i bit di controllo sono entrambi 1 ($c1=1, c0=1$) ho pensato di procedere usando il seguente ragionamento:

$$-A-B=-(A+B) = \text{not}(A+B) + 1$$

Quindi, calcolo prima la somma di due input e trasformare il risultato alla rappresentazione in complementi a 2,

```
p <= MuxA xor MuxB;
g <= MuxA and MuxB;
c(n+1 downto 1) <= g or (p and c(n downto 0));
c(0) <= sel(0) xor sel(1);

tmpUsc <= (not(p xor c(n downto 0)) + one)
when (sel(0)='1' and sel(1)='1') else
(p xor c(n downto 0));
```

```
end Behavioral;
```

Fuori il process, vengono eseguiti in modo concorrente l'operazione di somma.

In questa parte del codice, andiamo a sfruttare i segnali di "propagate" e "generate" per calcolare la somma $A+B$ e il riporto.

Il primo bit del riporto è dato dallo XOR di $c1$ e $c0$. Quando sono entrambi 0, posso fare tranquillamente la somma. Quando uno di questi bit è 1 (case 2 e case 3), posso sfruttare la formula (*) per eseguire l'operazione di sottrazione in quanto lo XOR di bit diversi dà 1. Mentre nell'ultimo caso, entrambi i bit sono 1 quindi lo XOR è 0. Posso eseguire semplicemente la somma e successivamente nel risultato vado ad applicare $\text{not}(A+B)+1$.

VHDL Code - Design Source

File di supporto

```
1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3
4 package myconst is
5     constant nbitin: integer := 8; -- 16, 32
6 end package myconst;
7
8 package myfunc is
9     function pow2(nbit : in integer) return integer;
10 end package myfunc;
11
12 package body myfunc is
13     function pow2(nbit: in integer) return integer is
14         variable i: integer;
15         variable pown: integer;
16     begin
17         pown := 1;
18         for i in 0 to nbit loop
19             exit when (i = nbit);
20             pown := pown*2;
21         end loop;
22         return pown;
23     end function pow2;
24 end package body myfunc;
```



Questo file di supporto preso dalla libreria work è costituita dai package che voglio importare dentro il mio VHDL file.

Con il package myconst, posso modificare direttamente il numero di bit dei miei input cambiando il valore della costante nbitin.

Il package myfunc mi servirà per effettuare una simulazione esaustiva dentro il Test Bench.

File di constraint

```
create_clock -period 10.000 -name myclock -waveform {0.000 5.000} [get_nets clk]
```

Per fare che il nostro circuito lavori ad una certa frequenza, introduciamo un constraint ovvero un limite sul segnale di clock.

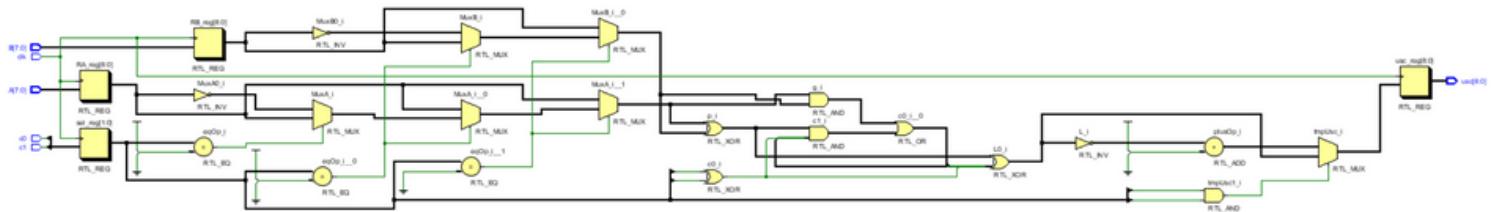
Nel nostro process scritto nel Test Bench, sono nascosti 2 parametri del segnale di clock:

- Frequenza - Il ciclo di clock dura 10ns ovvero 100MHz.
- Duty Cycle di 50% perché ho distribuito il periodo di clock 50-50 tra lo stato alto e lo stato basso ($T_{clk}/2$)

Nel file di constraint che chiamiamo my_clock, specifichiamo l'elemento reale nel nostro circuito che rappresenta il clock (clk) e al quale applichiamo il constraint. In questo file, ho settato il periodo in 10ns e di default, il clock viene scelto con un duty cycle di 50%.

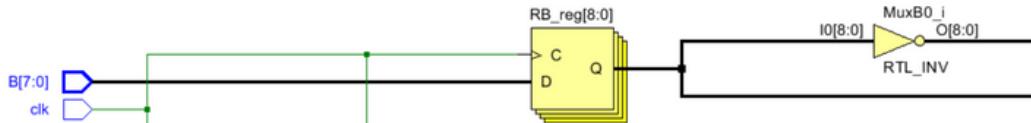
Elaborated Design (RTL Schematic)

Schematic Design del circuito

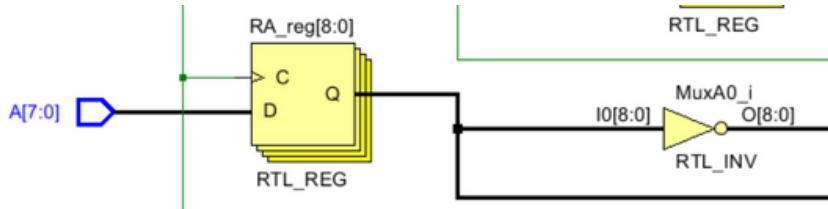


Zoom dei componenti

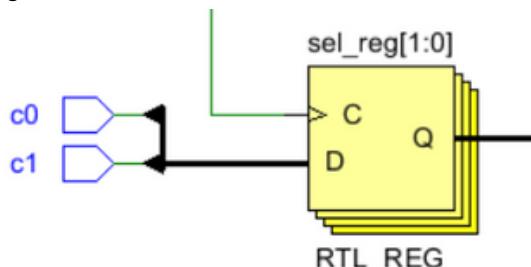
Registro di B



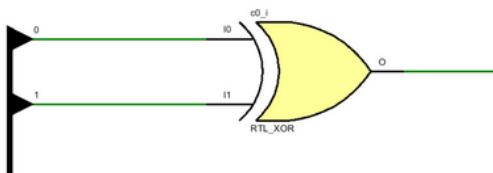
Registro di A



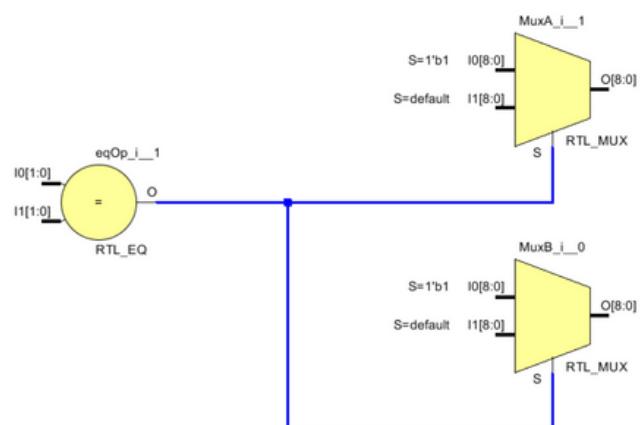
Registro dei bit di controllo c1 e c0



Porta XOR che opera tra sel(1)=c1 e sel(0) = c0 il cui output sarà c(0)

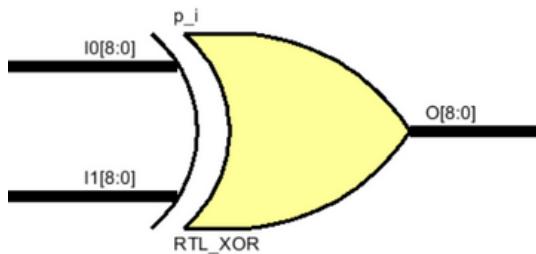


Multiplexer i cui selettori sono sel(1) per Mux A e sel(0) per Mux B



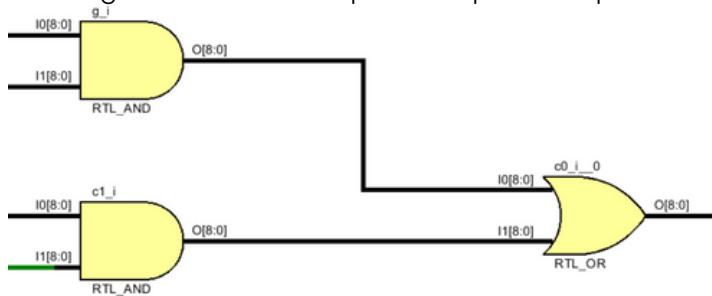
Elaborated Design

Porta XOR che calcola il segnale di propagate

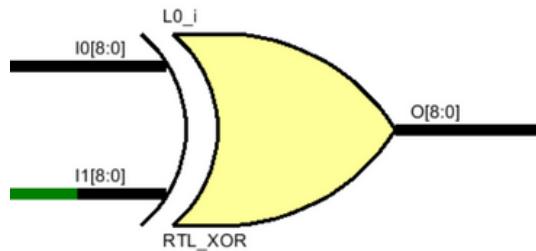


La prima porta AND calcola il segnale di generate

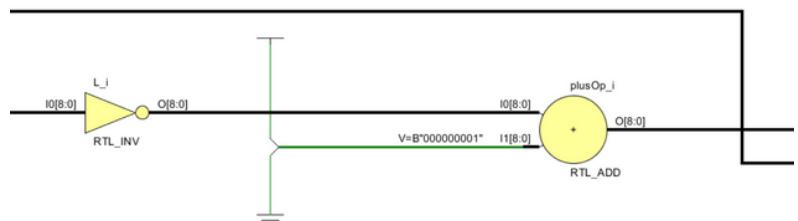
La seconda porta AND viene eseguito al segnale propagate e i bit da 0 ad n del riporto
I due segnali diventano l'input della porta OR per calcolare i bit da 1 ad n+1 del riporto.



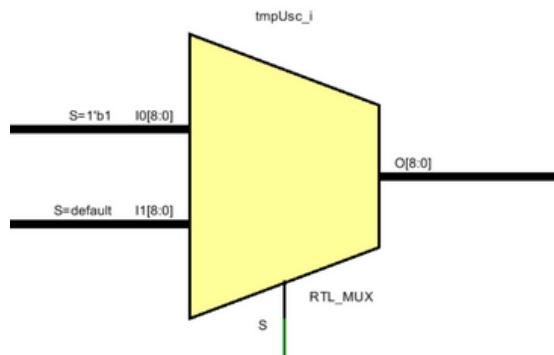
Porta XOR che calcola la somma



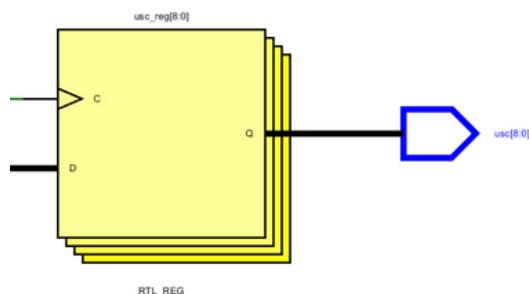
Quando siamo nel caso $c_i=1, c_0=1$, trasformiamo la somma in complemento a 2



Multiplexer che sceglie la somma finale



Registro dell'output finale



Jhenalyn Subol
matr 213485

04

Test Bench + Behavioral Simulation n=8bit

Caso 1:

- n=8 bit;
- c1=0, c0=0 -> A+B ;
- range tra -(2^(n-1)) e (2^(n-1))

Architecture

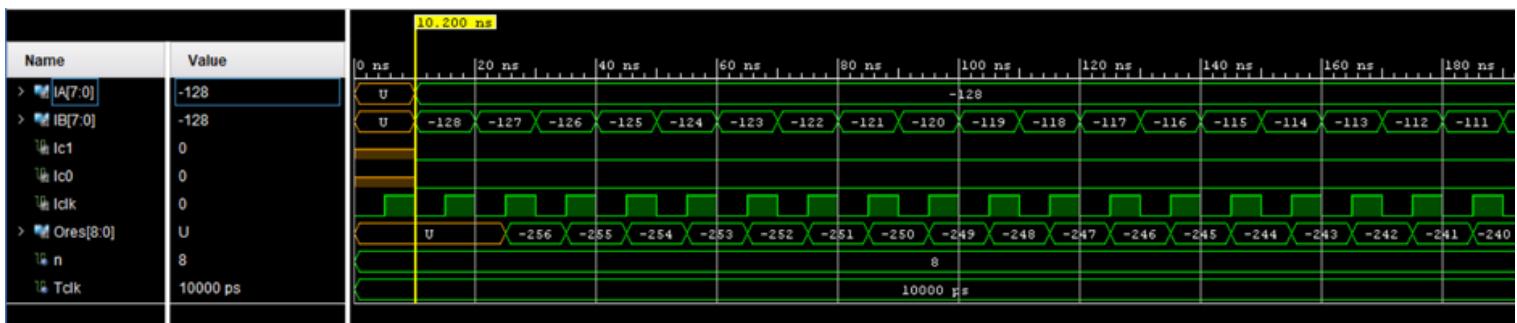
```

architecture MySim of TestBench is
component Circ is
  Generic (n : integer := nbitin);
  Port ( A : in STD_LOGIC_VECTOR (n-1 downto 0);
         B : in STD_LOGIC_VECTOR (n-1 downto 0);
         cl : in STD_LOGIC;
         c0 : in STD_LOGIC;
         clk : in STD_LOGIC;
         usc : out STD_LOGIC_VECTOR (n downto 0));
end component;

signal IA, IB : std_logic_vector(n-1 downto 0);
signal Icl, Ic0 : std_logic;
signal Iclk : std_logic := '0';
signal Ores : std_logic_vector(n downto 0);
constant Tclk : time := 10ns;
begin
  Pclk: process begin --andamento temporale del clock
    wait for Tclk/2; -- duty cycle 50%
    Iclk <= not Iclk;
  end process;

  stim_process : process begin --stimulus process
    wait for Tclk;
    Icl <= '0'; Ic0 <= '0'; -- A+B
    gi: for i in -(2**n) to ((2**n)-1) loop
      IA <= conv_std_logic_vector(i,n);
    gj: for j in -(2**n) to ((2**n)-1) loop
      IB <= conv_std_logic_vector(j,n);
      wait for Tclk;
    end loop;
    end loop;
  end process;
  CUT: Circ port map( IA, IB, Icl, Ic0, Iclk, Ores);
end MySim;

```



Behavioral Simulation waveform

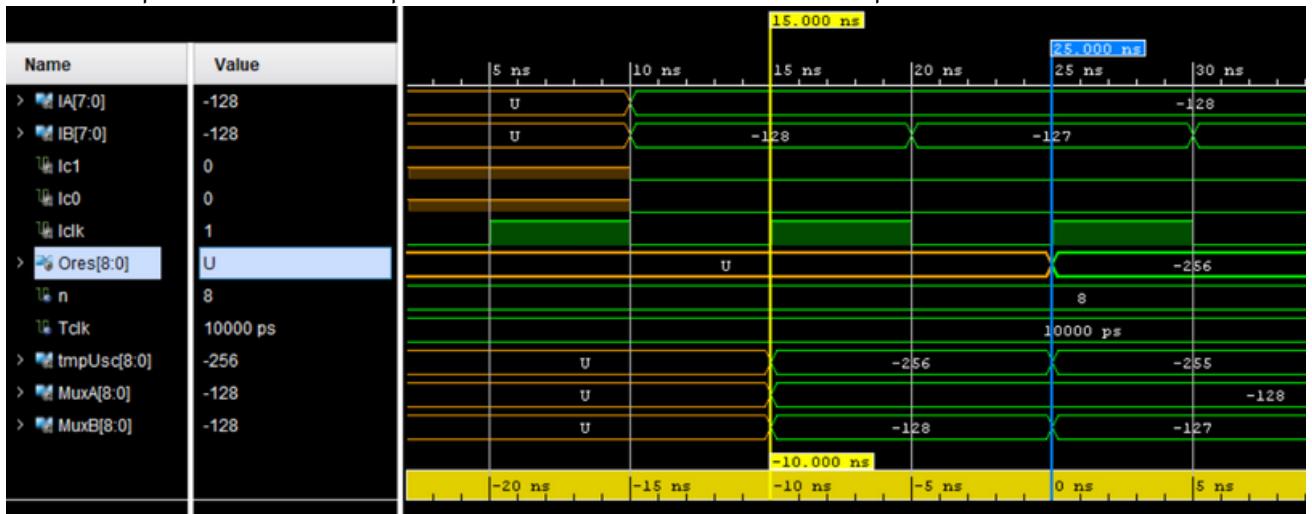
Test Bench + Behavioral Simulation n=8bit

Osservazioni:

- Dopo l'attesa di 10 ns(*fig1.* cursore giallo), il primo fronte utile(cursore blu) campiona IA=-128 e IB=-128 e li invia in ingresso ai registri che riproducono RA e RB. Durante questo istante di tempo, l'uscita prodotta del circuito è inizialmente non specificata(uninitialized).



- Se andiamo ad analizzare i segnali interni del circuito, osserviamo appunto che al tempo t=15ns(*fig2.* cursore giallo), vengono ricevuti MuxA e MuxB mentre il registro in uscita non ha ancora ricevuto alcun dato valido dal blocco che effettua il calcolo pertanto quello che viene campionato e reso disponibile in uscita è un valore non specificato.



- Al successivo fronte di clock utile (*fig2.* cursore blu), viene campionata la seconda coppia di dati e contemporaneamente viene campionato il risultato del calcolo precedente. Questa sorta di disallineamento viene chiamata *LATENZA DEL CIRCUITO*.
- Osserviamo che i dati d'ingresso e l'uscita sono effettivamente sincronizzati con il segnale del clock.

Test Bench + Behavioral Simulation n=8bit

Caso 2:

- n=8 bit;
- c1=0, c0=1 -> A-B ;
- range tra $-(2^{n-1})$ e $(2^{n-1})-1$

Architecture

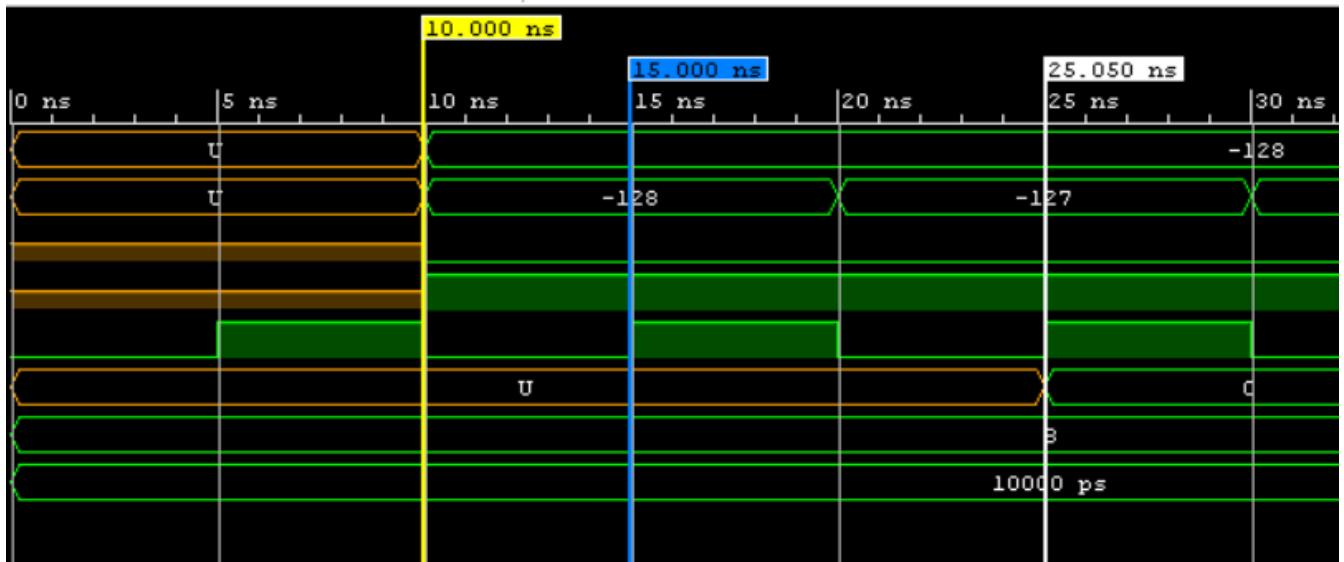
```

stim_process : process begin      --stimulus process
    wait for Tclk ;
    Icl <= '0'; Ic0 <= '1'; --   A-B
    gi: for i in -(2**n-1) to ((2**n-1)-1) loop
        IA <= conv_std_logic_vector(i,n);
    gj: for j in -(2**n-1) to ((2**n-1)-1) loop
        IB <= conv_std_logic_vector(j,n);
        wait for Tclk;
    end loop;
    end loop;
end process;

CUT: Circ port map( IA, IB, Icl, Ic0, Iclk, Ores);

```

Behavioral Simulation



Osservazioni:

- Dopo il tempo di attesa(cursore giallo) pari a $t=T_{clk}=10\text{ns}$, nell'istante di tempo pari a $t=15\text{ns}$ il primo fronte utile (cursore blu) campiona $IA=-128$, $IB=-128$ e li invia in ingresso ai registri che riproducono RA e RB. Come abbiamo osservato nel primo test, in questo fronte, l'uscita ORes è ancora pari a '0'.
- Nell secondo fronte utile(cursore bianco), vengono campionati nuovi valori all'ingresso e contemporaneamente, ORes fornisce il risultato precedente pari a $A-B=(-128)-(-128)=-128+128=0$.

Test Bench + Behavioral Simulation n=8bit

Caso 3:

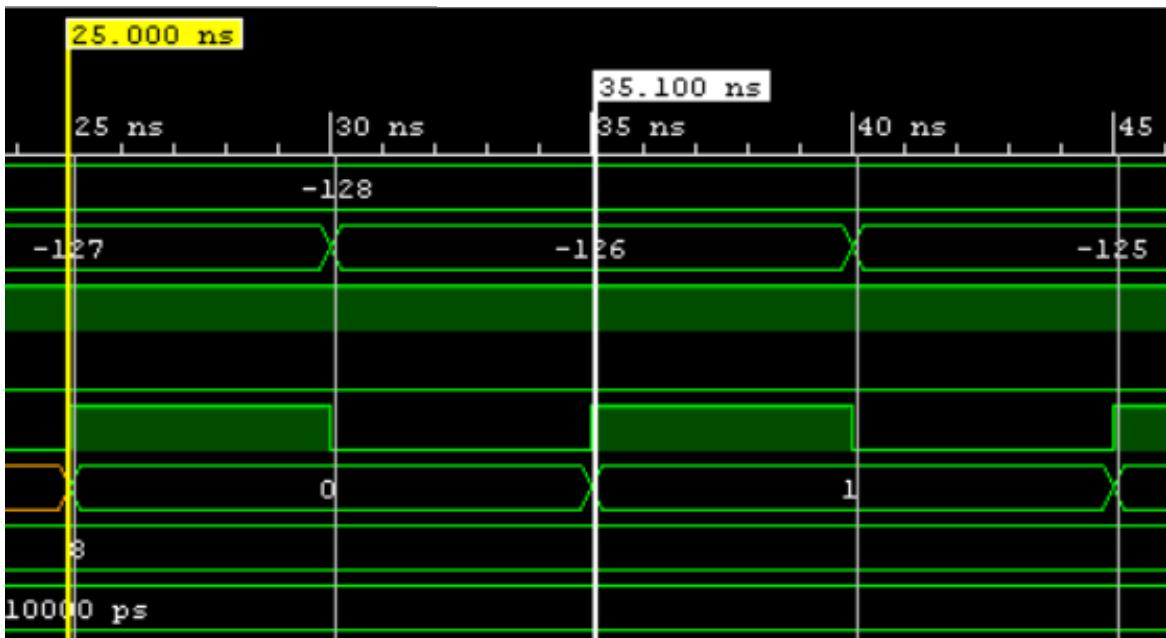
- n=8bit;
- c1=1, c0=0-> -A+B ;
- range tra $-(2^{n-1})$ e $(2^{n-1})-1$

Architecture

```
stim_process : process begin      --stimulus process
    wait for Tclk ;
    Icl <= '1'; Ic0 <= '0'; --  -A+B
    gi: for i in -(2**n-1) to ((2**n-1)-1) loop
        IA <= conv_std_logic_vector(i,n);
    gj: for j in -(2**n-1) to ((2**n-1)-1) loop
        IB <= conv_std_logic_vector(j,n);
        wait for Tclk;
    end loop;
    end loop;
end process;
```

Behavioral Simulation

```
CUT: Circ port map( IA, IB, Icl, Ic0, Iclk, Ores);
```



Osservazioni:

- Prendiamo l'istante di tempo in cui t=25ns(cursore giallo). In questo fronte vengono campionati IA=-128, IB= -126 e li invia in ingresso ai registri che riproducono RA e RB. Contemporaneamente in questo fronte, l'uscita ORes contiene il risultato del calcolo precedente.
- Nell successivo fronte utile(cursore bianco), vengono campionati nuovi valori all'ingresso e questa volta, ORes fornisce il risultato precedente pari a $-A+B = -(-128)+(-127)=128-128=1$.

Test Bench + Behavioral Simulation n=8bit

Caso 4:

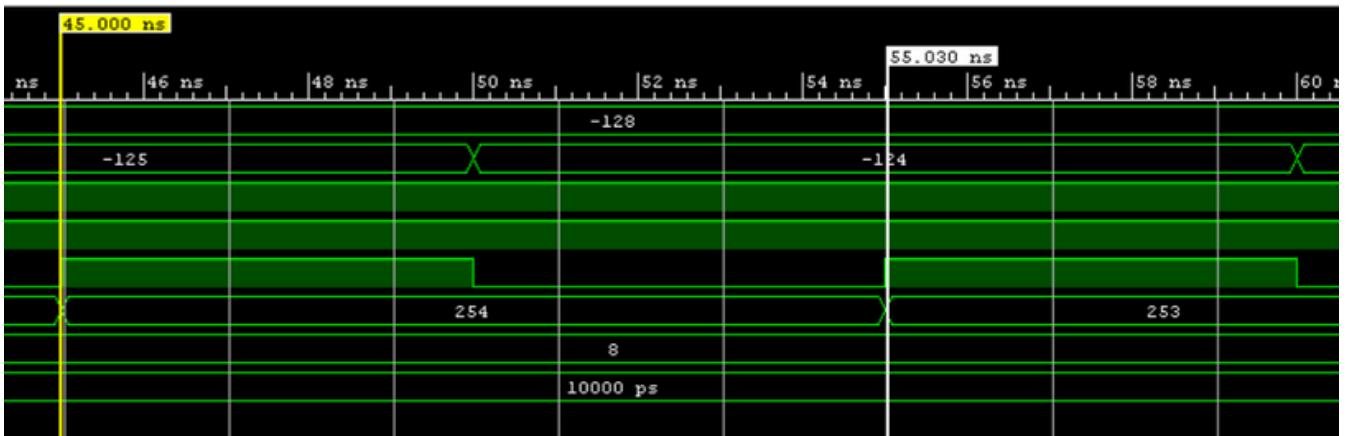
- n=8bit;
- c1=1, c0=1-> -A-B ;
- range tra $-(2^{n-1})$ e $(2^{n-1})-1$

Architecture

```
stim_process : process begin      --stimulus process
    wait for Tclk;
    Icl <= '1'; Ic0 <= '1'; -- -A-B
    gi: for i in -(2**n-1) to ((2**n-1)-1) loop
        IA <= conv_std_logic_vector(i,n);
    gj: for j in -(2**n-1) to ((2**n-1)-1) loop
        IB <= conv_std_logic_vector(j,n);
        wait for Tclk;
    end loop;
    end loop;
end process;

CUT: Circ port map( IA, IB, Icl, Ic0, Ores);
```

Behavioral Simulation



Osservazioni:

- Prendiamo l'istante di tempo in cui t=45ns(cursore giallo). In questo fronte vengono campionati IA=-128, IB= -125 e li invia in ingresso ai registri che riproducono RA e RB. Contemporaneamente in questo fronte, l'uscita ORes contiene il risultato del calcolo precedente.
- Nell successivo fronte utile(cursore bianco), vengono campionati nuovi valori all'ingresso e nello stesso tempo, ORes fornisce il risultato precedente pari a $-A-B = -(-128)-(-125)=128+125=253$.

Post-Synthesis Timing Simulation

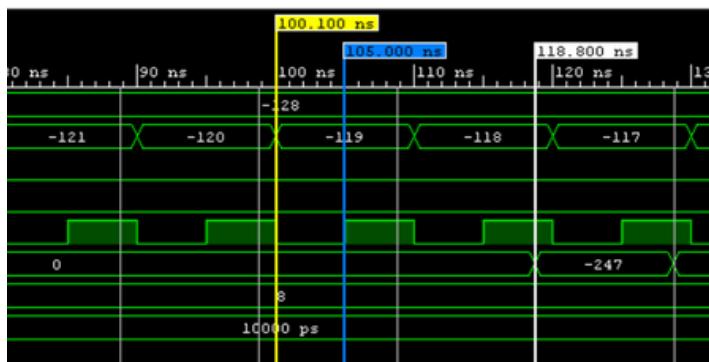
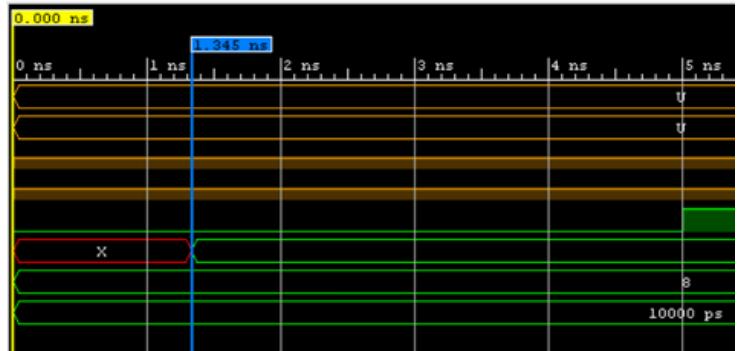
n=8bit

Attraverso la *Post-Synthesis Timing Simulation*, notiamo la presenza di una specificazione 'X' la quale indica che vi è una elaborazione degli ingressi in corso, evidenziando dunque un ritardo dovuto ai componenti.

In questa simulazione, il range che ho considerato va da $-(2^{(n-1)})$ a $(2^{(n-1)})-1$ con $n=8\text{bit}$ e ho settato i bit di controllo in $c1='0'$ e $c0='0'$. Dunque ci si aspetta che il circuito effettua l'operazione $A+B$ tra gli input.

OSSERVAZIONI:

- Dopo il *synthesis*, il ritardo iniziale dell'uscita sarà pari a 1,350ns (cursor blu) dovuto al fatto che *PST simulation* utilizza il *timing delay* stimato dalla macchina senza tenere in considerazione il ritardo di interconnessione.



- Osserviamo nel waveform della sintesi che all'inizio, si trova l'uscita ORes pari a '0'. Questo è dovuto al fatto che il simulatore XILINX simula il global reset del circuito per i primi 100ns (cursor giallo). Durante questo periodo, il circuito rimane appunto nel suo stato built in reset(configuration state).

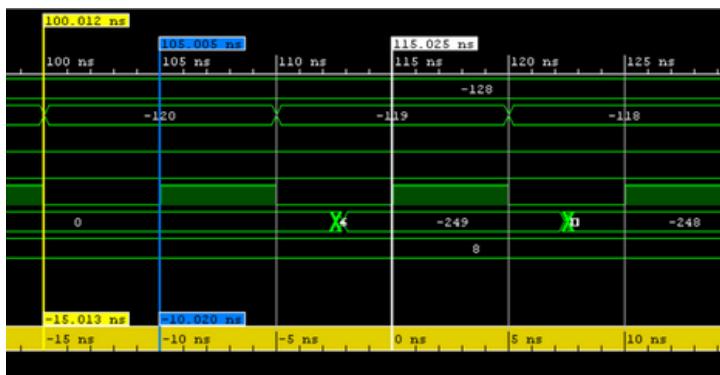
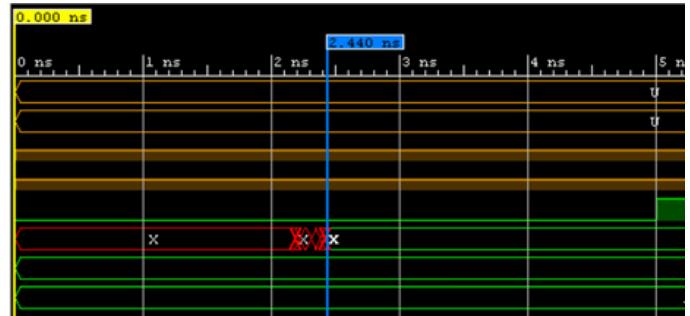
- Dopo il global reset il primo fronte utile (cursor blu) campiona IA=-128, IB= -119 e li invia in ingresso ai registri che riproducono RA e RB con un certo ritardo. Contemporaneamente, in questo fronte, l'uscita ORes è ancora pari a '0'. Nell secondo fronte utile(cursor bianco), vengono campionati nuovi valori all'ingresso e contemporaneamente, ORes fornisce il risultato precedente pari a -247.

Post- Implementation Timing Simulation

n=8bit

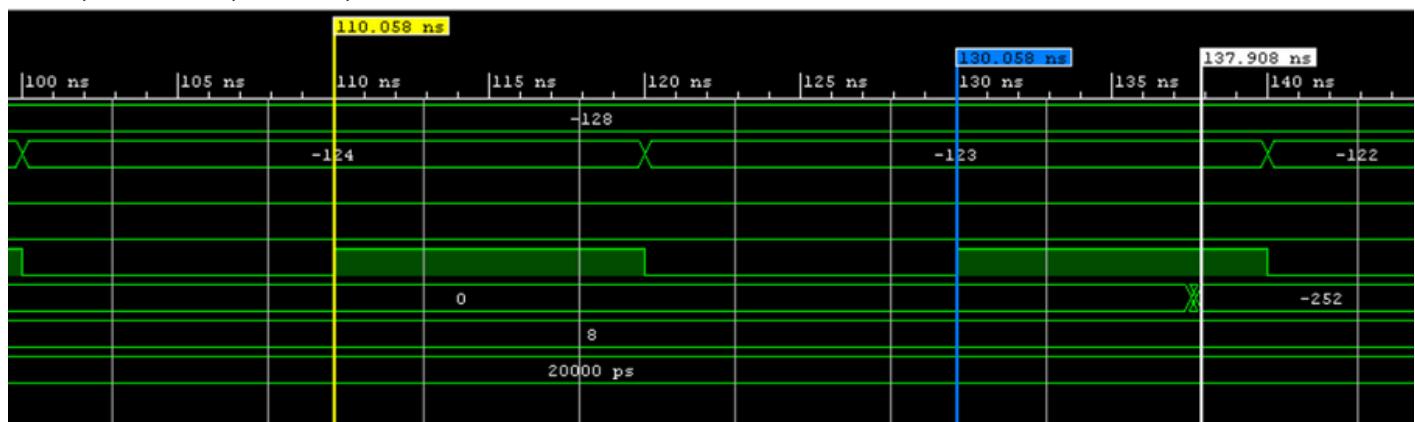
Nella Post-Implementation Timing Simulation, riconosciamo i ritardi dovuti sia alla logica che all'interconnessione.

Osserviamo infatti un ritardo iniziale nell'uscita pari a 2,550ns(cursore blu) superiore a quella della Post-Synthesis Timing Simulation, che era pari a 1,350ns.



Anche in PIT Simulation, andremo a considerare il global reset pari a 100ns(cursore giallo) in cui tutti i registri dentro il dispositivo sono bloccati. Osserviamo allora che il valore di ORes durante questo periodo di tempo è pari a '0'.

Dopo il global reset, il primo fronte utile(cursore blu) campiona IA=-128 e IB=-120 e li invia in ingresso ai registri che riproducono RA e RB con un certo ritardo. Come nella PST Simulation, in questo fronte, l'uscita ORes è ancora pari a '0'. Tuttavia, arrivato il prossimo fronte utile(cursore bianco), osserviamo che il valore di ORes non è il risultato corretto dell'operazione precedente. Questo errore è dovuto alla violazione nei vincoli temporali (tempo di setup e tempo di hold)



Infatti, facendo un nuovo test e modificando la durata del tempo di vita del clock (da 10ns ho settato a 20ns), osservo che dopo il global reset =100ns, il primo fronte utile(cursore giallo) campiona IA=-128 e IB=-124 e contemporaneamente, l'uscita è pari a 0. Al successivo fronte utile(cursore blu) si ottiene in Ores il risultato corretto dell'operazione precedente che è -252(cursore bianco).

Jhenalyn Subol

matr 213485

Caratterizzazione della struttura progettata n=8bit

I. Valutazione delle Risorse Occupate

Operando la Post-Implementation Simulation è possibile visualizzare quali componenti del device a nostra disposizione sono stati utilizzati dal circuito. I blocchi logici del dispositivo sono realizzati principalmente tramite Look-up Table(LUT) per la generazione delle funzioni combinatorie e latch/flip flop in uscita come elementi di memoria.

1. Slice Logic				
Site Type	Used	Fixed	Available	Util%
Slice LUTs	31	0	53200	0.06
LUT as Logic	31	0	53200	0.06
LUT as Memory	0	0	17400	0.00
Slice Registers	27	0	106400	0.03
Register as Flip Flop	27	0	106400	0.03
Register as Latch	0	0	106400	0.00
F7 Muxes	0	0	26600	0.00
F8 Muxes	0	0	13300	0.00

Questo report ci dice che sono disponibili 53200 LUT nel dispositivo e nel nostro circuito sono state utilizzate solo 31 LUT, utilizzate per svolgere operazioni logiche, ovvero per memorizzare tabelle di verità.

In più, nel nostro circuito 27 Registri costituiti da FlipFlop sono stati anche utilizzati.

1. Slice Logic Distribution				
Site Type	Used	Fixed	Available	Util%
Slice	14	0	13300	0.11
SLICEL	5	0		
SLICEM	9	0		
LUT as Logic	31	0	53200	0.06
using 05 output only	0			
using 06 output only	19			
using 05 and 06	12			
LUT as Memory	0	0	17400	0.00
LUT as Distributed RAM	0	0		
LUT as Shift Register	0	0		
LUT Flip Flop Pairs	7	0	53200	0.01
fully used LUT-FF pairs	1			
LUT-FF pairs with one unused LUT output	6			
LUT-FF pairs with one unused Flip Flop	2			
Unique Control Sets	1			

Come vengono distribuiti le slice logic ovvero i blocchi logici elementi del dispositivo ?

14 Slice sono state utilizzate nel nostro circuito. 5 delle quali sono di tipo SLICEL dove le LUT svolgono soltanto la funzione di generatori di funzioni logiche. Mentre 9 sono di tipo SLICEM che dispongono elementi molto più complessi al posto delle semplici LUT e che possono

funzionare sia come generatori di funzioni logiche sia come memorie RAM.

Inoltre, di queste 31 LUT, 19 sono state utilizzate per memorizzare una sola tabella di verità (1 uscita- 06 output) e 12 invece sono state impiegate per memorizzare funzioni logiche a 2 uscite(05, 06).

Caratterizzazione della struttura progettata n=8bit

II. Massima Frequenza di Funzionamento

Clock Summary				
Name	Waveform	Period (ns)	Frequency (MHz)	
myclock	{0.000 5.000}	10.000	100.000	
Design Timing Summary				
Setup		Hold		
Worst Negative Slack (WNS):	4.125 ns	Worst Hold Slack (WHS):	0.329 ns	
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	
Total Number of Endpoints:	9	Total Number of Endpoints:	9	
Pulse Width		Pulse Width		
		Worst Pulse Width Slack (WPWS):	4.500 ns	
		Total Pulse Width Negative Slack (TPWS):	0.000 ns	
		Number of Failing Endpoints:	0	
		Total Number of Endpoints:	28	
All user specified timing constraints are met.				
Intra-Clock Paths - myclock				
Statistics				
Type	Worst Slack	Total Violation	Failing Endpoints	Total Endpoints
Setup	4.125 ns	0.000 ns	0	9
Hold	0.329 ns	0.000 ns	0	9
Pulse Width	4.500 ns	0.000 ns	0	28

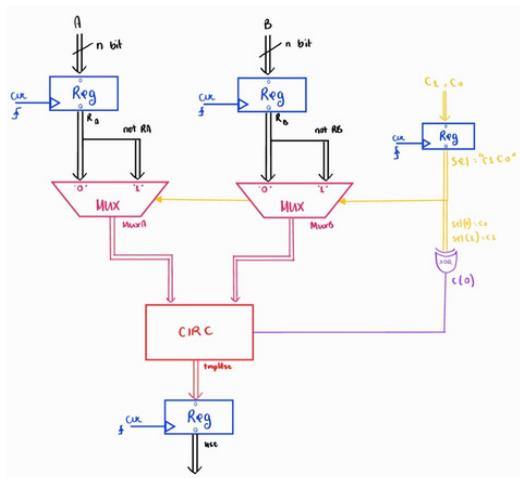
Nella PIT Simulation, la durata di un ciclo di clock è pari a t=10ns e il Worst Negative Slack era 4.125ns. Essendo positivo lo slack, ciò vuol dire che il circuito può lavorare ad una frequenza superiore a quella che ho fissato. Pertanto, il periodo di clock richiesto potrebbe essere $10 - 4.125 = 5.875$ ns ovvero 170.21Mhz. Questo vuol dire che posso ancora far lavorare il circuito più velocemente.

Caratterizzazione della struttura progettata n=8bit

III. Latenza

La LATENZA corrisponde al numero di ciclo di clock richiesto per calcolare tutti i valori di output dato un set di input. La latenza del circuito dipende da quanti livelli di registri vengono introdotti ovvero è data a $\text{num_liv} - 1$.

Prendiamo lo schema blocco del circuito:



La latenza è pari a:
 $\text{latency} = \text{num_level} - 1 = 2 - 1 = 1$

Possiamo anche conoscere il THROUGHTPUT del circuito che è dato dal numero di risultati prodotti ogni ciclo di clock. Nel circuito progettato, il throughput è pari ad 1.

Caratterizzazione della struttura progettata n=8bit

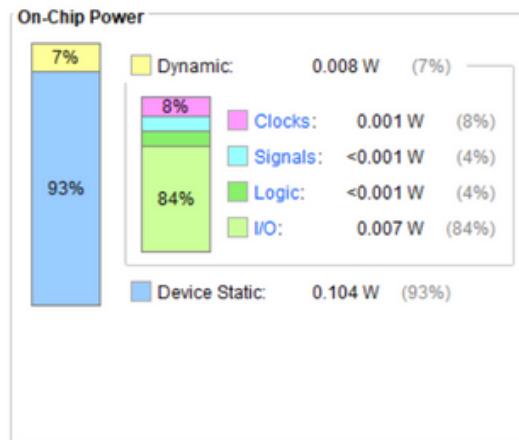
IV. Potenza Dissipata

Summary

Power analysis from implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	0.113 W
Design Power Budget:	Not Specified
Power Budget Margin:	N/A
Junction Temperature:	26,3°C
Thermal Margin:	58,7°C (4,9 W)
Effective RJA:	11,5°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

[Launch Power Constraint Advisor](#) to find and fix invalid switching activity



La dissipazione di potenza è stata stimata alla frequenza di funzionamento che corrisponde a 10ns definiti nel constraint. Quindi in questa analisi, è stato tenuto in considerazione il periodo di 10ns ovvero 100MHz.

Osserviamo in questo risultato che il percentuale della dissipazione della potenza statica è superiore a quella dinamica perché se andiamo ad analizzare il dispositivo, osserviamo che l'unica porzione utilizzata per svolgere l'operazione richiesta è molto piccola (Figura2). Queste risorse sono

sono quelle responsabili alla dissipazione dinamica mentre il resto è per la dissipazione statica.

Dal percentuale della dissipazione della potenza dinamica è compreso anche 8% dedicato al segnale di clock in quanto si tratta di un segnale particolare che ha bisogno di risorse per la sua distribuzione.

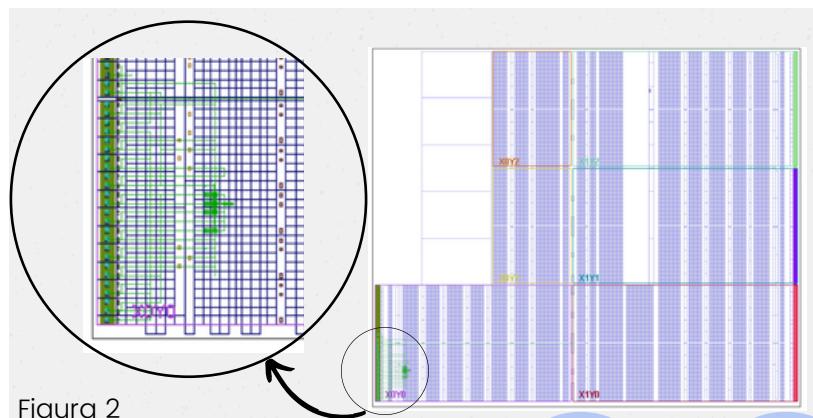
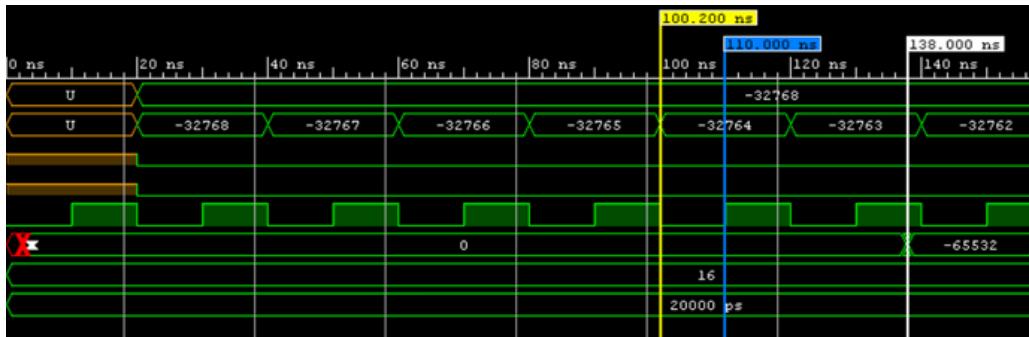


Figura 2

Ulteriori Test: n=16 bit

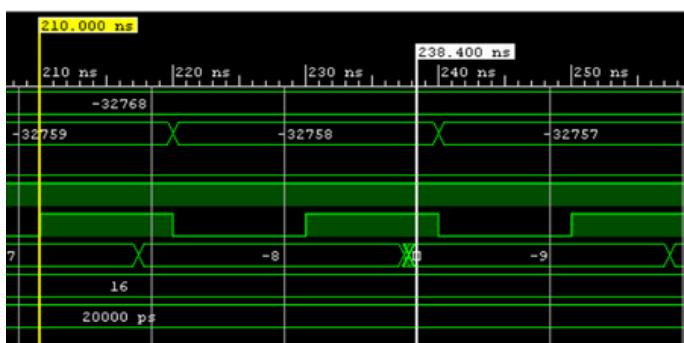
I. Verifica Funzionale (Post-Implementation Timing Simulation)

Case 1: A+B



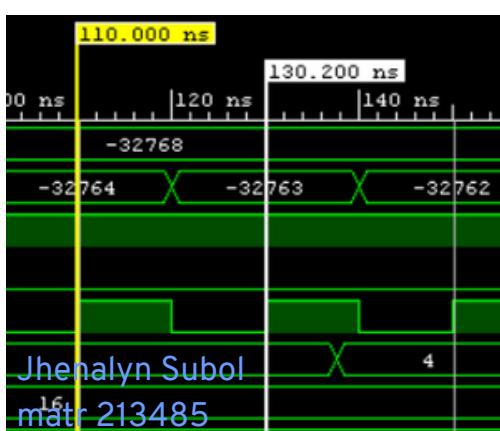
- Dopo il global reset(cursore giallo), il primo fronte utile(cursore blu) campiona IA=-32768 e IB=-32764 e li invia in ingresso ai registri che riproducono RA e RB. Contemporaneamente, ORes contiene ancora il valore precedente.
- Nell successivo fronte utile(cursore bianco), vengono campionati nuovi valori all'ingresso e contemporaneamente, ORes fornisce il risultato precedente pari a $A+B = (-32768) + (-32764) = -65532$.

Case 2: A-B



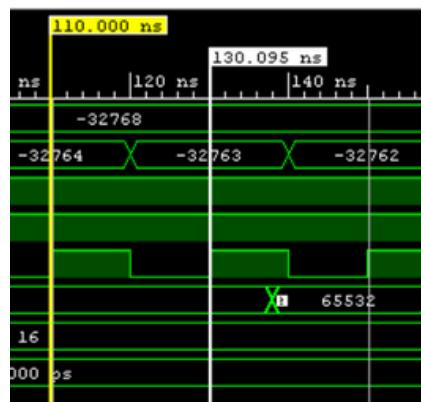
- In questo fronte di salita(cursore giallo) vengono campionati IA=-32768 e IB=-32759 e li invia in ingresso ai registri che riproducono RA e RB. Contemporaneamente, ORes contiene ancora il valore precedente.
- Nel successivo fronte utile(cursore bianco), vengono campionati nuovi valori all'ingresso e contemporaneamente, mentre ORes fornisce il risultato precedente pari a $A-B = (-32768) - (-32759) = -32768 + 32759 = -9$.

Case 3: -A+B



- Rising Edge 1(Cursore giallo): IA=-32768 , IB= -32764
- Rising Edge 2(Cursore bianco): $-A+B=-(-32768) + (-32764) = 32768 + 32764 = 65532$

Case 4: -A-B



- Rising Edge 1(Cursore giallo): IA=-32768 , IB= -32764
- Rising Edge 2(Cursore bianco): $-A-B=-(-32768) - (-32764) = 32768 + 32764 = 65532$

09

Ulteriori Test: n=16 bit

II. Valutazione di risorse occupate

1. Slice Logic			
Site Type	Used	Fixed	Available
Slice LUTs	69	0	53200
LUT as Logic	69	0	53200
LUT as Memory	0	0	17400
Slice Registers	51	0	106400
Register as Flip Flop	51	0	106400
Register as Latch	0	0	106400
F7 Muxes	0	0	26600
F8 Muxes	0	0	13300

2. Slice Logic Distribution

Site Type	Used	Fixed	Available	Util%
Slice	34	0	13300	0.26
SLICELE	22	0		
SLICEM	12	0		
LUT as Logic	69	0	53200	0.13
using O5 output only	0			
using O6 output only	47			
using O5 and O6	22			
LUT as Memory	0	0	17400	0.00
LUT as Distributed RAM	0	0		
LUT as Shift Register	0	0		
LUT Flip Flop Pairs	15	0	53200	0.03
fully used LUT-FF pairs	0			
LUT-FF pairs with one unused LUT output	15			
LUT-FF pairs with one unused Flip Flop	12			
Unique Control Sets	1			

* Note: Review the Control Sets Report for more information regarding control sets.

III. Massima Frequenza di Funzionamento

Design Timing Summary

Setup	Hold	Pulse Width
Worst Negative Slack (WNS): 2.238 ns	Worst Hold Slack (WHS): 0.215 ns	Worst Pulse Width Slack (WPWS): 4.500 ns
Total Negative Slack (TNS): 0.000 ns	Total Hold Slack (THS): 0.000 ns	Total Pulse Width Negative Slack (TPWS): 0.000 ns
Number of Failing Endpoints: 0	Number of Failing Endpoints: 0	Number of Failing Endpoints: 0
Total Number of Endpoints: 17	Total Number of Endpoints: 17	Total Number of Endpoints: 52

All user specified timing constraints are met.

$$F_{\max} = 10 \text{ ns} - 2.238 \text{ ns} = 7.762 \text{ ns} = 128.83 \text{ MHz}$$

IV. Latenza

$$\text{Latency} = \text{num_level} - 1 = 2 - 1 = 1$$

V. Potenza Dissipata

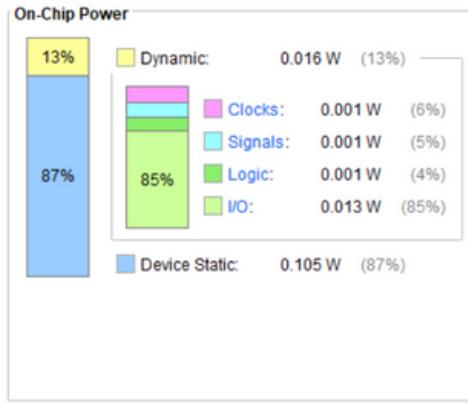
Summary

Power analysis from implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	0.12 W
Design Power Budget:	Not Specified
Power Budget Margin:	N/A
Junction Temperature:	26,4°C
Thermal Margin:	58,6°C (4,9 W)
Effective 9JA:	11,5°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

[Launch Power Constraint Advisor](#) to find and fix invalid switching activity

Jhenalyn Subol
matr 213485

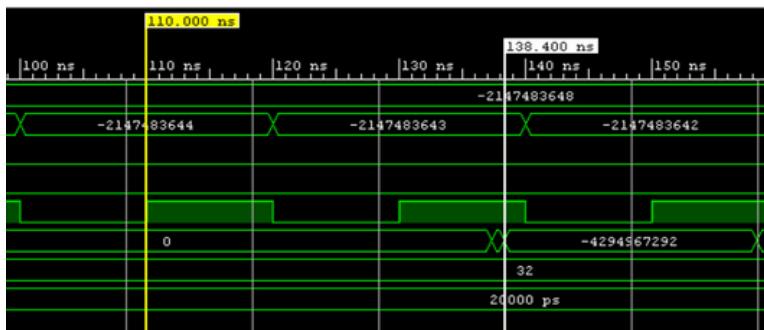


09

Ulteriori Test: n=32 bit

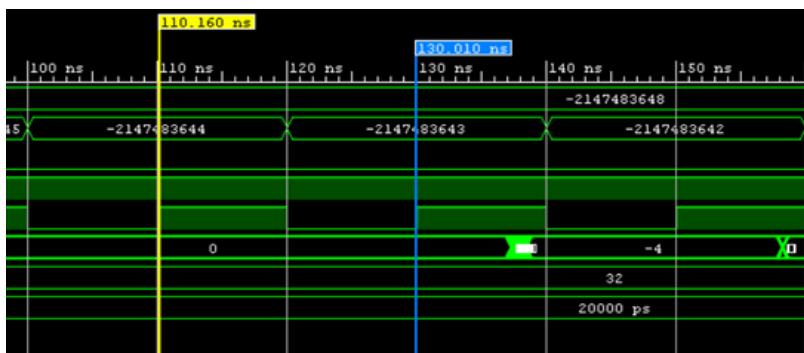
I. Verifica Funzionale (Post-Implementation Timing Simulation)

Case 1: A+B



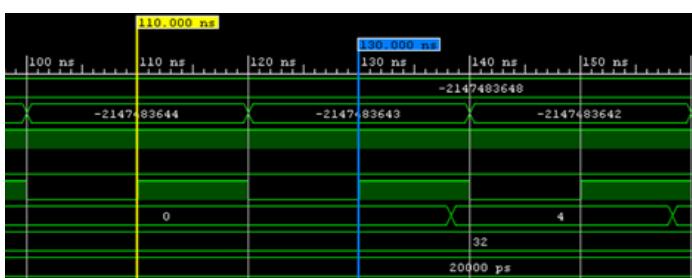
- Rising Edge 1(Cursore giallo):
 $A=-2147483648, IB=-2147483644$
- Rising Edge 2(Cursore blue):
 $A+B=-2147483648 + (-2147483644) = -4294957292$

Case 2: A-B



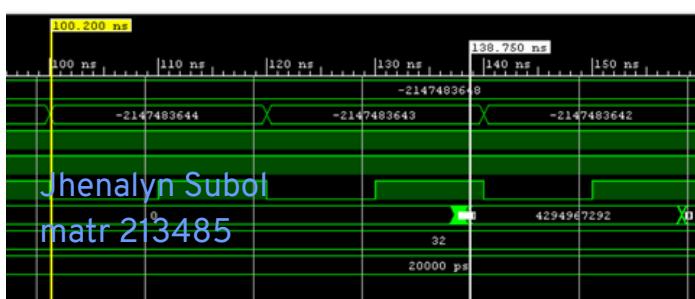
- Rising Edge 1(Cursore giallo):
 $A=-2147483648, IB=-2147483644$
- Rising Edge 2(Cursore blue):
 $A-B=-2147483648 - (-2147483644) = -4$

Case 3: -A+B



- Rising Edge 1(Cursore giallo):
 $A=-2147483648, IB=-2147483644$
- Rising Edge 2(Cursore blue):
 $-A+B=-(-2147483648) + (-2147483644) = 4$

Case 4: -A-B



- Rising Edge 1(Cursore giallo):
 $A=-2147483648, IB=-2147483644$
- Rising Edge 2(Cursore bianco):
 $-A-B=-(-2147483648) - (-2147483644) = 4294967292$

09

Ulteriori Test: n=32 bit

II. Valutazione di risorse occupate

Hierarchy								
Name	^ 1	Slice LUTs (53200)	Slice Registers (106400)	Slice (1330 0)	LUT as Logic (53200)	LUT Flip Flop Pairs (53200)	Bonded IOB (200)	BUFGCTRL (32)
Circ		140	99	62	140	32	100	1

1. Slice Logic								
2. Slice Logic Distribution								
Site Type								
	Used	Fixed	Available	Util%		Used	Fixed	Available
Site Type					Site			
Slice LUTs	140	0	53200	0.26	SLICE	62	0	13300
LUT as Logic	140	0	53200	0.26	SLICEM	45	0	
LUT as Memory	0	0	17400	0.00	LUT as Logic	17	0	
Slice Registers	99	0	106400	0.09	using 05 output only	140	0	53200
Register as Flip Flop	99	0	106400	0.09	using 06 output only	1		0.26
Register as Latch	0	0	106400	0.00	using 05 and 06	101		
F7 Muxes	0	0	26600	0.00	LUT as Memory	38		
F8 Muxes	0	0	13300	0.00	LUT as Distributed RAM	0	0	17400
					LUT as Shift Register	0	0	
					LUT Flip Flop Pairs	0	0	
					fully used LUT-FF pairs	32	0	53200
					LUT-FF pairs with one unused LUT output	0		0.06
					LUT-FF pairs with one unused Flip Flop	30		
					Unique Control Sets	20		
						1		

III. Massima Frequenza di Funzionamento

Clock Summary			
Name	Waveform	Period(ns)	Frequency (MHz)
myclock	{0.000 5.000}	10.000	100.000

Setup	Hold	Pulse Width
Worst Negative Slack (WNS): 0,700 ns	Worst Hold Slack (WHS): 0,310 ns	Worst Pulse Width Slack (WPWS): 4,500 ns
Total Negative Slack (TNS): 0,000 ns	Total Hold Slack (THS): 0,000 ns	Total Pulse Width Negative Slack (TPWS): 0,000 ns
Number of Failing Endpoints: 0	Number of Failing Endpoints: 0	Number of Failing Endpoints: 0
Total Number of Endpoints: 33	Total Number of Endpoints: 33	Total Number of Endpoints: 100

All user specified timing constraints are met.

$$F_{\max} = 10\text{ns} - 0.700\text{ns} = 9.3\text{ns} = 107.53\text{MHz}$$

IV. Latenza

$$\text{Latency} = \text{num_level} - 1 = 2 - 1 = 1$$

V. Potenza Dissipata

Summary

Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	0.136 W
Design Power Budget:	Not Specified
Power Budget Margin:	N/A
Junction Temperature:	26,6°C
Thermal Margin:	58,4°C (4,9 W)
Effective θJA:	11,5°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low
Launch Power Constraint Advisor to find and fix invalid switching activity	

