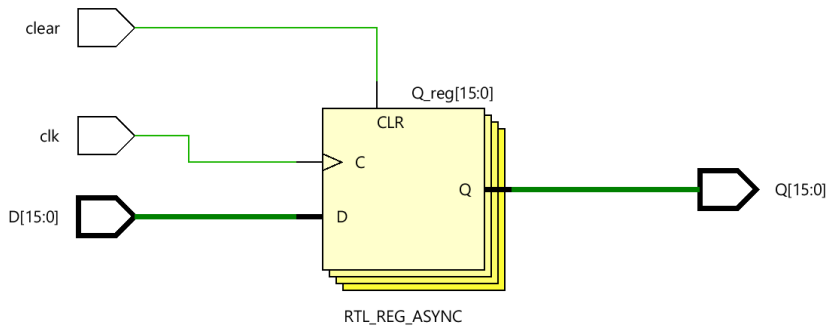
RELAZIONE ELETTRONICA DIGITALE

TRACCIA: SI RICHIEDE DI SCRIVERE IL CODICE VHDL DI UN SOMMATORE CON TRE OPERANDI SIGNED A N-BIT E DEL TESTBENCH DA IMPIEGARE PER LA SIMULAZIONE. È RICHIESTO ANCHE DI ESEGUIRE LA SIMULAZIONE POST-SYNTHESIS E QUELLA POST-IMPLEMENTATION.

IL PROGETTO IN ESAME MIRA A IMPLEMENTARE UN COMPONENTE HARDWARE IN LINGUAGGIO VHDL IN GRADO DI EFFETTUARE LA SOMMA DI TRE VALORI A N BIT. IL CIRCUITO È DIVISO IN TRE PARTI PRINCIPALI: I REGISTRI (REGISTRO), L'ADDER (ADDER), E L'UNITÀ DI SOMMA A TRE OPERANDI (SOMMA3R).

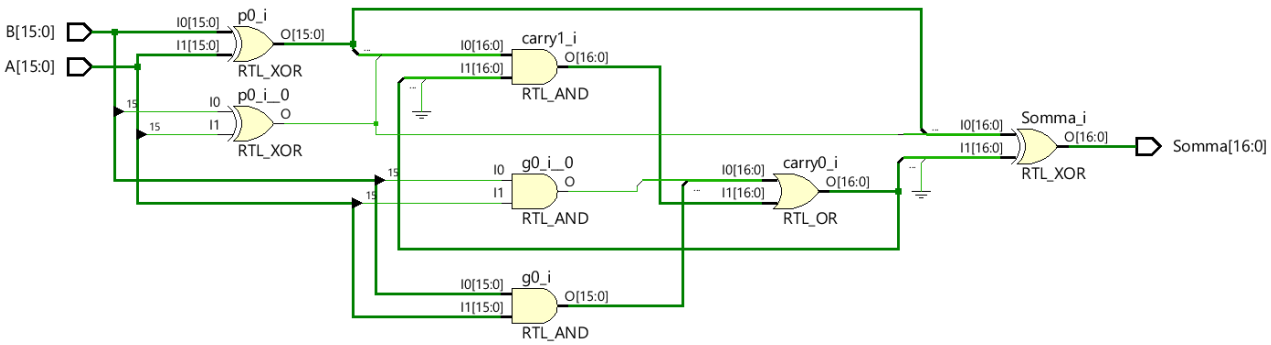
IL **REGISTRO** È IMPLEMENTATO UTILIZZANDO UN PROCESSO SENSITIVO AI FRONTI DI CLOCK E AL SEGNALE DI CLEAR. SE IL SEGNALE DI CLEAR È ALTO, TUTTI I BIT DEL REGISTRO VENGONO AZZERATI; ALTRIMENTI, ALLA PRESENZA DI UN FRONTE DI SALITA DEL CLOCK, I DATI IN INGRESSO VENGONO CARICATI NEL REGISTRO.

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

entity Registro is
    generic (n:integer:=16);
    Port ( clk : in STD_LOGIC;
           clear : in STD_LOGIC;
           D : in STD_LOGIC_VECTOR (n-1 downto 0);
           Q : out STD_LOGIC_VECTOR (n-1 downto 0));
end Registro;

architecture Behavioral of Registro is

begin
  process(clk, clear) begin
     if (clear='1') then
         Q<=(others=>'0');
     elsif rising_edge(clk) then
         Q<=D;
     end if;
  end process;

end Behavioral;
```



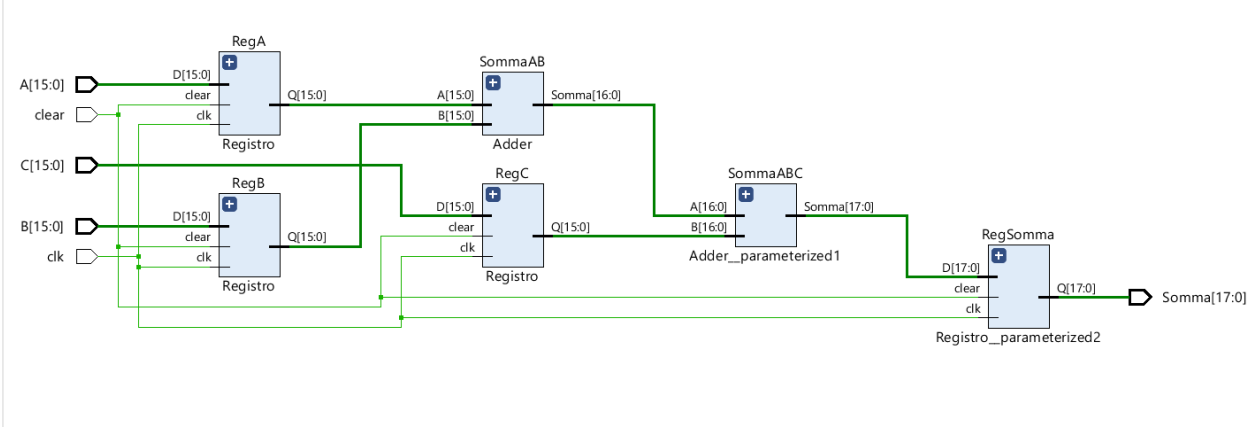
L'ENTITÀ **ADDER** RAPPRESENTA UN SOMMATORE BINARIO A N BIT. UTILIZZA I SEGNALI DI INPUT A E B E PRODUCE IL RISULTATO IN OUTPUT NELLA VARIABILE SOMMA. IL SOMMATORE È IMPLEMENTATO ATTRAVERSO L'UTILIZZO DI SEGNALI INTERMEDI P, G, E CARRY, OTTENUTI MEDIANTE LE OPERAZIONI LOGICHE XOR E AND. IL RISULTATO FINALE È CALCOLATO EFFETTUANDO LA SOMMA BIT A BIT CONSIDERANDO ANCHE IL RIPORTO

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

entity Adder is
    generic(n:Integer:=16);
    port(A,B : in STD_LOGIC_VECTOR(n-1 downto 0);
         Somma: out STD_LOGIC_VECTOR(n downto 0));
end Adder;

architecture Behavioral of Adder is
    signal p,g : STD_LOGIC_VECTOR(n downto 0);
    signal carry: STD_LOGIC_VECTOR(n+1 downto 0);
begin
    p<=( B(n-1) xor A(n-1)) & (B xor A);
    g<=( B(n-1) and A(n-1)) & (B and A);
    carry(0)<='0';
    carry(n+1 downto 1)<= g or (p and carry(n downto 0));
    Somma<=p xor carry(n downto 0);

end Behavioral;
```



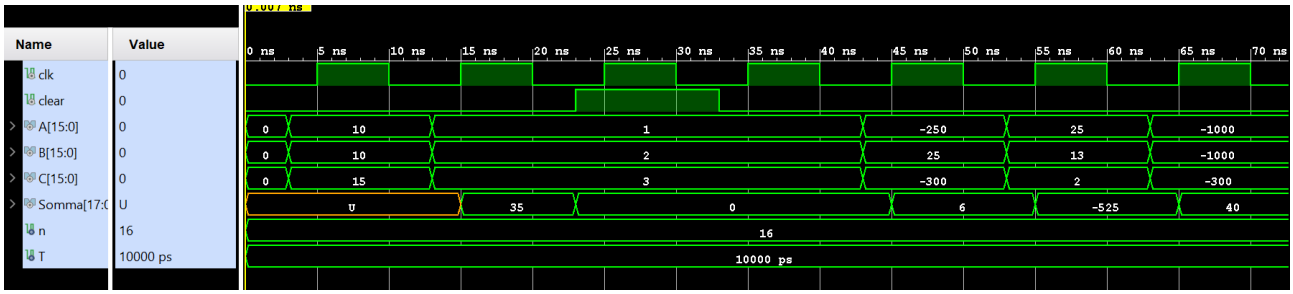
L'ENTITÀ PRINCIPALE, **SOMMA3R**, INTEGRA LE ENTITÀ REGISTRO E ADDER PER SOMMARE TRE VALORI A N BIT. I TRE VALORI IN INGRESSO (A, B, C) VENGONO CARICATI IN TRE REGISTRI SEPARATI (REGA, REGB, E REGC). SUCCESSIVAMENTE, A E B VENGONO SOMMATI MEDIANTE L'ADDER, PRODUCENDO LA VARIABILE SUMAB. IL SEGNALE C VIENE ESTESO A N+1 BIT E SOMMATO A SUMAB ATTRAVERSO UN SECONDO ADDER, GENERANDO LA VARIABILE ISOMMA. INFINE, IL RISULTATO VIENE CARICATO IN UN ULTERIORE REGISTRO (REGSOMMA).

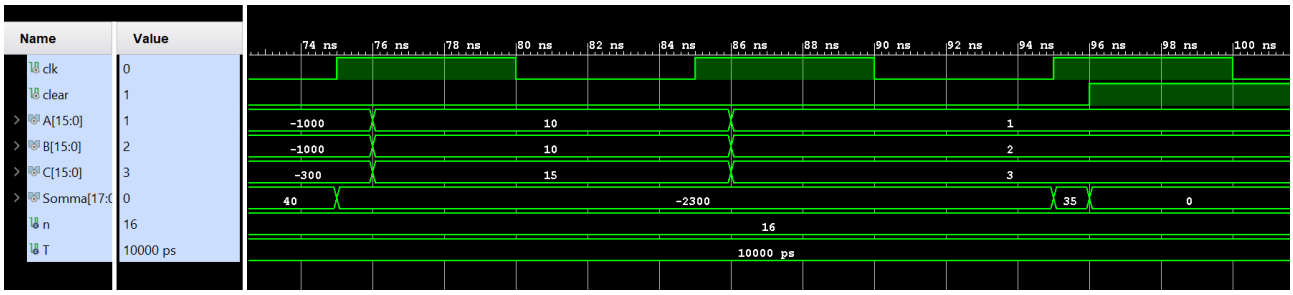
```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity Somma3R is
    generic(n:Integer:=16);
    port(A,B,C : in STD_LOGIC_VECTOR(n-1 downto 0);
         Somma : out STD_LOGIC_VECTOR(n+1 downto 0);
         clk: in STD_LOGIC;
         clear: in STD_LOGIC);
end Somma3R;

architecture Behavioral of Somma3R is

component Registro is
    generic(n:Integer:=16);
     port( clk: in STD_LOGIC;
           clear : in STD_LOGIC;
           D : in STD_LOGIC_VECTOR (n-1 downto 0);
           Q : out STD_LOGIC_VECTOR (n-1 downto 0));
end component;

component Adder is
    generic(n:Integer:=16);
    port(A,B : in STD_LOGIC_VECTOR(n-1 downto 0);
         Somma: out STD_LOGIC_VECTOR(n downto 0));
end component;

signal Ia,Ib,Ic: STD_LOGIC_VECTOR(n-1 downto 0);
signal SumAB, C_esteso: STD_LOGIC_VECTOR(n downto 0);
signal ISomma: STD_LOGIC_VECTOR(n+1 downto 0);


begin
RegA: Registro generic map(n) port map(clk, clear, A, Ia);
RegB: Registro generic map(n) port map(clk, clear, B, Ib);
RegC: Registro generic map(n) port map(clk, clear, C, Ic);

--A+B
SommaAB: Adder generic map(n)port map(Ia,Ib,SumAB);

--estendo C
C_esteso(n)<=Ic(n-1);
C_esteso(n-1 downto 0) <= Ic(n-1 downto 0);

--(A+B) +C
SommaABC: Adder generic map(n+1) port map(SumAB, C_esteso, ISomma);
--inserisco il risultato nel registro
RegSomma: Registro generic map(n+2) port map(clk, clear, ISomma, Somma);

end Behavioral;
```



IL TESTBENCH **SOMMA3R_TB** È STATO PROGETTATO PER SIMULARE IL FUNZIONAMENTO DEL SOMMATORE.

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

entity Somma3R_TB is
end Somma3R_TB;

architecture Behavioral of Somma3R_TB is
    constant n : integer := 16;

    component Somma3R is
    generic(n:Integer:=16);
    port(A,B,C : in STD_LOGIC_VECTOR(n-1 downto 0);
        Somma : out STD_LOGIC_VECTOR(n+1 downto 0);
        clk: in STD_LOGIC;
        clear: in STD_LOGIC);
    end component;

    signal clk : STD_LOGIC := '0';
    signal clear : STD_LOGIC := '0';
    signal A, B, C : STD_LOGIC_VECTOR(n-1 downto 0):=(others=>'0');
    signal Somma : STD_LOGIC_VECTOR(n+1 downto 0);
    constant T : time := 10 ns;

begin

  Sum: Somma3R generic map(n) port map(A, B, C, Somma, clk, clear);

    process begin
        clk<='0';
        wait for T/2;
        clk<='1';
        wait for T/2;
    end process;

    [illegible]

process begin
    clear<='0';
    wait for 3ns;
    A <= CONV_STD_LOGIC_VECTOR(10,n);
    B <= CONV_STD_LOGIC_VECTOR(10,n);
    C <= CONV_STD_LOGIC_VECTOR(15,n);
    wait for T;
    A <= CONV_STD_LOGIC_VECTOR(1,n);
    B <= CONV_STD_LOGIC_VECTOR(2,n);
    C <= CONV_STD_LOGIC_VECTOR(3,n);
    wait for T;
    clear<='1';
    wait for T;
    clear<='0';
    wait for T;
    A <= CONV_STD_LOGIC_VECTOR(-250,n);
    B <= CONV_STD_LOGIC_VECTOR(25,n);
    C <= CONV_STD_LOGIC_VECTOR(-300,n);
    wait for T;
    A <= CONV_STD_LOGIC_VECTOR(25,n);
    B <= CONV_STD_LOGIC_VECTOR(13,n);
    C <= CONV_STD_LOGIC_VECTOR(2,n);
    wait for T;
    A <= CONV_STD_LOGIC_VECTOR(-1000,n);
    B <= CONV_STD_LOGIC_VECTOR(-1000,n);
    C <= CONV_STD_LOGIC_VECTOR(-300,n);
    wait for T;
end process;
end Behavioral;
```

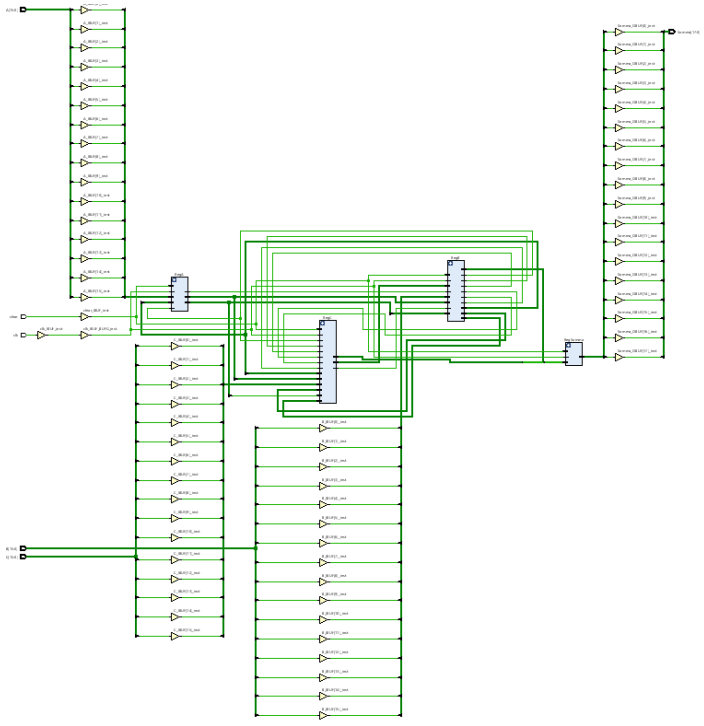




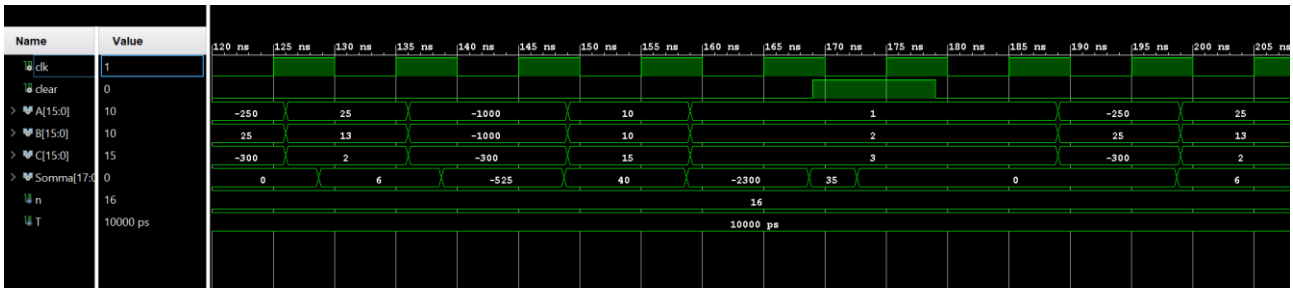
DURANTE IL PROCESSO DI SINTESI, SONO STATI INCLUSI VINCOLI SPECIFICI PER OTTIMIZZARE L'IMPLEMENTAZIONE FISICA DEL CIRCUITO. IN PARTICOLARE, È STATO DEFINITO UN VINCOLO PER IL CLOCK, INDICATO COME "MYCLOCK".

```
1 create_clock -period 10.000 -name my_clock -waveform {0.000 5.000} [get_ports clk]
2
3
```

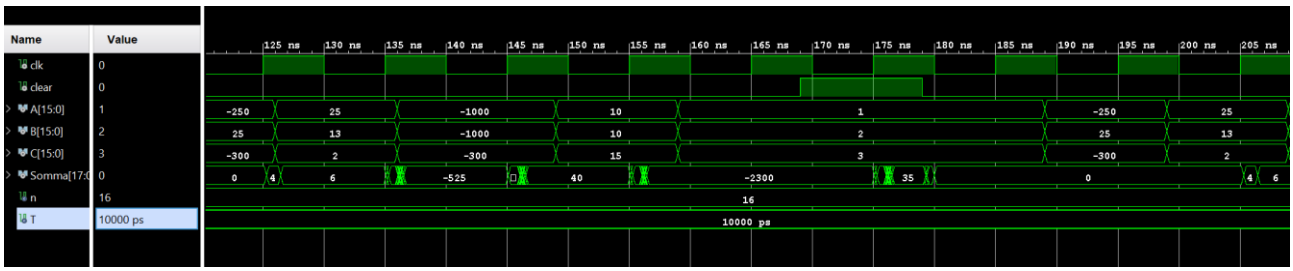
SYNTHESIS



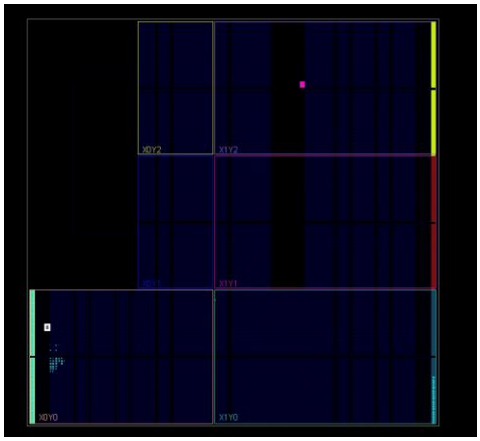
SIMULAZIONE POST-SYNTHESIS - TIMING



SIMULAZIONE POST-IMPLEMENTATION – TIMING



OPERANDO LA POST-IMPLEMENTATION SIMULATION È POSSIBILE VISUALIZZARE QUALI COMPONENTI DEL DEVICE A NOSTRA DISPOSIZIONE SONO UTILIZZATI DAL CIRCUITO

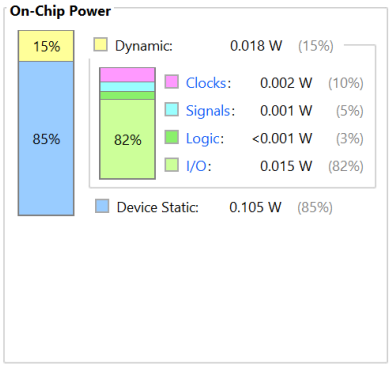


TEMPERATURA DI GIUNZIONE E POWER ON-CHIP CON CONSTRAINT

Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

Total On-Chip Power:	**0.123 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 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



Site Type	Used	Fixed	Available	Util%
Slice LUTs*	56	0	53200	0.11
LUT as Logic	56	0	53200	0.11
LUT as Memory	0	0	17400	0.00
Slice Registers	66	0	106400	0.06
Register as Flip Flop	66	0	106400	0.06
Register as Latch	0	0	106400	0.00
F7 Muxes	0	0	26600	0.00
F8 Muxes	0	0	13300	0.00

Design Timing Summary

WNS(ns)	TNS(ns)	TNS Failing Endpoints	TNS Total Endpoints
2.507	0.000	0	18

All user specified timing constraints are met.

WHS(ns)	THS(ns)	THS Failing Endpoints	THS Total Endpoints
0.191	0.000	0	18