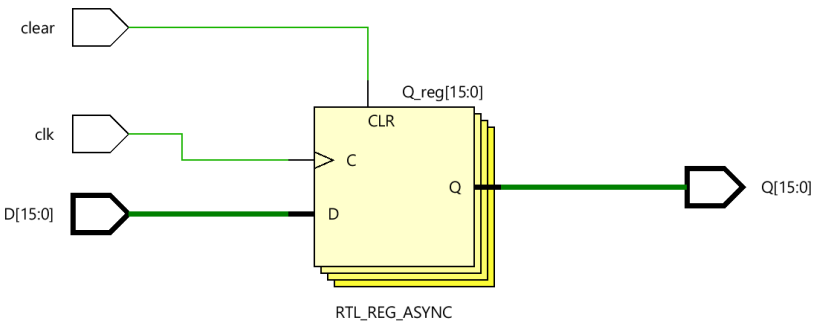
RELAZIONE ELETTRONICA DIGITALE

TRACCIA: DESCRIVERE UN SOMMATORE CARRY-SAVE, A TRE OPERANDI IN COMPLEMENTO A DUE, PROVVISTO DI REGISTRI IN INGRESSO ED IN USCITA.ESEGUIRE LE SIMULAZIONI BEHAVIOURAL, POST-SYNTHESIS E POST-IMPLEMENTATION. RIPORTARE NELLA RELAZIONE, OLTRE AL CODICE E AGLI SCREEN SHOTS DELLE SIMULAZIONI, TUTTI I DATI RELATIVI AD UTILIZZO DI RISIRSE E PATH CRITICO.

IL PROGETTO IN OGGETTO SI PROPONE DI IMPLEMENTARE UN SOMMATORE CARRY-SAVE A TRE OPERANDI IN COMPLEMENTO A DUE, UTILIZZANDO REGISTRI IN INGRESSO ED IN USCITA. L'OBIETTIVO PRINCIPALE È REALIZZARE UN CIRCUITO CHE EFFETTUI LA SOMMA DI TRE NUMERI IN FORMATO BINARIO A COMPLEMENTO A DUE. IL PROGETTO È SUDDIVISO IN DIVERSE COMPONENTI, OGNUNA DELLE QUALI SVOLGE UN RUOLO SPECIFICO NEL CALCOLO DELLA SOMMA.

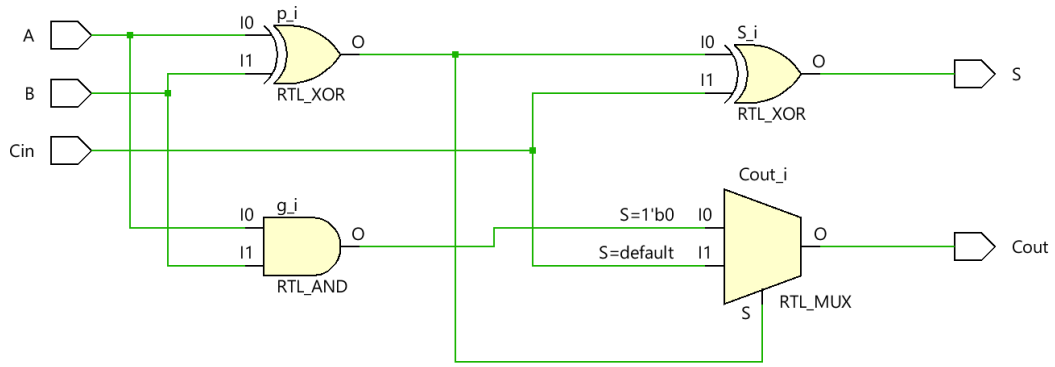
IL **REGISTRO** È IMPLEMENTATO UTILIZZANDO UN PROCESSO SENSITIVO AI FRONTI DI CLOCK E AL SEGNALE DI CLEAR. SE IL SEGNALE DI CLEAR È ALTO, TUTTI I BIT DEL REGISTRO VENGONO AZZERATI; ALTRIMENTI, ALLA PRESENZA DI UN FRONTE DI SALITA DEL CLOCK, I DATI IN INGRESSO VENGONO CARICATI NEL REGISTRO.

```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;
```



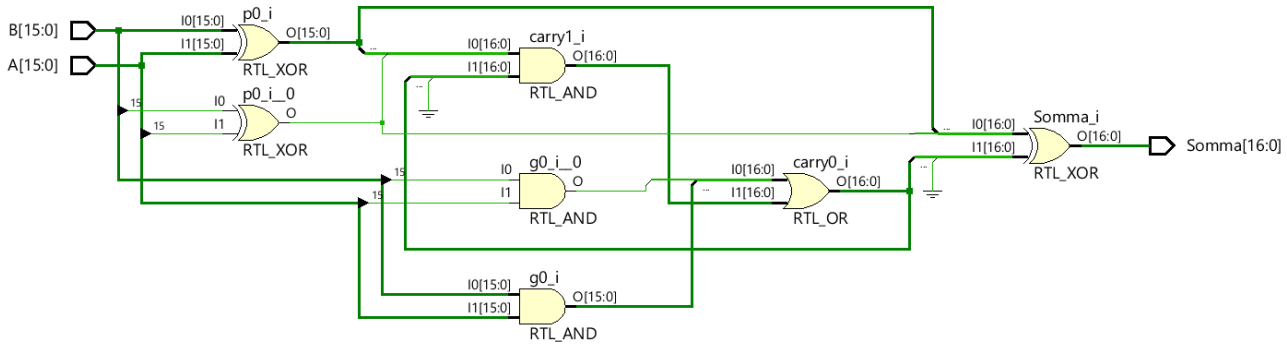
FULL ADDER: CIRCUITO SOMMATORE IN GRADO DI SOMMARE 3 BIT: A, B, CIN E GENERA COME USCITA S E IL RIPORTO COUT

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4
5  entity FullAdder is
6      port (A,B,Cin: in STD_LOGIC;
7                Cout,S: out STD_LOGIC);
8  end FullAdder;
9
10 architecture Behavioral of FullAdder is
11 signal p,g: STD_LOGIC;
12 begin
13     Cout<=g when p='0' else Cin;
14     S<= p xor Cin;
15     p<=a xor b;
16     g<= a and b;
17
18 end Behavioral;
```



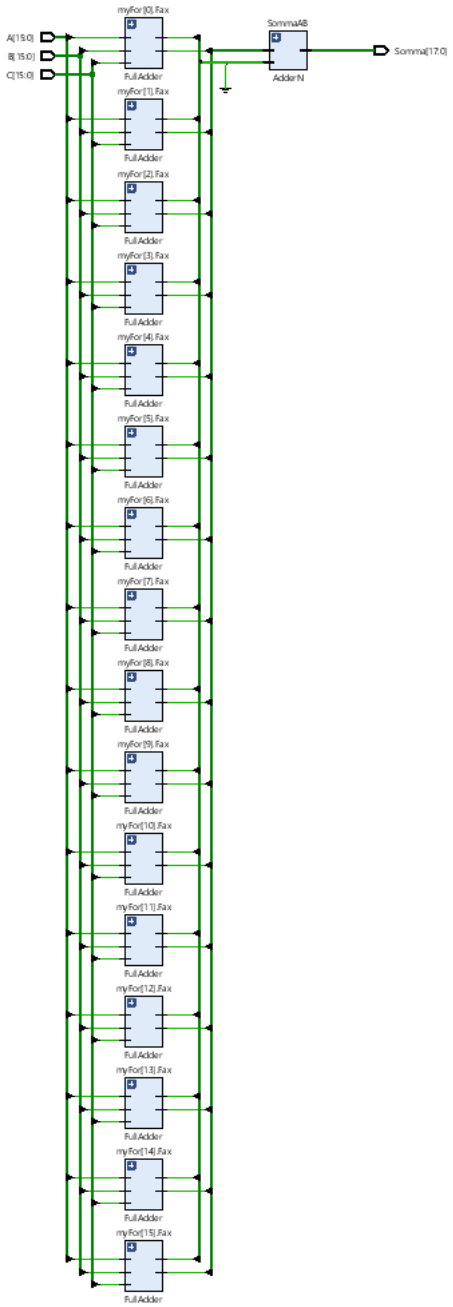
ADDERN: SOMMATORE BINARIO A N BIT. UTILIZZA I SEGNALI DI INPUT A E B E PRODUCE IL RISULTATO IN OUTPUT NELLA VARIABILE SOMMA. IL SOMMATORE È IMPLEMENTATO ATTRAVERSO L'UTILIZZO DI SEGNALI INTERMEDI P, G, E CARRY, OTTENUTI MEDIANTE LE OPERAZIONI LOGICHE XOR E AND. IL RISULTATO FINALE È CALCOLATO EFFETTUANDO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

```
2  library IEEE;
3  use IEEE.STD_LOGIC_1164.ALL;
4
5  entity AdderN is
6      generic(n:Integer:=16);
7      port(A,B : in STD_LOGIC_VECTOR(n-1 downto 0);
8           Somma: out STD_LOGIC_VECTOR(n downto 0));
9  end AdderN;
10
11 architecture Behavioral of AdderN is
12     signal p,g : STD_LOGIC_VECTOR(n downto 0);
13     signal carry: STD_LOGIC_VECTOR(n+1 downto 0);
14 begin
15     p<=( B(n-1) xor A(n-1)) & (B xor A);
16     g<=( B(n-1) and A(n-1)) & (B and A);
17     carry(0)<='0';
18     carry(n+1 downto 1)<= g or (p and carry(n downto 0));
19     Somma<=p xor carry(n downto 0);
20
21 end Behavioral;
```



IL **SOMMATORE** È IL COMPONENTE PRINCIPALE DEL PROGETTO. COMBINA L'UTILIZZO DI REGISTRI, FULL ADDERS E L'ADDERN PER ESEGUIRE LA SOMMA CARRY-SAVE A TRE OPERANDI IN COMPLEMENTO A DUE.

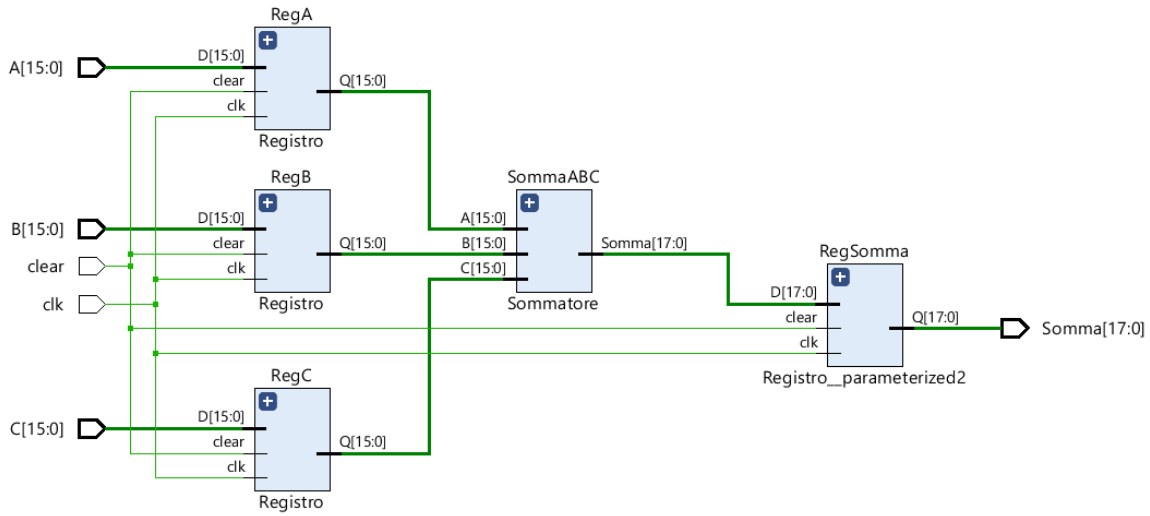
```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;



entity Sommatore 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));
end Sommatore;

architecture Behavioral of Sommatore is

    component FullAdder is
        port (A,B,Cin: in STD_LOGIC;
              Cout,S: out STD_LOGIC);
    end component;

    component AdderN 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 SP, VR: STD_LOGIC_VECTOR(n-1 downto 0);
signal VR_shift, SP_esteso:STD_LOGIC_VECTOR(n downto 0);
signal IA, IB: STD_LOGIC_VECTOR(n+1 downto 0);
signal sum: STD_LOGIC_VECTOR(n+2 downto 0);

begin

    myFor: for i in 0 to n-1 generate
        Fax: FullAdder port map(A(i), B(i), C(i), VR(i), SP(i));
    end generate;

    --shift sx VR
    VR_shift(n downto 1)<=VR(n-1 downto 0);
    VR_shift(0)<='0';

    --estendo SP
    SP_esteso(n)<=SP(n-1);
    SP_esteso(n-1 downto 0)<=SP(n-1 downto 0);

    --estendo entrambi i vettori
    IA(n+1)<=SP_esteso(n);
    IA(n downto 0)<=SP_esteso(n downto 0);

    IB(n+1)<=VR_shift(n);
    IB(n downto 0)<=VR_shift(n downto 0);

    --sommo i due vettori
    SommaAB: AdderN generic map(n+2) port map(IA,IB,sum);

    Somma<= sum(n+1 downto 0);


end Behavioral;
```



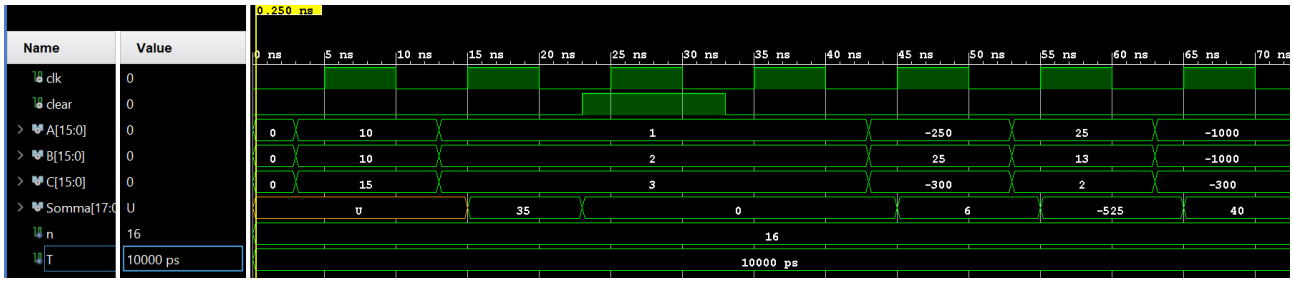
CARRY-SAVE RAPPRESENTA L'ENTITÀ CHE ASSEMBLA LE COMPONENTI PRECEDENTI PER OTTENERE IL RISULTATO FINALE DELLA SOMMA CARRY-SAVE. UTILIZZA TRE REGISTRI (REGA, REGB, REGC), UN SOMMATORE (SOMMAABC), E UN REGISTRO (REGSOMMA) PER MEMORIZZARE IL RISULTATO.

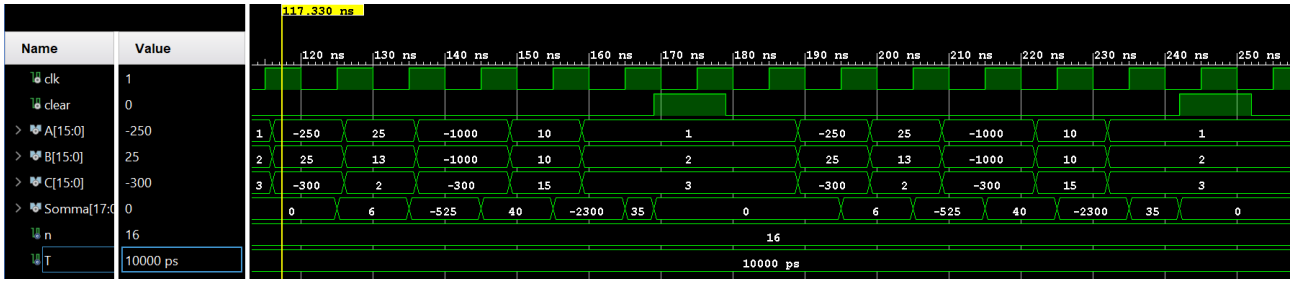
```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity CarrySave is
5      generic(n:Integer:=16);
6      port(A,B,C : in STD_LOGIC_VECTOR(n-1 downto 0);
7           Somma: out STD_LOGIC_VECTOR(n+1 downto 0);
8           clk : in STD_LOGIC;
9           clear: in STD_LOGIC);
10
11 end CarrySave;
12
13 architecture Behavioral of CarrySave is
14
15 component Registro is
16     generic (n:integer:=16);
17     Port ( clk : in STD_LOGIC;
18            clear : in STD_LOGIC;
19            D : in STD_LOGIC_VECTOR (n-1 downto 0);
20            Q : out STD_LOGIC_VECTOR (n-1 downto 0));
21 end component;
22
23 component Sommatore is
24     generic(n:Integer:=16);
25     port(A,B,C : in STD_LOGIC_VECTOR(n-1 downto 0);
26          Somma: out STD_LOGIC_VECTOR(n+1 downto 0));
27 end component;
28
29 signal IA, IB, IC: STD_LOGIC_VECTOR(n-1 downto 0);
30 signal Isomma:STD_LOGIC_VECTOR(n+1 downto 0);
```

```
31
32 begin
33 RegA: Registro generic map(n) port map(clk, clear, A, Ia);
34 RegB: Registro generic map(n) port map(clk, clear, B, Ib);
35 RegC: Registro generic map(n) port map(clk, clear, C, Ic);
36
37 SommaABC: Sommatore generic map(n) port map(IA,IB,IC, Isomma);
38
39 RegSomma: Registro generic map(n+2) port map(clk, clear, ISomma, SOmma);
40
41 end Behavioral;
42
```



TESTBENCH: È STATO CREATO PER VERIFICARE IL FUNZIONAMENTO CORRETTO DEL SOMMATORE.

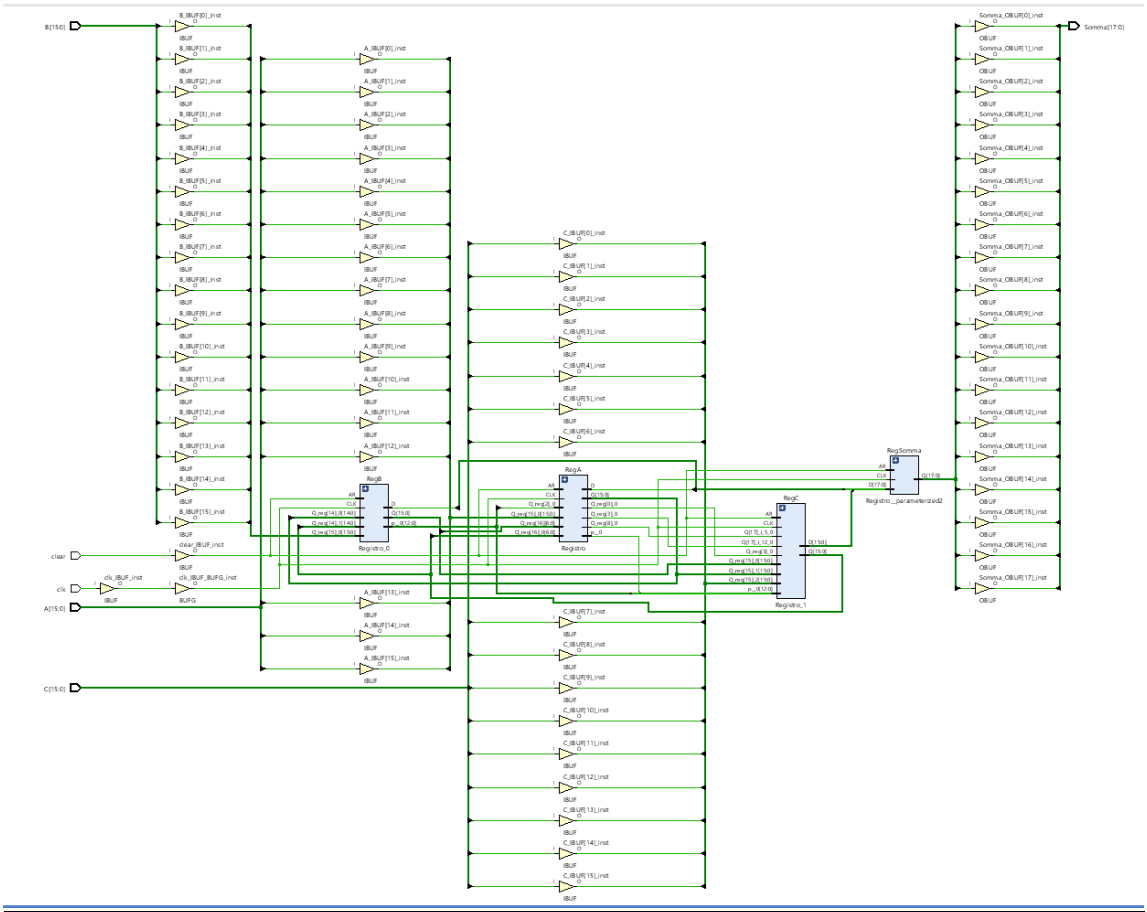




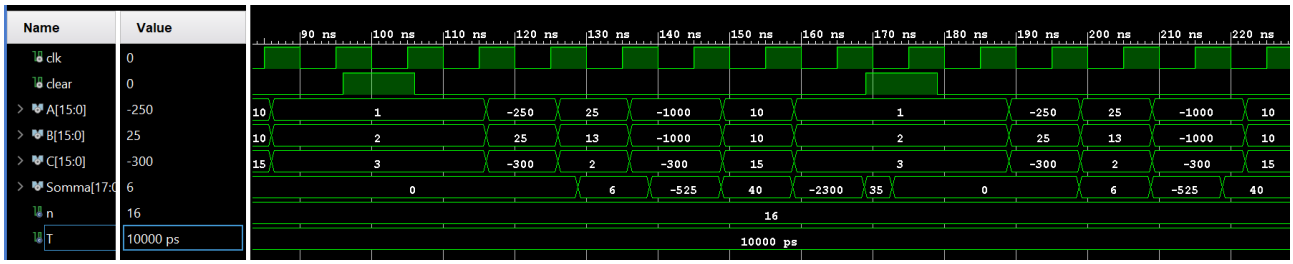
DURANTE IL PROCESSO DI SINTESI, SONO STATI INCLUSI VINCOLI SPECIFICI PER OTTIMIZZARE L'IMPLEMENTAZIONE FISICA DEL CIRCUITO. IN PARTICOLARE, È STATO DEFINITO UN VINCOLO PER IL CLOCK, INDICATO COME "MYCLOCK".

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

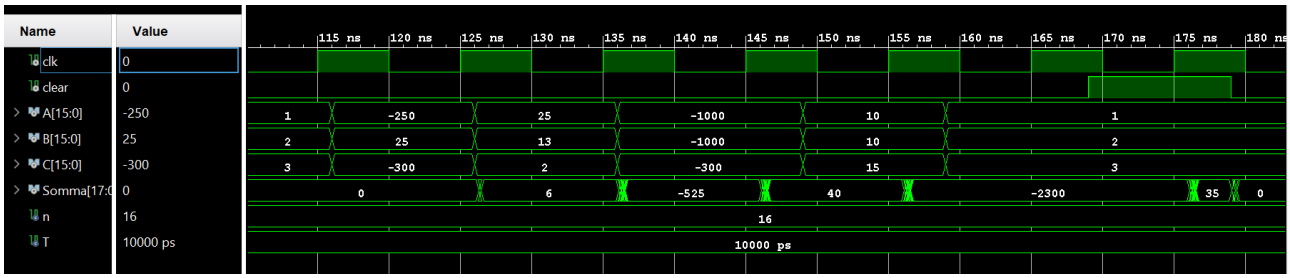
SYNTHESIS



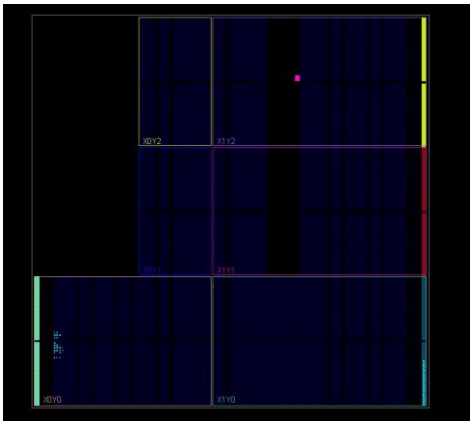
SIMULAZIONE POST-SYNTHESIS - TIMING



SIMULAZIONE POST-IMPLEMENTATION – TIMING



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

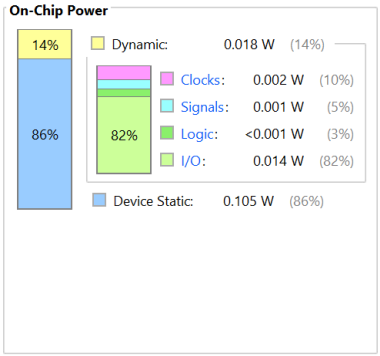


TEMPERATURA DI GIUNZIONE E POWER ON-CHIP CON CONSTRAINT

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:	**0.122 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



WNS(ns)	TNS(ns)	TNS Failing Endpoints	TNS Total Endpoints	WHS(ns)	THS(ns)	THS Failing Endpoints	THS Total Endpoints
4.381	0.000	0	18	0.166	0.000	0	18

Site Type	Used	Fixed	Available	Util%
Slice LUTs	62	0	53200	0.12
LUT as Logic	62	0	53200	0.12
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

```
Max Delay Paths
--------------------------------------------------------------------------------------
Slack (MET) :             4.381ns  (required time - arrival time)
  Source:                 RegC/Q_reg[3]/C
                            (rising edge-triggered cell FDCE clocked by my_clock  {rise@0.000ns fall@5.000ns period=10.000ns})
  Destination:            RegSomma/Q_reg[15]/D
                            (rising edge-triggered cell FDCE clocked by my_clock  {rise@0.000ns fall@5.000ns period=10.000ns})
  Path Group:             my_clock
  Path Type:              Setup (Max at Slow Process Corner)
  Requirement:            10.000ns  (my_clock rise@10.000ns - my_clock rise@0.000ns)
  Data Path Delay:        5.589ns  (logic 1.614ns (28.876%)  route 3.975ns (71.124%))
  Logic Levels:           7  (LUT3=1 LUT6=6)
  Clock Path Skew:        -0.023ns (DCD - SCD + CPR)
    Destination Clock Delay (DCD):    4.578ns = ( 14.578 - 10.000 )
    Source Clock Delay      (SCD):    5.098ns
    Clock Pessimism Removal (CPR):    0.497ns
  Clock Uncertainty:      0.035ns  ((TSJ^2 + TIJ^2)^1/2 + DJ) / 2 + PE
    Total System Jitter     (TSJ):    0.071ns
    Total Input Jitter      (TIJ):    0.000ns
    Discrete Jitter          (DJ):    0.000ns
    Phase Error              (PE):    0.000ns
```