*Università "Federico II" - Napoli - A.A. 2011/2012*
*Corso di Laurea in Ingegneria Elettronica*


# *Architettura dei Sistemi Integrati – Elaborato*
# *Progettazione microprocessore "Pico16"*

Alunno
**X**
**X**

Professore
**Antonio Strollo**

## 1. Progettazione ALU

22/03/2012 - Esercitazione 2

Come primo passo verso la progettazione del microprocessore *Pico16*, è stata progettata la *Arithmetic Logic Unit*. In una prima accezione, essa può essere vista come suddivisa in due sezioni: una sezione logica ed una aritmetica, utilizzando un multiplexer a valle per selezionare l'uscita dei due blocchi. In tale caso, è possibile progettare parte aritmetica e parte logica come due blocchi separati.
Focalizzando l'attenzione sull'unità aritmetica, è possibile realizzare il circuito mostrato in figura 1.1 per svolgere numerose operazioni aritmetiche.

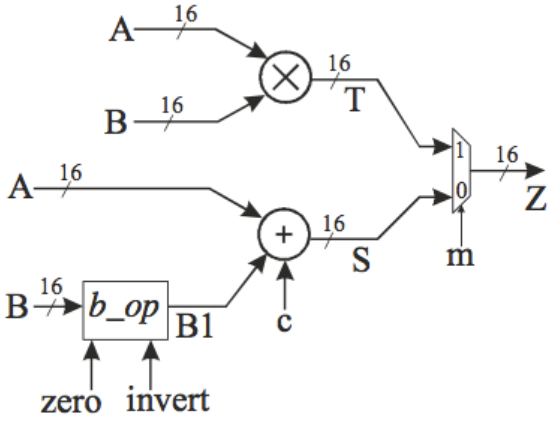


**Figura 1.1 –Unità aritmetica**

Il blocco b_op riceve in ingresso due segnali di selezione, *zero* ed *invert*, in base ai quali l'uscita B1 assumerà valori diversi.
Il codice VHDL utilizzato per implementare il blocco aritmetico è di seguito riportato:

```
--Architettura dei Sistemi Integrati * ESERCITAZIONE 2 * 22/03/2012 *
--Progettazione della parte aritmetica della ALU

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity aritm is
       port (
              A : in std_logic_vector (15 downto 0);
              B : in std_logic_vector (15 downto 0);
              zero, invert, c, m : in std_logic;
              Z : out std_logic_vector (15 downto 0)
            );
end entity;

architecture dataflow of aritm is
signal B1, S, T : signed (15 downto 0);
signal mult : signed (31 downto 0);
signal tmp : std_logic_vector (1 downto 0);
signal cnum : signed (15 downto 0);

begin

--Struttura blocco di condizionamento segnale B
```

```
tmp <= (zero,invert);
with tmp select
       B1 <= signed (B) when "00",
             signed (not B) when "01",
             (others => '0') when "10",
             (others => '1') when others;

--Blocco che calcola la somma
--E' necessario che le operazioni vadano fatte su segnali signed
       cnum <= (0 => c, others => '0');      --Trasforma il bit c in un
--numero da 16 bit
       S <= B1 + signed(A) + cnum;

--Blocco che calcola la moltiplicazione

       mult <= signed (A) * signed (B);

--Trasformo mult in T

       T (15 downto 0) <= mult (31 downto 16); --In T riporto i 16 bit più
--significativi di mult

--Multiplexer che seleziona  l'uscita

       Z <= std_logic_vector(T) when m='1' else
            std_logic_vector(S);

end dataflow;
```

L'unità aritmetica definita, è stata simulata utilizzando un test bench. In figura 1.2 sono riportati i segnali in ingresso ed in uscita all'unità aritmetica; è possibile verificare il corretto funzionamento della stessa verificando l'assenza di messaggi di errore nella console del software *SimVision* (figura 1.3).
Il test bench utilizzato per la simulazione è il seguente:

```
--Architettura dei Sistemi Integrati * ESERCITAZIONE 2 * 22/03/2012 *
--Progettazione della parte aritmetica della ALU - testbench

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity tb_aritm is
end tb_aritm;

architecture test of tb_aritm is
component aritm is port (
                A : in std_logic_vector (15 downto 0);
             B : in std_logic_vector (15 downto 0);
             zero, invert, c, m : in std_logic;
             Z : out std_logic_vector (15 downto 0)
           );
end component;

signal myA, myB, myZ : std_logic_vector (15 downto 0);
signal codice : std_logic_vector (3 downto 0);

begin


dut : aritm port map (
A => myA,
B => myB,
m => codice(3),
```

```
invert => codice(2),
zero => codice(1),
c => codice(0),
Z => myZ );

stimulus : process

begin

myA <= std_logic_vector (to_signed (99, 16));
myB <= std_logic_vector (to_signed (43, 16));
codice <= "0000";
wait for 10 ns;
assert myZ <= std_logic_vector (to_signed (142, 16))
report "errore in A+B"
severity warning;

myA <= std_logic_vector (to_signed (99, 16));
myB <= std_logic_vector (to_signed (43, 16));
codice <= "0010";
wait for 10 ns;
assert myZ <= std_logic_vector (to_signed (99, 16))
report "errore in Z=A"
severity warning;

myA <= std_logic_vector (to_signed (99, 16));
myB <= std_logic_vector (to_signed (43, 16));
codice <= "0011";
wait for 10 ns;
assert myZ <= std_logic_vector (to_signed (100, 16))
report "errore in A+1"
severity warning;

myA <= std_logic_vector (to_signed (99, 16));
myB <= std_logic_vector (to_signed (43, 16));
codice <= "0101";
wait for 10 ns;
assert myZ <= std_logic_vector (to_signed (56, 16))
report "errore in A-B"
severity warning;

myA <= std_logic_vector (to_signed (99, 16));
myB <= std_logic_vector (to_signed (111, 16));
codice <= "0101";
wait for 10 ns;
assert myZ <= std_logic_vector (to_signed (-12, 16))
report "errore in A-B"
severity warning;

myA <= std_logic_vector (to_signed (99, 16));
myB <= std_logic_vector (to_signed (111, 16));
codice <= "0110";
wait for 10 ns;
assert myZ <= std_logic_vector (to_signed (98, 16))
report "errore in A-1"
severity warning;

myA <= std_logic_vector (to_signed (4352, 16));
myB <= std_logic_vector (to_signed (2304, 16));
codice <= "1000";
wait for 10 ns;
assert myZ <= std_logic_vector (to_signed (153, 16))
report "errore in A*B"
severity warning;

myA <= std_logic_vector (to_signed (4352, 16));
myB <= std_logic_vector (to_signed (2304, 16));
codice <= "1101";
```

```
wait for 10 ns;
assert myZ <= std_logic_vector (to_signed (153, 16))
report "errore in A*B"
severity warning;

myA <= std_logic_vector (to_signed (7549, 16));
myB <= std_logic_vector (to_signed (13773, 16));
codice <= "1000";
wait for 10 ns;
assert myZ <= std_logic_vector (to_signed (1586, 16))
report "errore in A*B"
severity warning;

myA <= std_logic_vector (to_signed (-5888, 16));
myB <= std_logic_vector (to_signed (2560, 16));
codice <= "1000";
wait for 10 ns;
assert myZ <= std_logic_vector (to_signed (-230, 16))
report "errore in A*B"
severity warning;

wait;
end process;

end test;
```

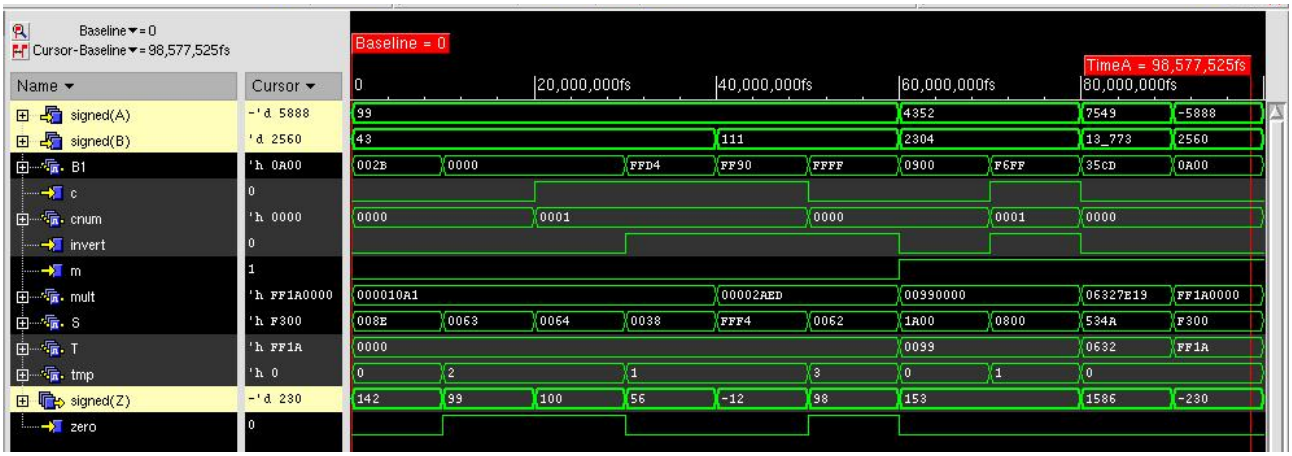


**Figura 1.2 – Simulazione unità aritmetica**

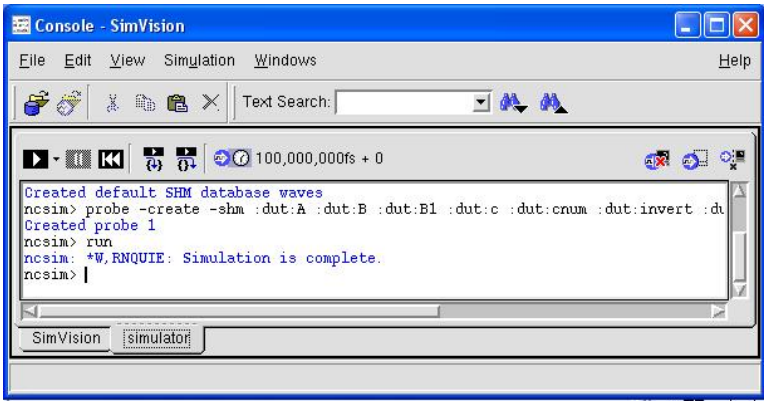


**Figura 1.3 – Console SimVision – Assenza di errori di simulazione**

Per quanto riguardala parte logica dell'ALU, è possibile implementare una qualsiasi funzione logica tra i segnali in ingresso A e B sfruttando un multiplexer come blocco logico universale.

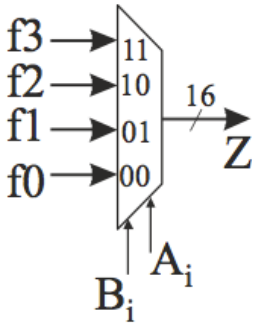


**Figura 1.4 – Multiplexer come blocco logico universale**

La funzione logica implementata dal multiplexer è la seguente:

$$Z = f_0\bar{a}\bar{b} + f_1 a\bar{b} + f_2\bar{a}b + f_3 ab$$

Il codice VHDL utilizzato per la descrizione dell'unità logica ed il codice per il suo test sono di seguito riportati:

```
--Architettura dei Sistemi Integrati * ESERCITAZIONE 2 * 22/03/2012 *
--Progettazione della parte logica della ALU
library ieee;

use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity bool is
       port (
              A : in std_logic_vector (15 downto 0);
              B : in std_logic_vector (15 downto 0);
              f3, f2, f1, f0 : in std_logic;
              W : out std_logic_vector (15 downto 0)
            );
end entity bool;

architecture dataflow of bool is

signal f3_16, f2_16, f1_16, f0_16: std_logic_vector (15 downto 0);

--Trasformo i bit di controllo in 16 bit per
--poter utilizzare le funzioni logiche previste dal linguaggio VHDL



begin
f3_16 <= (others => f3);
f2_16 <= (others => f2);
f1_16 <= (others => f1);
f0_16 <= (others => f0);

W <= (f3_16 AND A AND B) OR (f2_16 AND (NOT A) AND B) OR (f1_16 AND A AND (NOT B)) OR
(f0_16 AND (NOT A) AND (NOT B));

end dataflow;
--Architettura dei Sistemi Integrati * ESERCITAZIONE 2 * 22/03/2012 *
```

```
--Progettazione della parte logica della ALU - testbench



library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity tb_bool is
end tb_bool;

architecture test of tb_bool is

component bool is port (
             A : in std_logic_vector (15 downto 0);
             B : in std_logic_vector (15 downto 0);
             f3, f2, f1, f0 : in std_logic;
             W : out std_logic_vector (15 downto 0)
           );
end component;

signal myA, myB, myW : std_logic_vector (15 downto 0);
signal codice : std_logic_vector (3 downto 0);

begin

dut : bool port map (
A => myA,
B => myB,
f3 => codice(3),
f2 => codice(2),
f1 => codice(1),
f0 => codice(0),
W => myW );
stimulus : process

begin

myA <= b"0000_0011_0000_1111";
myB <= b"1010_1101_1111_0000";

codice <= "0000";
wait for 10 ns;
assert myW <= b"0000_0000_0000_0000"
report "errore in 0"
severity warning;

myA <= b"0000_0011_0000_1111";
myB <= b"1010_1101_1111_0000";
codice <= "0001";
wait for 10 ns;
assert myW <= b"0101_0000_0000_0000"
report "errore in A NOR B"
severity warning;

myA <= b"0000_0011_0000_1111";
myB <= b"1010_1101_1111_0000";
codice <= "0011";
wait for 10 ns;
assert myW <= b"0101_0010_0000_1111"
report "errore in 0011"
severity warning;

myA <= b"0000_0011_0000_1111";
myB <= b"1010_1101_1111_0000";
codice <= "0101";
wait for 10 ns;
```

```
assert myW <= b"1111_1100_1111_0000"
report "errore in 0101"
severity warning;

myA <= b"0000_0011_0000_1111";
myB <= b"1010_1101_1111_0000";
codice <= "0110";
wait for 10 ns;
assert myW <= b"1010_1110_1111_1111"
report "errore in A XOR B"
severity warning;

myA <= b"0000_0011_0000_1111";
myB <= b"1010_1101_1111_0000";
codice <= "1010";
wait for 10 ns;
assert myW <= b"0000_0011_0000_1111"
report "errore in 1010"
severity warning;

myA <= b"0000_0011_0000_1111";
myB <= b"1010_1101_1111_0000";
codice <= "1100";
wait for 10 ns;
assert myW <= b"1010_1101_1111_0000"
report "errore in 1100"
severity warning;


myA <= b"0000_0011_0000_1111";
myB <= b"1010_1101_1111_0000";
codice <= "1110";
wait for 10 ns;
assert myW <= b"1010_1111_1111_1111"
report "errore in A OR B"
severity warning;

myA <= b"0000_0011_0000_1111";
myB <= b"1010_1101_1111_0000";
codice <= "1111";
wait for 10 ns;
assert myW <= b"1111_1111_1111_1111"
report "errore in 1"
severity warning;

wait;

end process;

end test;
```

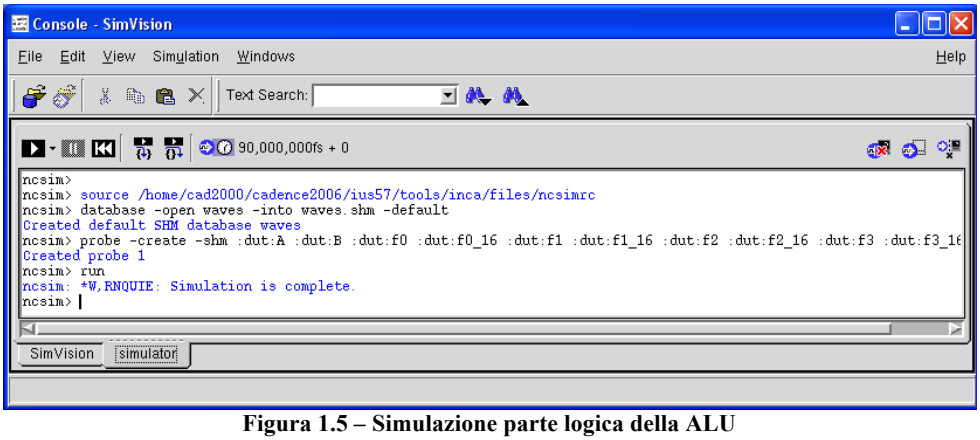


**Figura 1.5 – Simulazione parte logica della ALU**

La simulazione è eseguita correttamente, come dimostra l'assenza di warnings nella console di *SimVision* (figura 1.5). Per la progettazione della ALU finale, occorre dunque utilizzare una descrizione gerarchica che utilizzi insieme sia l'unità aritmetica che quella booleana definita. All'uopo, è stato progettato in VHDL un nuovo componente, ovvero un multplexer 2->1, che sarà utile nella descrizione della ALU come selettore dell'uscita, ovvero per poter scegliere se operare in modalità algebrica o logica.

```
--Architettura dei Sistemi Integrati * ESERCITAZIONE 2 * 22/03/2012 *
--Descrizione MUX2->1 che servirà nella definizione strutturale della
--ALU

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity Mux2 is
       port (
              a0 : in std_logic_vector (15 downto 0);
              b0 : in std_logic_vector (15 downto 0);
              s : in std_logic;
              z0 : out std_logic_vector (15 downto 0)
            );
end entity;

architecture dataflow of Mux2 is
begin
z0 <= a0 when s = '1' else
       b0;
end dataflow;

--Architettura dei Sistemi Integrati * ESERCITAZIONE 2 *
--Progettazione della ALU come unione dei componenti algebrici e
--logici in precedenza progettati

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity alu is
       port (
              A : in std_logic_vector (15 downto 0);
              B : in std_logic_vector (15 downto 0);
              opcode : in std_logic_vector (4 downto 0);
              W : out std_logic_vector (15 downto 0)
            );

end entity alu;

architecture structural of alu is

signal Wbool, Warit : std_logic_vector (15 downto 0);

component aritm
       port (
              A : in std_logic_vector (15 downto 0);
              B : in std_logic_vector (15 downto 0);
              zero, invert, c, m : in std_logic;
              Z : out std_logic_vector (15 downto 0)
            );
end component;
component bool
           port (
```

```
                A : in std_logic_vector (15 downto 0);
                B : in std_logic_vector (15 downto 0);
                f3, f2, f1, f0 : in std_logic;
                W : out std_logic_vector (15 downto 0)
             );

end component ;
component Mux2
       port (
              a0 : in std_logic_vector (15 downto 0);
              b0 : in std_logic_vector (15 downto 0);
              s : in std_logic;
              z0 : out std_logic_vector (15 downto 0)
            );

end component;
begin

bool1 : bool
          port map (
                    A => A, B => B, f3 => opcode(3), f2 => opcode(2),
                    f1 => opcode(1), f0 => opcode(0), W => Wbool
                    );
aritm1 : aritm
          port map (
                   A => A, B => B, zero => opcode(1),
                   m => opcode(3),
                   c => opcode(0),
                   invert => opcode(2),
                   Z => Warit
                   );
Mux21 : Mux2
          port map (
                   a0 => Warit, b0 => Wbool, s => opcode(4),
                   z0 => W
                   );

end structural;
```

La ALU è stata simulata semplicemente inserendo in cascata i test bench della parte aritmetica e della parte logica, con l'unica differenza che è necessario un ulteriore bit di controllo (*m*) che permette di selezionare la modalità (figura 1.6).

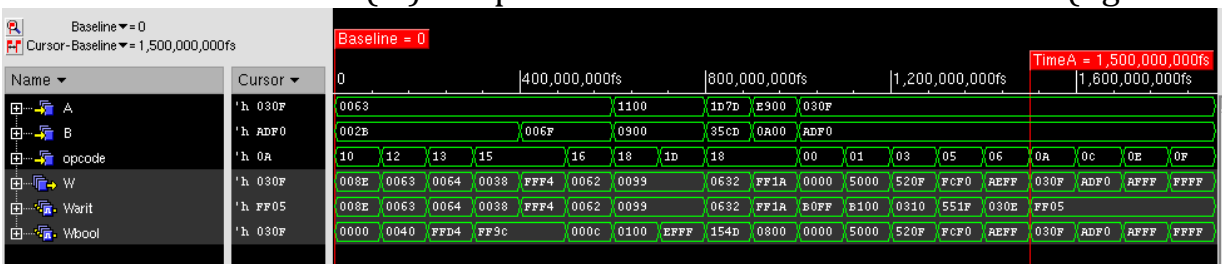


Figura 1.6 – Test bench della ALU

Come ulteriore verifica del lavoro svolto, è possibile utilizzare *SimVision* per visualizzare in forma schematica i blocchi realizzati. In questa maniera, sono ben distinguibili tutti blocchi utilizzati nella descrizione gerarchica della ALU (figura 1.7), così come i singoli blocchi logici utilizzati per implementare le entità logica e aritmetica in precedenza definite (figure 1.8 ed 1.9).

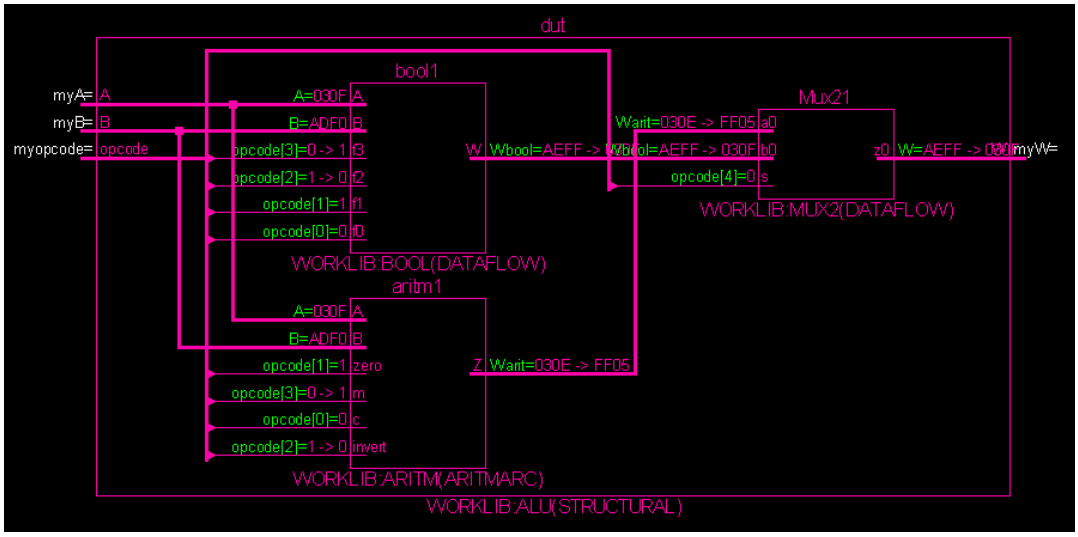


Figura 1.7 – Rappresentazione gerarchica della ALU

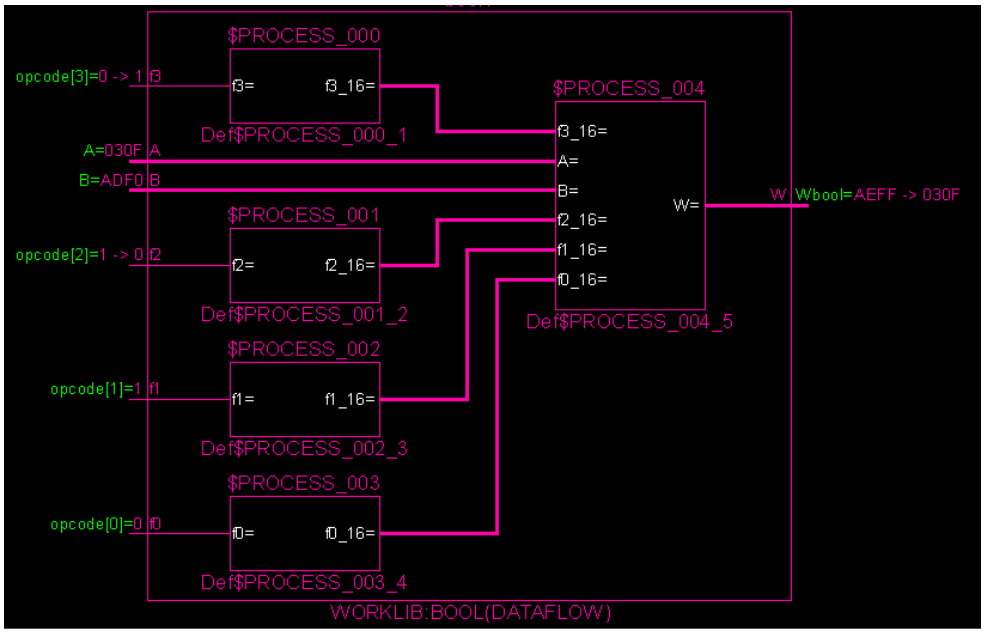


Figura 1.8 – Blocco logico

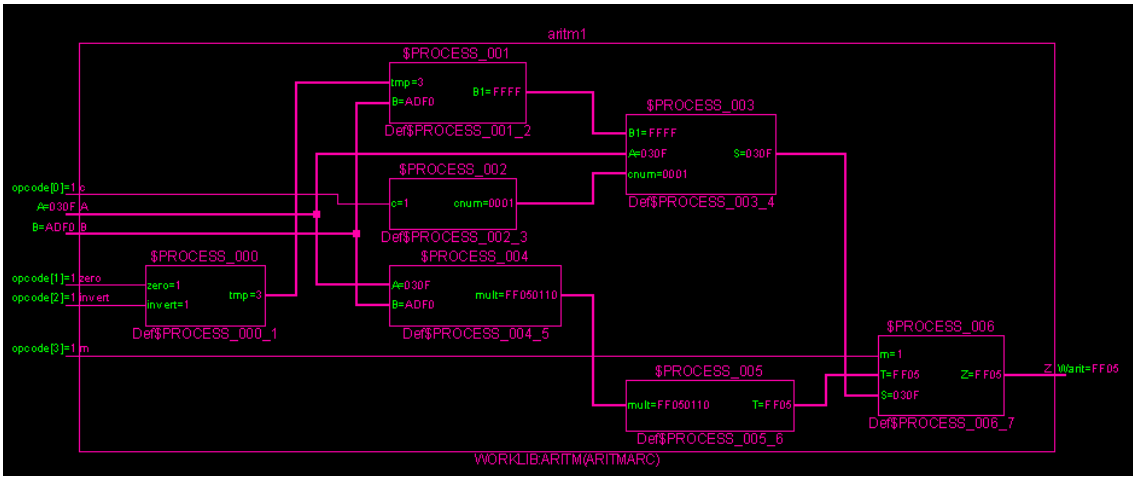


Figura 1.9 – Blocco algebrico

## 2. Sintesi della ALU

29/03/2012 - Esercitazione 3

Per sintetizzare la ALU progettata, è possibile utilizzare una libreria di celle standard, in formato *liberty* fornitoci dall'azienda Synopsys. Come software CAD, invece, è stato utilizzato *BuildGate Extreme Synthesis*. Con esso, è possibile utilizzare le celle di libreria per avere una prima stima di parametri fondamentali per la progettazione di sistemi digitali, quali potenza dissipata, occupazione di area e massima frequenza di funzionamento. Inoltre, il CAD permette di fissare diversi *constraints* nel flusso di progetto in modo da poter ottimizzare il circuito in base alle necessità del progettista. È possibile infatti ottimizzare il circuito per avere la massima velocità di funzionamento, la minima area occupata, e così via.
È possibile inoltre ottenere, grazie al software, accurate informazioni su area occupata o tempi di propagazione delle porte, già prima di definire i vincoli di progetto, ovvero in una sintesi *unconstrainted*. Attraverso il comando `report_area –hier –cells` è possibile conoscere l'occupazione di area dei blocchi logici utilizzati:

| Report | report_area |
|---|---|
| Options | -hier -cells > reports/area.rpt |
| Date | 20120329.152546 |
| Tool | bgx_shell64 |
| Release | v5.16-s014 |
| Version | Jan 27 2006 09:26:29 |
| Module | alu |

| Block report for module 'alu' | Current Module | Cumulative |
|---|---|---|
| Number of combinational instances | 0 | 1050 |
| Number of noncombinational instances | 0 | 0 |
| Number of hierarchical instances | 3 | 5 |
| Number of blackbox instances | 0 | 0 |
| Total number of instances | 3 | 1055 |
| Area of combinational cells | 0.00 | 1348.35 |
| Area of non-combinational cells | 0.00 | 0.00 |
| Total cell area | 0.00 | 1348.35 |
| Number of nets | 85 | 1162 |
| Area of nets | 0.00 | 0.00 |
| Total area | 0.00 | 1348.35 |

Cell Usage Table

| Cellref | Library | Number of Instances | Cell Type | Cell Area | Total Area |
|---|---|---|---|---|---|
| Mux2 | netlist | 1 | hier | 29.79 | 29.79 |
| aritm | netlist | 1 | hier | 1229.19 | 1229.19 |
| bool | netlist | 1 | hier | 89.38 | 89.38 |

| Block report for module 'Mux2' | Current Module | Cumulative |
|---|---|---|
| Number of combinational instances | 16 | 16 |
| Number of noncombinational instances | 0 | 0 |
| Number of hierarchical instances | 0 | 0 |
| Number of blackbox instances | 0 | 0 |
| Total number of instances | 16 | 16 |
| Area of combinational cells | 29.79 | 29.79 |
| Area of non-combinational cells | 0.00 | 0.00 |
| Total cell area | 29.79 | 29.79 |
| Number of nets | 49 | 49 |
| Area of nets | 0.00 | 0.00 |
| Total area | 29.79 | 29.79 |

Cell Usage Table

| Cellref | Library | Number of Instances | Cell Type | Cell Area | Total Area |
|---|---|---|---|---|---|
| MUX2_X1 | NangateOpenCellLibrary | 16 | comb | 1.86 | 29.79 |

| Block report for module 'aritm' | Current Module | Cumulative |
|---|---|---|
| Number of combinational instances | 51 | 986 |
| Number of noncombinational instances | 0 | 0 |
| Number of hierarchical instances | 2 | 2 |
| Number of blackbox instances | 0 | 0 |
| Total number of instances | 53 | 988 |
| Area of combinational cells | 65.97 | 1229.19 |
| Area of non-combinational cells | 0.00 | 0.00 |
| Total cell area | 65.97 | 1229.19 |
| Number of nets | 119 | 1097 |
| Area of nets | 0.00 | 0.00 |
| Total area | 65.97 | 1229.19 |

Cell Usage Table

| Cellref | Library | Number of Instances | Cell Type | Cell Area | Total Area |
|---|---|---|---|---|---|
| AWDP_MULT_1 | netlist | 1 | hier | 1079.43 | 1079.43 |
| AWDP_partition_0 | netlist | 1 | hier | 83.79 | 83.79 |
| INV_X1 | NangateOpenCellLibrary | 1 | comb | 0.53 | 0.53 |
| NOR2_X1 | NangateOpenCellLibrary | 1 | comb | 0.80 | 0.80 |
| MUX2_X1 | NangateOpenCellLibrary | 16 | comb | 1.86 | 29.79 |
| OAI211_X1 | NangateOpenCellLibrary | 16 | comb | 1.33 | 21.28 |
| NAND2_X1 | NangateOpenCellLibrary | 17 | comb | 0.80 | 13.57 |

| Block report for module 'bool' | Current Module | Cumulative |
|---|---|---|
| Number of combinational instances | 48 | 48 |
| Number of noncombinational instances | 0 | 0 |
| Number of hierarchical instances | 0 | 0 |
| Number of blackbox instances | 0 | 0 |
| Total number of instances | 48 | 48 |
| Area of combinational cells | 89.38 | 89.38 |
| Area of non-combinational cells | 0.00 | 0.00 |
| Total cell area | 89.38 | 89.38 |
| Number of nets | 84 | 84 |
| Area of nets | 0.00 | 0.00 |
| Total area | 89.38 | 89.38 |

| Cell Usage Table | | | | | |
|---|---|---|---|---|---|
| Cellref | Library | Number of Instances | Cell Type | Cell Area | Total Area |
| MUX2_X1 | NangateOpenCellLibrary | 48 | comb | 1.86 | 89.38 |

| Block report for module 'AWDP_MULT_1' | Current Module | Cumulative |
|---|---|---|
| Number of combinational instances | 864 | 864 |
| Number of noncombinational instances | 0 | 0 |
| Number of hierarchical instances | 0 | 0 |
| Number of blackbox instances | 0 | 0 |
| Total number of instances | 864 | 864 |
| Area of combinational cells | 1079.43 | 1079.43 |
| Area of non-combinational cells | 0.00 | 0.00 |
| Total cell area | 1079.43 | 1079.43 |
| Number of nets | 956 | 956 |
| Area of nets | 0.00 | 0.00 |
| Total area | 1079.43 | 1079.43 |

| Cell Usage Table | | | | | |
|---|---|---|---|---|---|
| Cellref | Library | Number of Instances | Cell Type | Cell Area | Total Area |
| MUX2_X2 | NangateOpenCellLibrary | 5 | comb | 1.86 | 9.31 |
| OR2_X2 | NangateOpenCellLibrary | 15 | comb | 1.06 | 15.96 |
| AND2_X2 | NangateOpenCellLibrary | 16 | comb | 1.06 | 17.02 |
| NOR2_X2 | NangateOpenCellLibrary | 16 | comb | 0.80 | 12.77 |
| NAND2_X2 | NangateOpenCellLibrary | 19 | comb | 0.80 | 15.16 |
| FA_X1 | NangateOpenCellLibrary | 44 | comb | 4.26 | 187.26 |
| XNOR2_X2 | NangateOpenCellLibrary | 52 | comb | 1.60 | 82.99 |
| XOR2_X2 | NangateOpenCellLibrary | 106 | comb | 1.60 | 169.18 |
| INV_X4 | NangateOpenCellLibrary | 271 | comb | 0.53 | 144.17 |
| AOI22_X2 | NangateOpenCellLibrary | 320 | comb | 1.33 | 425.60 |

| Block report for module 'AWDP_partition_0' | Current Module | Cumulative |
|---|---|---|
| Number of combinational instances | 71 | 71 |
| Number of noncombinational instances | 0 | 0 |
| Number of hierarchical instances | 0 | 0 |
| Number of blackbox instances | 0 | 0 |
| Total number of instances | 71 | 71 |
| Area of combinational cells | 83.79 | 83.79 |
| Area of non-combinational cells | 0.00 | 0.00 |
| Total cell area | 83.79 | 83.79 |
| Number of nets | 119 | 119 |
| Area of nets | 0.00 | 0.00 |
| Total area | 83.79 | 83.79 |

| Cell Usage Table | | | | | |
|---|---|---|---|---|---|
| Cellref | Library | Number of Instances | Cell Type | Cell Area | Total Area |
| XNOR2_X2 | NangateOpenCellLibrary | 8 | comb | 1.60 | 12.77 |
| AOI22_X2 | NangateOpenCellLibrary | 15 | comb | 1.33 | 19.95 |
| INV_X4 | NangateOpenCellLibrary | 24 | comb | 0.53 | 12.77 |
| XOR2_X2 | NangateOpenCellLibrary | 24 | comb | 1.60 | 38.30 |

Dai report, è evidente che dei tre “macroblocchi” utilizzati per la descrizione della ALU (blocco aritmetico, blocco logico e mux), quello che occupa più area è il blocco aritmetico. Focalizzando invece l’attenzione sulle singole celle di libreria, la più ingombrante utilizzata è la AOI22_X2, utilizzata nel blocco 'AWDP_MULT_1' (moltiplicatore) della parte aritmetica.
Per quanto riguarda invece i tempi di propagazione, attraverso il comando `report_timing –late –unconstrainted` è possibile visualizzare il cammino critico del dato all’interno del circuito:

| Report | report_timing |
|---|---|
| Options | -late -unconstrained > reports/timing_unconstrained.rpt |
| Date | 20120329.152255 |
| Tool | bgx_shell64 |
| Release | v5.16-s014 |
| Version | Jan 27 2006 09:26:29 |
| Module | alu |
| Timing | LATE |
| Slew Propagation | WORST |
| Operating Condition | typical |
| PVT Mode | max |
| Tree Type | balanced |
| Process | 1.00 |
| Voltage | 1.10 |
| Temperature | 25.00 |
| time unit | 1.00 ns |
| capacitance unit | 1.00 pF |
| resistance unit | 1.00 kOhm |

```
Path 1:Endpoint:   W[14] (v)
Beginpoint: B[1]  (v) (unconstrained input)
     Clock Rise Edge                       0.00
     + Input Delay                         0.00
     = Beginpoint Arrival Time             0.00
```

| Instance | Arc | Cell | Delay | Arrival Time |
|---|---|---|---|---|
| | B[1] v | | | 0.00 |
| aritm1 | B[1] v | aritm | | 0.00 |
| aritm1/i_12 | B[1] v | AWDP_MULT_1 | | 0.00 |
| aritm1/i_12/i_417 | B v -> ZN ^ | XNOR2_X2 | 0.26 | 0.26 |
| aritm1/i_12/i_426 | A ^ -> ZN v | INV_X4 | 0.09 | 0.35 |
| aritm1/i_12/i_439 | A1 v -> ZN ^ | AOI22_X2 | 0.09 | 0.44 |
| aritm1/i_12/i_440 | A ^ -> ZN v | INV_X4 | 0.01 | 0.46 |
| aritm1/i_12/i_441 | A1 v -> ZN ^ | AOI22_X2 | 0.06 | 0.51 |
| aritm1/i_12/i_199 | A ^ -> S v | FA_X1 | 0.14 | 0.65 |
| aritm1/i_12/i_200 | CI v -> S ^ | FA_X1 | 0.16 | 0.82 |
| aritm1/i_12/i_1139 | A ^ -> Z ^ | XOR2_X2 | 0.09 | 0.91 |
| aritm1/i_12/i_1141 | A ^ -> ZN v | INV_X4 | 0.01 | 0.92 |
| aritm1/i_12/i_1142 | A1 v -> ZN ^ | AOI22_X2 | 0.04 | 0.96 |
| aritm1/i_12/i_1147 | A2 ^ -> ZN v | AOI22_X2 | 0.03 | 0.99 |
| aritm1/i_12/i_1151 | B2 v -> ZN ^ | AOI22_X2 | 0.07 | 1.06 |
| aritm1/i_12/i_1156 | A2 ^ -> ZN v | AOI22_X2 | 0.03 | 1.09 |
| aritm1/i_12/i_1160 | B2 v -> ZN ^ | AOI22_X2 | 0.07 | 1.16 |
| aritm1/i_12/i_1165 | A2 ^ -> ZN v | AOI22_X2 | 0.03 | 1.19 |
| aritm1/i_12/i_1169 | B2 v -> ZN ^ | AOI22_X2 | 0.07 | 1.26 |
| aritm1/i_12/i_1174 | A2 ^ -> ZN v | AOI22_X2 | 0.03 | 1.29 |
| aritm1/i_12/i_1178 | B2 v -> ZN ^ | AOI22_X2 | 0.07 | 1.36 |
| aritm1/i_12/i_1183 | A2 ^ -> ZN v | AOI22_X2 | 0.04 | 1.40 |
| aritm1/i_12/i_1187 | B2 v -> ZN ^ | AOI22_X2 | 0.09 | 1.49 |
| aritm1/i_12/i_1192 | A2 ^ -> ZN v | AOI22_X2 | 0.04 | 1.53 |
| aritm1/i_12/i_1196 | B2 v -> ZN ^ | AOI22_X2 | 0.09 | 1.62 |
| aritm1/i_12/i_1201 | A2 ^ -> ZN v | AOI22_X2 | 0.04 | 1.66 |
| aritm1/i_12/i_1205 | B2 v -> ZN ^ | AOI22_X2 | 0.09 | 1.75 |
| aritm1/i_12/i_1210 | A2 ^ -> ZN v | AOI22_X2 | 0.04 | 1.79 |

```
| aritm1/i_12/i_1214 | B2 v -> ZN ^ | AOI22_X2     |  0.09 |    1.88 |
| aritm1/i_12/i_1219 | A2 ^ -> ZN v | AOI22_X2     |  0.04 |    1.92 |
| aritm1/i_12/i_1223 | B2 v -> ZN ^ | AOI22_X2     |  0.09 |    2.01 |
| aritm1/i_12/i_1228 | A2 ^ -> ZN v | AOI22_X2     |  0.04 |    2.05 |
| aritm1/i_12/i_1232 | B2 v -> ZN ^ | AOI22_X2     |  0.09 |    2.14 |
| aritm1/i_12/i_1237 | A2 ^ -> ZN v | AOI22_X2     |  0.04 |    2.18 |
| aritm1/i_12/i_1241 | B2 v -> ZN ^ | AOI22_X2     |  0.09 |    2.27 |
| aritm1/i_12/i_1246 | A2 ^ -> ZN v | AOI22_X2     |  0.04 |    2.31 |
| aritm1/i_12/i_1248 | B v -> Z v   | XOR2_X2      |  0.09 |    2.40 |
| aritm1/i_12        | mult[30] v   | AWDP_MULT_1  |       |    2.40 |
| aritm1/i_14        | B v -> Z v   | MUX2_X1      |  0.11 |    2.51 |
| aritm1             | Z[14] v      | aritm        |       |    2.51 |
| Mux21              | a0[14] v     | Mux2         |       |    2.51 |
| Mux21/i_14         | B v -> Z v   | MUX2_X1      |  0.11 |    2.62 |
| Mux21              | z0[14] v     | Mux2         |       |    2.62 |
|                    | W[14] v      |              |  0.00 |    2.62 |
+-------------------------------------------------------------------+
```

Il cammino critico risulta dunque essere quello che, nel blocco aritmetico, percorre il dato dall'ingresso B[1] all'uscita W[14].
Si prosegue dunque assegnando al sintetizzatore dei vincoli riguardanti la velocità del circuito realizzato. A tal fine, è stato fornito il file constraints.tcl:

```
# constraints.tcl

# Rimuovo eventuali vincoli precedenti
remove_assertions [find -ports]

# Definisco un clock idealde, di periodo 10ns
create_clock -period $periodo -name my_clk

# Definisco il ritardo della logica di ingresso e di uscita
set_input_delay  -clock my_clk -max 0.100 [all_inputs]
set_output_delay -clock my_clk -max 0.150 [all_outputs]

# Tempo di salita dei segnali di ingresso: 50ps
set_input_transition 0.050 [all_inputs]

# Carico in uscita: 10fF
set_load 0.010 [all_outputs]
```

Sono evidenti dunque parametri fondamentali come periodo del clock, ritardo dei segnali di ingresso, tempo di salita degli stessi, e capacità da pilotare in uscita. Attraverso questi parametri, il sintetizzatore opera una nuova timing analysis:

```
+-------------------------------------------------------------------+
| Report               | report_timing                              |
|----------------------+--------------------------------------------|
| Options              | -late > reports/timing_constrained_late.rpt |
+----------------------+--------------------------------------------+
| Date                 | 20120329.153805                            |
| Tool                 | bgx_shell64                                |
| Release              | v5.16-s014                                 |
| Version              | Jan 27 2006 09:26:29                       |
+----------------------+--------------------------------------------+
| Module               | alu                                        |
| Timing               | LATE                                       |
| Slew Propagation     | WORST                                      |
| Operating Condition | typical                                    |
| PVT Mode             | max                                        |
| Tree Type            | balanced                                   |
| Process              | 1.00                                       |
| Voltage              | 1.10                                       |
| Temperature          | 25.00                                      |
| time unit            | 1.00 ns                                    |
| capacitance unit     | 1.00 pF                                    |
| resistance unit      | 1.00 kOhm                                  |
+-------------------------------------------------------------------+
Path 1: MET External Delay Assertion
Endpoint:   W[14] (v) checked with  leading edge of 'my_clk'
Beginpoint: B[1]  (v) triggered by  leading edge of 'my_clk'
```

```
Other End Arrival Time          0.00
- External Delay                0.15
+ Phase Shift                  10.00
= Required Time                 9.85
- Arrival Time                  2.82
= Slack Time                    7.03
     Clock Rise Edge                     0.00
     + Input Delay                       0.10
     = Beginpoint Arrival Time           0.10
```

Parametro fondamentale è lo slack: in questo caso esso è positivo e dunque non ci sono violazioni di timing.
Se provassimo a modificare, per esempio, il tempo di salita dei segnali in ingresso, i ritardi dei segnali sarebbero differenti :

```
+-------------------------------------------------------------------+
| Report              | report_timing                               |
|---------------------+---------------------------------------------|
| Options             | -late > reports/timing_constrained_late_150p.rpt |
+---------------------+---------------------------------------------+
| Date                | 20120329.154256                             |
| Tool                | bgx_shell64                                 |
| Release             | v5.16-s014                                  |
| Version             | Jan 27 2006 09:26:29                        |
+---------------------+---------------------------------------------+
| Module              | alu                                         |
| Timing              | LATE                                        |
| Slew Propagation    | WORST                                       |
| Operating Condition | typical                                     |
| PVT Mode            | max                                         |
| Tree Type           | balanced                                    |
| Process             | 1.00                                        |
| Voltage             | 1.10                                        |
| Temperature         | 25.00                                       |
| time unit           | 1.00 ns                                     |
| capacitance unit    | 1.00 pF                                     |
| resistance unit     | 1.00 kOhm                                   |
+-------------------------------------------------------------------+
Path 1: MET External Delay Assertion
Endpoint:   W[14] (v) checked with  leading edge of 'my_clk'
Beginpoint: B[1]  (v) triggered by  leading edge of 'my_clk'
Other End Arrival Time          0.00
- External Delay                0.15
+ Phase Shift                  10.00
= Required Time                 9.85
- Arrival Time                  2.86
= Slack Time                    6.99
     Clock Rise Edge                     0.00
     + Input Delay                       0.10
     = Beginpoint Arrival Time           0.10
```

Come è possibile notare, il tempo di arrivo richiesto dai segnali è leggermente maggiore, e ciò influisce negativamente sullo slack. Il discorso è identico se viene triplicata la capacità in uscita da pilotare:

```
+-------------------------------------------------------------------+
| Report              | report_timing                               |
|---------------------+---------------------------------------------|
| Options             | -late > reports/timing_constrained_late_dload.rpt |
+---------------------+---------------------------------------------+
| Date                | 20120329.160757                             |
| Tool                | bgx_shell64                                 |
| Release             | v5.16-s014                                  |
| Version             | Jan 27 2006 09:26:29                        |
+---------------------+---------------------------------------------+
| Module              | alu                                         |
| Timing              | LATE                                        |
| Slew Propagation    | WORST                                       |
| Operating Condition | typical                                     |
| PVT Mode            | max                                         |
| Tree Type           | balanced                                    |
```

```
| Process             | 1.00                                                  |
| Voltage             | 1.10                                                  |
| Temperature         | 25.00                                                 |
| time unit           | 1.00 ns                                               |
| capacitance unit    | 1.00 pF                                               |
| resistance unit     | 1.00 kOhm                                             |
+-----------------------------------------------------------------------+
Path 1: MET External Delay Assertion
Endpoint:   W[14] (v) checked with  leading edge of 'my_clk'
Beginpoint: B[1]  (v) triggered by  leading edge of 'my_clk'
Other End Arrival Time          0.00
- External Delay                0.15
+ Phase Shift                 10.00
= Required Time                 9.85
- Arrival Time                  2.88
= Slack Time                    6.97
     Clock Rise Edge                     0.00
     + Input Delay                       0.10
     = Beginpoint Arrival Time           0.10
```

Se invece venisse portato il periodo del clock a 2,5 ns, il vincolo sul timing non sarebbe più rispettato, e ciò si rifletterebbe in uno slack negativo, come testimonia il report ottenuto dalla timing analysis:

```
+-----------------------------------------------------------------------+
| Report              | report_timing                                   |
|---------------------+-------------------------------------------------|
| Options             | -late > reports/timing_constrained_late_neg_slack.rpt |
+---------------------+-------------------------------------------------+
| Date                | 20120329.160945                                 |
| Tool                | bgx_shell64                                     |
| Release             | v5.16-s014                                      |
| Version             | Jan 27 2006 09:26:29                            |
+---------------------+-------------------------------------------------+
| Module              | alu                                             |
| Timing              | LATE                                            |
| Slew Propagation    | WORST                                           |
| Operating Condition | typical                                         |
| PVT Mode            | max                                             |
| Tree Type           | balanced                                        |
| Process             | 1.00                                            |
| Voltage             | 1.10                                            |
| Temperature         | 25.00                                           |
| time unit           | 1.00 ns                                         |
| capacitance unit    | 1.00 pF                                         |
| resistance unit     | 1.00 kOhm                                       |
+-----------------------------------------------------------------------+
Path 1: VIOLATED External Delay Assertion
Endpoint:   W[14] (v) checked with  leading edge of 'my_clk'
Beginpoint: B[1]  (v) triggered by  leading edge of 'my_clk'
Other End Arrival Time          0.00
- External Delay                0.15
+ Phase Shift                  2.50
= Required Time                 2.35
- Arrival Time                  2.82
= Slack Time                   -0.47
     Clock Rise Edge                     0.00
     + Input Delay                       0.10
     = Beginpoint Arrival Time           0.10
```

È dunque necessario eseguire un'ulteriore ottimizzazione del circuito, chiedendo al sintetizzatore di occupare celle più ingombranti rispettando al contempo il vicolo sul periodo del clock. L'ottimizzazione di area viene eseguita attraverso il comando `do_optimize -reclaim_maximum_area`.
Dalla successiva timing analysis risulta che il constraint sul tempo è adesso rispettato:

```
+-------------------------------------------------------------------------------+
| Report              | report_timing                                           |
|---------------------+---------------------------------------------------------|
| Options             | -late > reports/timing_constrained_late_good_slack.rpt  |
+---------------------+---------------------------------------------------------+
| Date                | 20120329.161105                                         |
| Tool                | bgx_shell64                                             |
| Release             | v5.16-s014                                              |
| Version             | Jan 27 2006 09:26:29                                    |
+---------------------+---------------------------------------------------------+
| Module              | alu                                                     |
| Timing              | LATE                                                    |
| Slew Propagation    | WORST                                                   |
| Operating Condition | typical                                                 |
| PVT Mode            | max                                                     |
| Tree Type           | balanced                                                |
| Process             | 1.00                                                    |
| Voltage             | 1.10                                                    |
| Temperature         | 25.00                                                   |
| time unit           | 1.00 ns                                                 |
| capacitance unit    | 1.00 pF                                                 |
| resistance unit     | 1.00 kOhm                                               |
+-------------------------------------------------------------------------------+
Path 1: MET External Delay Assertion
Endpoint:   W[15] (v) checked with  leading edge of 'my_clk'
Beginpoint: B[0]  (v) triggered by  leading edge of 'my_clk'
Other End Arrival Time          0.00
- External Delay                0.15
+ Phase Shift                   2.50
= Required Time                 2.35
- Arrival Time                  2.31
= Slack Time                    0.04
     Clock Rise Edge                         0.00
     + Input Delay                           0.10
     = Beginpoint Arrival Time               0.10
```

Per rispettare tale vincolo sono però state usate celle logiche più grandi, come testimonia una nuova analisi sull'area occupata:

```
+-----------------------------------------------+
| Report  | report_area                         |
|---------+-------------------------------------|
| Options | -hier -cells > reports/new_area.rpt |
+---------+-------------------------------------+
| Date    | 20120329.161211                     |
| Tool    | bgx_shell64                         |
| Release | v5.16-s014                          |
| Version | Jan 27 2006 09:26:29                |
+---------+-------------------------------------+
| Module  | alu                                 |
+-----------------------------------------------+

+-------------------------------------------------------------+
|    Block report for module 'alu'     | Current | Cumulative |
|                                      |  Module |            |
|--------------------------------------+---------+------------|
| Number of combinational instances    |       0 |       1093 |
| Number of noncombinational instances |       0 |          0 |
| Number of hierarchical instances     |       3 |          5 |
| Number of blackbox instances         |       0 |          0 |
| Total number of instances            |       3 |       1098 |
| Area of combinational cells          |    0.00 |    1421.77 |
| Area of non-combinational cells      |    0.00 |       0.00 |
| Total cell area                      |    0.00 |    1421.77 |
| Number of nets                       |      85 |       1205 |
| Area of nets                         |    0.00 |       0.00 |
| Total area                           |    0.00 |    1421.77 |
+-------------------------------------------------------------+
```

Confrontando la nuova area totale della ALU con quella ottenuta nella sintesi *unconstrainted*, è possibile notare un incremento del 5,44% dell'area occupata.
Infine, è possibile analizzare differenti parametri del circuito al variare del periodo di clock, per avere una più accurata visione della variazione delle prestazioni del circuito.

| Periodo | Slack [ns] | Minimo periodo di funzionamento [ns] | Area [$\mu m^2$] | Architettura moltiplicatore e addizionatore |
|---|---|---|---|---|
| 5 ns | 2,03 | 2,97 | 1348,35 | ripple – ripple/booth |
| 2,5 ns | 0,03 | 2,47 | 1419,38 | ripple – ripple/booth |
| 2,2 ns | 0,02 | 2,18 | 1386,13 | ripple – cla/booth |
| 2 ns | 0,00 | 2,00 | 1381,60 | ripple – cla/booth |
| 1,8 ns | 0,00 | 1,80 | 1595,63 | ripple – fcla/booth |
| 1,6 ns | -0,00 | 1,60 | 1690,16 | cla – fcla/booth |

## 3. Progettazione datapath

12/04/2012 - Esercitazione 4

Il datapath è un sottoinsieme del processore, che include la ALU sviluppata in precedenza, il registro che contiene il dato in ingresso dalla memoria, l'accumulatore che immagazzina il risultato prodotto dall'ALU ed un semplice blocco combinatorio che fornisce un'uscita alta qualora il valore dell'accumulatore sia zero.

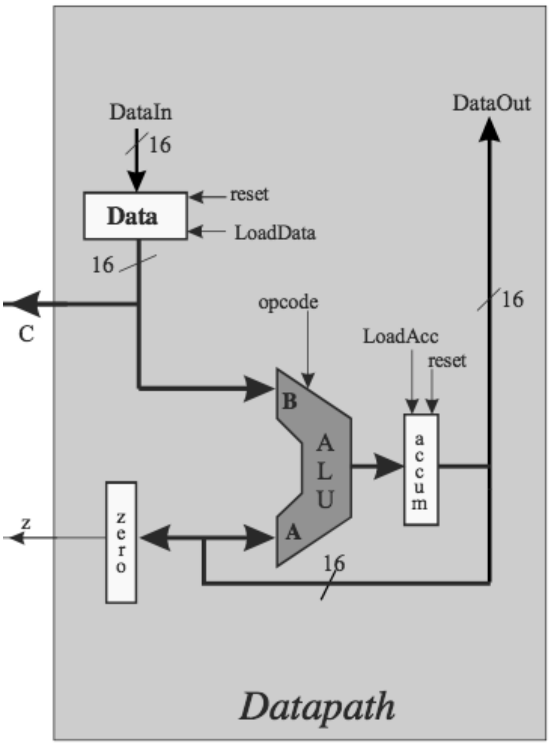


**Figura 3.1 – Datapath del *Pico16***

Nella descrizione VHDL del datapath, è stata utilizzata una descrizione ibrida, vale a dire strutturale e procedurale contemporaneamente. Il codice è di seguito riportato:

```
--Architettura dei Sistemi Integrati * ESERCITAZIONE 4 * 12/04/2012 *
--Progettazione del datapath del microporcessore
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity datapath is
	port (
		DataIn : in std_logic_vector (15 downto 0);
		opcode : in std_logic_vector (4 downto 0);
		reset, LoadData, LoadAcc, clk : in std_logic;
		z      : out std_logic;
		C, DataOut : out std_logic_vector (15 downto 0)
	      );
end entity datapath;

--Utilizzo una descrizione mista, con una parte strutturale ed una parte comportamentale
architecture dp of datapath is

--Dichiaro dei segnali ausiliari da utilizzare nella descrizione
signal W : std_logic_vector (15 downto 0);  --Uscita della ALU, ingresso del registro "accum"
signal A : std_logic_vector (15 downto 0);  --In ingresso alla ALU, uscita del registro "accum"
signal B : std_logic_vector (15 downto 0);  --In ingresso alla ALU, uscia del registro "Data"

component alu
	port (
		A : in std_logic_vector (15 downto 0);
		B : in std_logic_vector (15 downto 0);
		opcode : in std_logic_vector (4 downto 0);
		W : out std_logic_vector (15 downto 0)
	);
end component;

begin

alu1 : alu
	port map (
		A => A, B => B,  W => W, opcode => opcode
);

--Descrizione dei registri
process (clk)
begin
if rising_edge(clk) then
	--Registro Data
		if (reset='1') then
		B <= b"0000_0000_0000_0000";
		elsif (LoadData='1') then
		B <= DataIn;
		end if;
	--Registro accum
		if (reset='1') then
		A <= b"0000_0000_0000_0000";
		elsif (LoadAcc='1') then
		A <= W;
		end if;
end if;
end process;

--Descrizione del blocco zero
	z <= '1' when A = b"0000_0000_0000_0000" else
	     '0';
--Assegnazione dei terminali di uscita
	C <= B;
	DataOut <= A;

end dp;
```

Il datapath così definito è stato simulato utilizzando il test bench fornito, ed i risultati sono stati verificati controllando l'andamento *waveform* dei segnali din ingresso e di uscita.

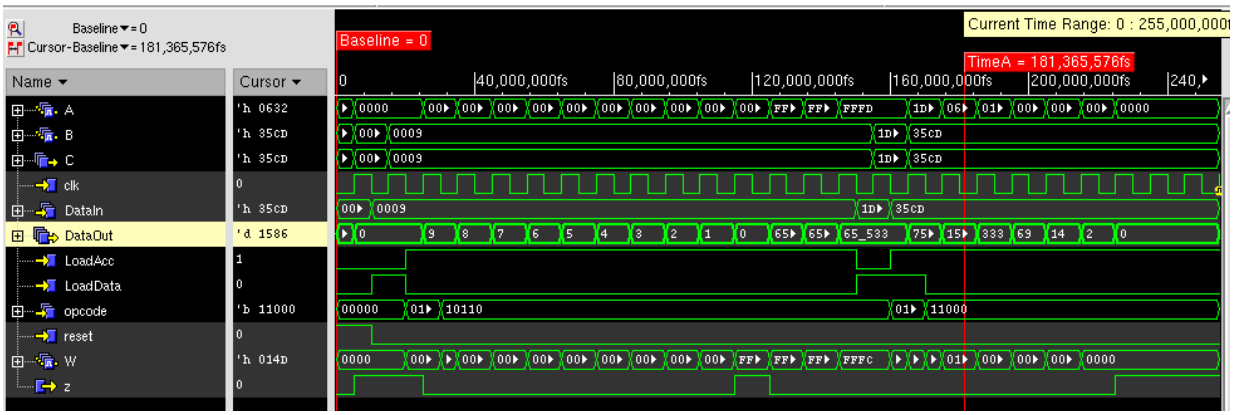


**Figura 3.2 – Simulazione datapath**

Si procede dunque alla sintesi del datapath, realizzata impostando dei ritardi sia minimi che massimi sulla logica di ingresso e di uscita.
In primo luogo, si realizza la sintesi per controllare che siano verificati i vincoli sui tempi di setup e di hold. Per fare ciò, è utilizzato lo stesso script di automazione utilizzato nell'esercitazione 3, settando come periodo di clock 2.5 ns. Attraverso il comando `report_timing -late`, è possibile controllare il soddisfacimento del vincolo sul ritardo massimo, ovvero sul tempo di setup:

```
+-------------------------------------------------------------------------------------+
|                                Sequential Elements                                  |
|-------------------------------------------------------------------------------------|
|  Module  |       File        | Line |  Name  | Type | Width | AS | AR | SS | SR |
+-------------------------------------------------------------------------------------+
| datapath | hdl/datapath.vhd | 44   | A_reg | FF   | 16    | N  | N  | N  | N  |
| datapath | hdl/datapath.vhd | 44   | B_reg | FF   | 16    | N  | N  | N  | N  |
+-------------------------------------------------------------------------------------+
            Finished processing module: 'datapath' <MODGEN-110>.
    Info:   Setting 'datapath' as the top of the design hierarchy <FNP-704>.
    Info:   Setting 'datapath' as the default top timing module <FNP-705>.
bgx_shell64[8]>
```

**Figura 3.3 – Il sintetizzatore riconosce la presenza di registri nella descrizione VHDL**

```
+-----------------------------------------------------+
| Report               | report_timing                |
|----------------------+------------------------------|
| Options              | -late > reports/timing_late.txt |
+----------------------+------------------------------+
| Date                 | 20120412.160414              |
| Tool                 | bgx_shell64                  |
| Release              | v5.16-s014                   |
| Version              | Jan 27 2006 09:21:08         |
+----------------------+------------------------------+
| Module               | datapath                     |
| Timing               | LATE                         |
| Slew Propagation     | WORST                        |
| Operating Condition  | typical                      |
| PVT Mode             | max                          |
| Tree Type            | balanced                     |
| Process              | 1.00                         |
| Voltage              | 1.10                         |
| Temperature          | 25.00                        |
```

```
| time unit           | 1.00 ns                          |
| capacitance unit    | 1.00 pF                          |
| resistance unit     | 1.00 kOhm                        |
+-----------------------------------------------------+
Path 1: MET Setup Check with Pin A_reg_11/CK
Endpoint:   A_reg_11/D (v) checked with  leading edge of 'my_clk'
Beginpoint: B_reg_11/Q (v) triggered by  leading edge of 'my_clk'
Other End Arrival Time          0.00
- Setup                         0.03
+ Phase Shift                   2.50
= Required Time                 2.47
- Arrival Time                  2.45
= Slack Time                    0.02
     Clock Rise Edge                      0.00
     = Beginpoint Arrival Time            0.00
```

Lo slack positivo è indice che non ci sono problemi di temporizzazione del circuito. In particolare, si può notare che il percorso critico è quello che va dal bit 11 di B (uscita del registro Data ed ingresso della ALU) al bit 11 di A (uscita del registro accumulatore).

Per quanto riguarda il vincolo sul tempo di hold, è invece necessario eseguire un'analisi dei ritardi minimi, attraverso il comando `report_timing -early`.

```
+-----------------------------------------------------+
| Report              | report_timing                 |
|---------------------+-------------------------------|
| Options             | -early > reports/timing_early.txt |
+---------------------+-------------------------------+
| Date                | 20120412.160414               |
| Tool                | bgx_shell64                   |
| Release             | v5.16-s014                    |
| Version             | Jan 27 2006 09:21:08          |
+---------------------+-------------------------------+
| Module              | datapath                      |
| Timing              | EARLY                         |
| Slew Propagation    | WORST                         |
| Operating Condition | fast                          |
| PVT Mode            | min                           |
| Tree Type           | balanced                      |
| Process             | 1.00                          |
| Voltage             | 1.25                          |
| Temperature         | 0.00                          |
| time unit           | 1.00 ns                       |
| capacitance unit    | 1.00 pF                       |
| resistance unit     | 1.00 kOhm                     |
+-----------------------------------------------------+
Path 1: MET Hold Check with Pin A_reg_0/CK
Endpoint:   A_reg_0/D (^) checked with  leading edge of 'my_clk'
Beginpoint: A_reg_0/Q (^) triggered by  leading edge of 'my_clk'
Other End Arrival Time          0.00
+ Hold                         0.03
+ Phase Shift                  0.00
= Required Time                0.03
  Arrival Time                 0.19
  Slack Time                   0.17
     Clock Rise Edge                      0.00
     = Beginpoint Arrival Time            0.00
```

Anche in questo caso si ha uno slack positivo, e dunque non è necessario utilizzare dispositivi più piccoli o inserire dei buffer per far si che il vincolo sul tempo di hold sia rispettato.

Per un'analisi più accurata, è dunque opportuno caratterizzare il circuito in presenza di non idealità del segnale di clock, quali possono essere jitter o skew. Per portare in conto di questi fenomeni in fase di sintesi, è possibile assegnare un opportuno comando al sintetizzatore che inserisce un'incertezza di 100ps sui tempi di arrivo dei fronti del clock. Eseguendo una nuova timing analysis, è

possibile notare che la presenza di queste non idealità modifica sostanzialmente l'analisi sul ritardo massimo:

```
+-------------------------------------------------------------+
| Report              | report_timing                         |
|---------------------+---------------------------------------|
| Options             | -late > reports/timing_late_nonideale.txt |
+---------------------+---------------------------------------+
| Date                | 20120412.161037                       |
| Tool                | bgx_shell64                           |
| Release             | v5.16-s014                            |
| Version             | Jan 27 2006 09:21:08                  |
+---------------------+---------------------------------------+
| Module              | datapath                              |
| Timing              | LATE                                  |
| Slew Propagation    | WORST                                 |
| Operating Condition | typical                               |
| PVT Mode            | max                                   |
| Tree Type           | balanced                              |
| Process             | 1.00                                  |
| Voltage             | 1.10                                  |
| Temperature         | 25.00                                 |
| time unit           | 1.00 ns                               |
| capacitance unit    | 1.00 pF                               |
| resistance unit     | 1.00 kOhm                             |
+-------------------------------------------------------------+
Path 1: MET Setup Check with Pin A_reg_14/CK
Endpoint:   A_reg_14/D (v) checked with  leading edge of 'my_clk'
Beginpoint: B_reg_7/Q  (v) triggered by  leading edge of 'my_clk'
Other End Arrival Time          0.00
- Setup                         0.03
+ Phase Shift                   2.50
- Uncertainty                   0.10
= Required Time                 2.37
- Arrival Time                  2.37
= Slack Time                    0.00
     Clock Rise Edge                       0.00
     = Beginpoint Arrival Time             0.00
```

Non solo lo slack, infatti, si è portato al valore limite di 0.00ns, ma anche il percorso critico si è modificato: adesso è il percorso che va dal bit 7 di B al bit 14 di A. È possibile fare considerazioni analoghe anche riguardo il timing sul ritardo minimo. A questo punto, è creato un file output contenente la netlist VHDL del circuito sintetizzato, in modo da poter eseguire il place & route e la simulazione post-layout.

Analizzando il file *circuit.sdf*, creato attraverso lo script *write_output*,è possibile notare ogni porta logica elementare che viene istanziata, così come i tempi di salita e di discesa dei segnali in ingresso ed in uscita alla porta. In particolare, si può notare che per la stessa porta logica istanziata, è possibile avere ritardi differenti, questo poiché ad influire sui ritardi non è solo la porta logica stessa ma anche parametri come ritardi degli altri segnali del circuito oppure capacità in uscita che la porta deve pilotare. A questo punto del progetto, è possibile dunque eseguire una simulazione che tenga conto dei ritardi, utilizzando il software *nclaunch*. In realtà, il ritardo individuato non corrisponde esattamente a quello indicato dal path critico poiché è molto improbabile che nel test bench venga eccitato proprio tale percorso del dato. In ogni caso, i ritardi individuati sono coerenti con quanto fornito dal software di sintesi.

## 4. Placement & Routing del Datapath

26/04/2012 - Esercitazione 5

Per completare la progettazione del *datapath*, viene dunque eseguita la fase finale di P&R, attraverso il software *SoC Encounter* della Cadence. Durante questa fase, non solo viene stabilita, appunto, la disposizione delle celle logiche elementari e il collegamento tra le stesse, ma vengono anche eseguite ulteriori operazioni atte a migliorare l'efficienza del circuito stesso, quali la definizione ed il piazzamento dell'albero di clock, ulteriori miglioramenti *in-place*.
La prima fase da eseguire è il *Floorplan*. In essa, è definita la forma geometrica del circuito che verrà realizzato, così come le linee di alimentazione e massa. In particolare, si è deciso di utilizzare una forma rettangolare, con rapporto lunghezza/altezza pari a 0,7, in cui le celle logiche occupino solamente il 60% dello spazio disponibile. Lo "spazio vuoto", infatti, servirà in seguito per l'inserimento dell'albero di clock. Si utilizzano, inoltre, tre coppie di linee di alimentazione e massa. Il risultato del floorplan è riportato in figura 4.1.

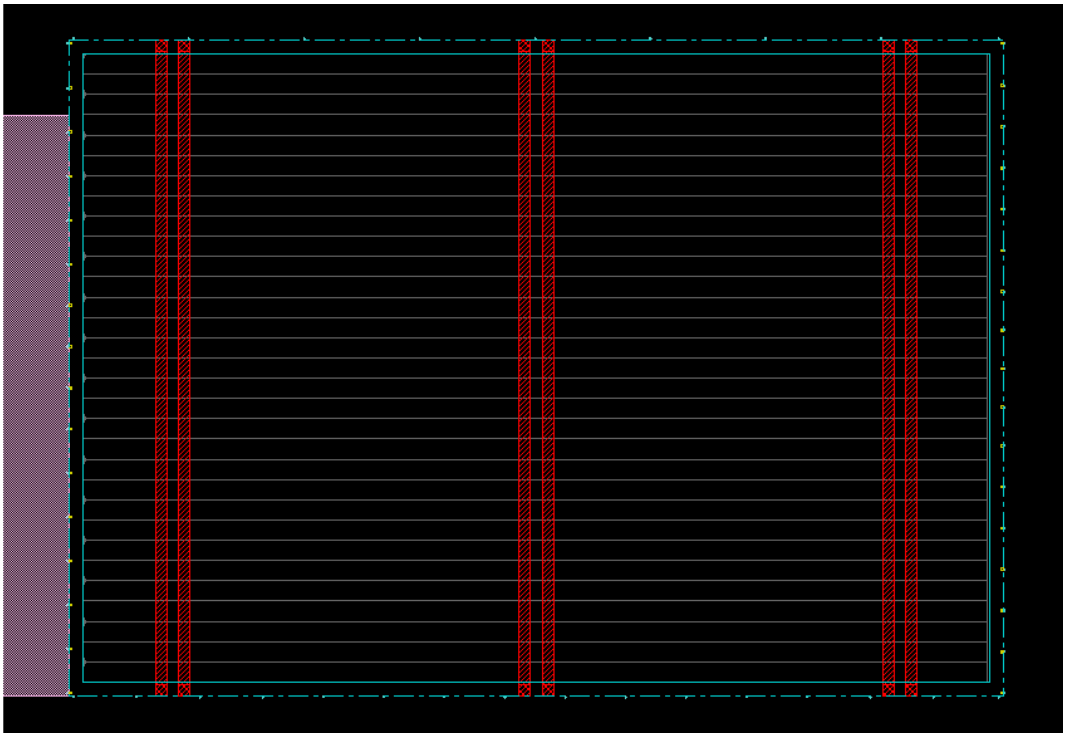

**Figura 4.1 – Floorplan del circuito**

La fase successiva è quella di *Placement*. In questa fase sono disposte le celle logiche e vengono anche realizzati dei collegamenti "fittizi" utilizzati per ottenere una stima del carico capacitivo delle interconnessioni. In figura 4.2 è riportato il risultato dell'operazione di *placement* delle celle logiche. È evidente lo spazio non occupato da alcuna cella logica.

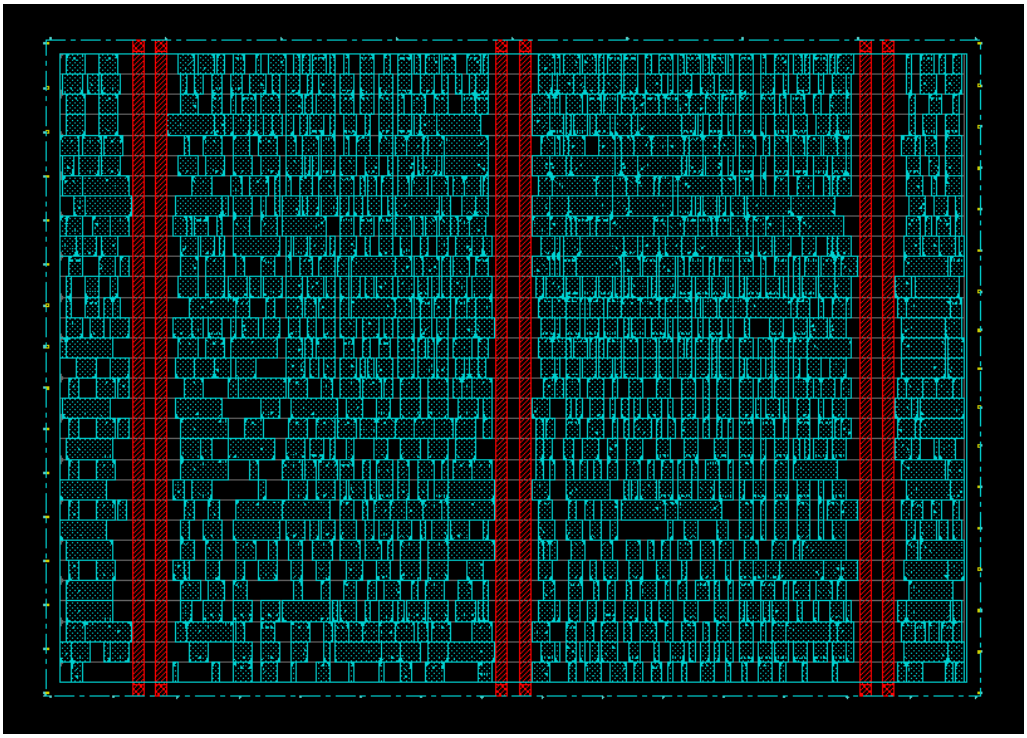

**Figura 4.2 – Placement celle logiche**

Il software *Encounter* permette inoltre, dopo la fase di *placing*, di eseguire una nuova ottimizzazione detta *in-place*. È possibile, infatti, fornire al tool dei constraints temporali che esso cercherà di rispettare modificando l'occupazione di area del circuito. In seguito alla fase di ottimizzazione, in cui si è richiesto di ottenere uno slack target di 100 ps, l'area occupata, rispetto alla prima fase di *placing*, sarà, infatti, incrementata, come è possibile verificare analizzando i *Summary Report* forniti dal software:

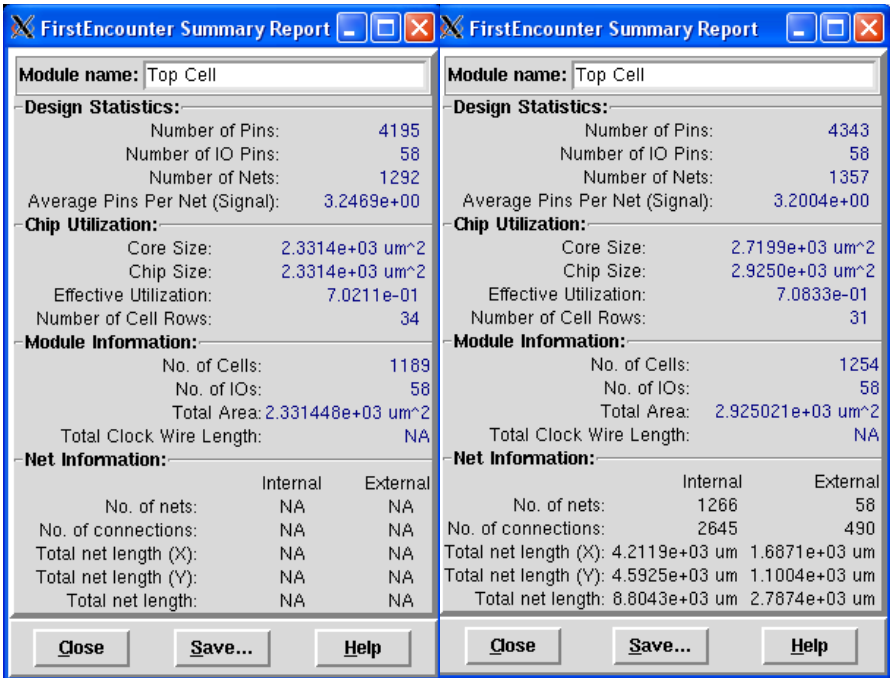


**Figura 4.3 – Area utilizzata prima e dopo l'ottimizzazione *in-place***

È possibile analizzare le stime dei ritardi e dei percorsi critici individuate dal tool durante le diverse fasi del progetto:

```
PRE_PLACE----------------------------------------------------------------------------------------------------
Path #: 1
Startpoint: B_reg_7/Q
            (clocked by my_clk R)
Endpoint: A_reg_11/D
            (Setup time: 0.029, clocked by my_clk R)
Data required time: 2.371 (minus uncertainty: 0.100)
Data arrival time: 2.103
Slack: 0.268

----------------------------------------------------------------------------------------------------

PRE_CTS------------------------------------------------------------------------------------------------
Path #: 1
Startpoint: B_reg_9/Q
            (clocked by my_clk R)
Endpoint: A_reg_3/D
            (Setup time: 0.030, clocked by my_clk R)
Data required time: 2.370 (minus uncertainty: 0.100)
Data arrival time: 2.465
Slack: -0.095 (SETUP VIOLATION)

----------------------------------------------------------------------------------------------------

PRE_CTS_OPTIIZED---------------------------------------------------------------------------------------
Path #: 1
Startpoint: B_reg_9/Q
            (clocked by my_clk R)
Endpoint: A_reg_3/D
            (Setup time: 0.029, clocked by my_clk R)
Data required time: 2.371 (minus uncertainty: 0.100)
Data arrival time: 2.153
Slack: 0.217

----------------------------------------------------------------------------------------------------
```

È dunque evidente non solo che il percorso critico, durante le varie fasi del flusso di progetto è stato modificato, ma anche che, in seguito alla fase di *placing*, lo slack è diventato negativo e dunque il circuito non sarebbe stato funzionante a meno di una nuova ottimizzazione, appunto quella *in-place*, che ha riportato lo slack ad un valore tollerabile. È inoltre evidente che lo slack è molto più elevato di quello trovato in fase di sintesi, e ciò è dovuto al fatto che il software di sintesi operava un'approssimazione dei valori parassiti dei collegamenti attraverso una stima degli stessi (*Wire Load Model*), stima assente invece nelle elaborazioni del software di P&R.
Si procede dunque con la realizzazione dell'albero di clock, necessario per garantire un clock-skew basso. Utilizzando un apposito script, *Encounter* realizza l'albero di clock salvando i risultati ottenuti in un report:

**Clock Skew Summary**

| Clock Name | Nr. Sink | Nr. Buffer | Rise Phase Delay | Fall Phase Delay | Trig. Edge Skew | Rise Skew | Fall Skew | Max. Rise Buffer Tran | Max. Fall Buffer Tran | Max. Rise Sink Tran | Max. Fall Sink Tran | Max. Local Skew | Min. Rise Buffer Tran | Min. Fall Buffer Tran | Min. Rise Sink Tran | Min. Fall Sink Tran | Detail Report |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| clk | 32 | 5 | 167.8(ps) ~ 173.9(ps) | 162.8(ps) ~ 166.3(ps) | 6.1(ps) | 6.1(ps) | 3.5(ps) | 50(ps) | 50(ps) | 49.9(ps) | 25.7(ps) | 44(ps) | 24.6(ps) | 44.2(ps) | 23.3(ps) | - | report |

**Figura 4.4 – Riassunto del Clock Skew**

Lo skew ottenuto è limitatissimo (ordine dei picosecondi) giacché i registri da pilotare sono pochi e vicini tra loro. È possibile individuare i registri facenti

parte l'albero di clock sul circuito integrato utilizzando le apposite funzionalità di *Encounter* (figura 4.5).

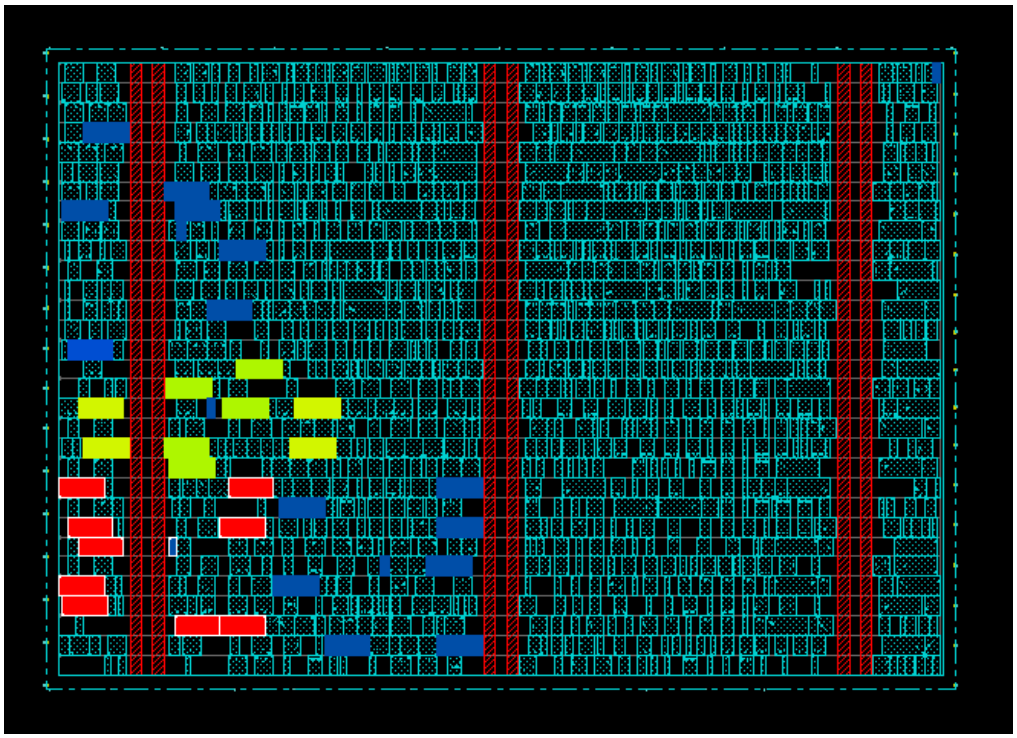

**Figura 4.5 – Posizione *clock tree***

Resta dunque da compiere la fase di *Routing*. In essa sono realizzati i collegamenti delle celle logiche tra di loro e con le linee di alimentazione e massa. Il risultato del routing è riportato in figura 4.6; è possibile inoltre analizzare il report per trovare informazioni utili:

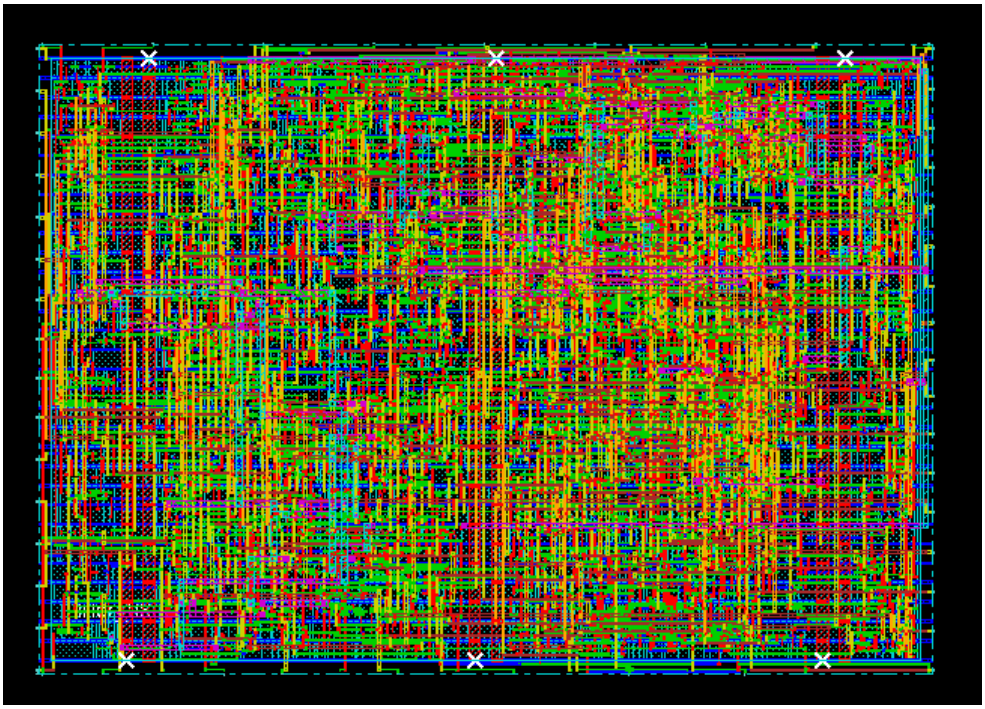

**Figura 4.6 – Circuito dopo la fase P&R**

```
#Complete Global Routing.
#Total wire length = 13277 um.
#Total half perimeter of net bounding box = 12067 um.
#Total wire length on LAYER metal1 = 145 um.
#Total wire length on LAYER metal2 = 3937 um.
#Total wire length on LAYER metal3 = 5320 um.
#Total wire length on LAYER metal4 = 2340 um.
#Total wire length on LAYER metal5 = 950 um.
#Total wire length on LAYER metal6 = 147 um.
#Total wire length on LAYER metal7 = 273 um.
#Total wire length on LAYER metal8 = 155 um.
#Total wire length on LAYER metal9 = 13 um.
#Total wire length on LAYER metal10 = 0 um.
#Total number of vias = 6900
#Up-Via Summary (total 6900):
#
#-----------------------
#  Metal 1         3267
#  Metal 2         2511
#  Metal 3          744
#  Metal 4          213
#  Metal 5           73
#  Metal 6           48
#  Metal 7           42
#  Metal 8            2
#-----------------------
#                  6900
```

Il livello di Metal più utilizzato è dunque il 3, che è lo stesso livello di Metal scelto in fase di progettazione dei pin di I/O per i collegamenti nord/sud del circuito. È possibile poi eseguire delle verifiche che garantiscano che i collegamenti siano stati portati a termine correttamente e che le regole di progetto e composizione non siano state violate.
Dopo aver eseguito l'ultimo script automatizzato, è possibile utilizzare *encounter* per ottenere informazioni riguardanti la potenza dissipata dal circuito. Inoltre, è possibile visualizzare graficamente la posizione delle celle che dissipano la maggiore potenza (figura 4.7).

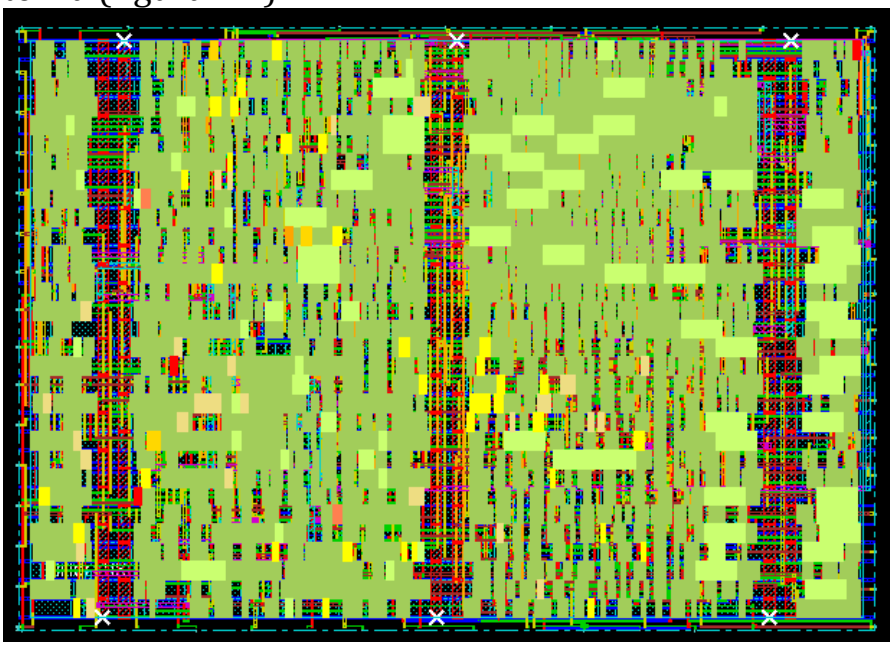

**Figura 4.7 – Celle che dissipano maggiore potenza**

E dunque possibile visualizzare il variare dei parametri fondamentali del circuito quale area occupata, potenza dissipata e periodo minimo di funzionamento del circuito, al variare della percentuale di occupazione delle celle logiche sul die:

| % Occupazione | Area [$\mu m^2$] | Potenza Dissipata [mW] | | Slack [ns] | Minimo Periodo di clock [ns] |
|---|---|---|---|---|---|
| | | Totale | Clock tree | | |
| 60 | 2,925E3 | 2,4873 | 1,0295E-1 | 0,081 | 2,419 |
| 65 | 2,71E3 | 2,772 | 1,019E-1 | 0,084 | 2,416 |
| 70 | 2,522E3 | 2,544 | 9,699E-2 | 0,076 | 2,424 |
| 75 | 2,36E3 | 2,601 | 1,006E-1 | 0,078 | 2,422 |
| 90 | 1,98E3 | - | - | **-0,385** | - |

Com'è evidente, nel momento in cui si prova a inserire un valore di percentuale di area occupata eccessivamente elevato, il circuito ha slack negativo. Graficamente, *encounter* evidenzia le celle logiche in cui i vincoli temporali non sono rispettati, colorandole di bianco (figura 4.8).

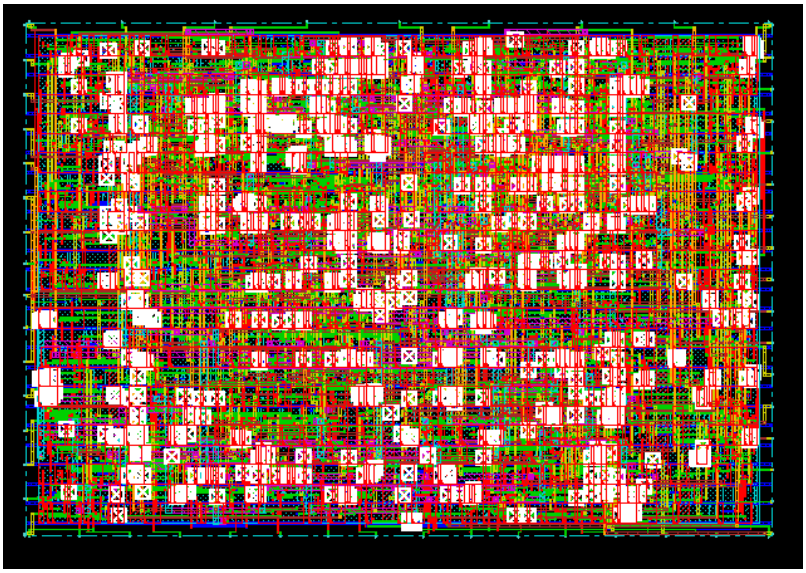

**Figura 4.8 – Circuito troppo denso**

È possibile, infine, prendere nota della posizione fisica del morsetto di clock nei registri per cui il ritardo sullo stesso assume il valore massimo ed il valore minimo, ciò sarà utile in seguito quando verrà effettuata la simulazione circuitale della distribuzione del clock. In particolare, i due registri, nel circuito sintetizzato, avranno coordinate (15,588;15,993) e (1,860;12,850).

## 5. Simulazione a Livello Transistor

03/05/2012 - Esercitazione 6

È necessario dunque eseguire una simulazione più accurata del circuito sintetizzato, utilizzando dei modelli accurati per i transistor. In particolare, per le simulazioni sarà utilizzato il modello BSIM versione 4, che garantisce un modello accurato per i MOSFET con dimensioni di canale sub-micrometriche.
Attraverso una lettura dei file di libreria forniti, è possibile analizzare i parametri dei MOSFET che saranno utilizzati e ipotizzare, secondo il modello semplificato utilizzato per le analisi carta e penna, quale sarà il valore della corrente di saturazione di tali dispositivi. In particolare, utilizzando il modello semplificato di MOSFET, attraverso l'equazione $I_{D_{SAT}} = K(V_{GS} - V_t)^2$, è possibile calcolare il valore atteso di corrente di pinch-off dei dispositivi, quando questi sono sottoposti a tensioni $V_{GS} = V_{DS} = 1V$ ed hanno una gate larga 90nm e lunga 45 nm. Attraverso il simulatore SPICE *spectre*, è possibile dunque eseguire un'analisi delle caratteristiche DC dei MOSFET, eseguendo per esempio uno sweep in continua sia sulla tensione di Gate sia su quella di Drain. Il risultato è poi visualizzato in maniera grafica attraverso il software *wavescan* (figura 5.1).

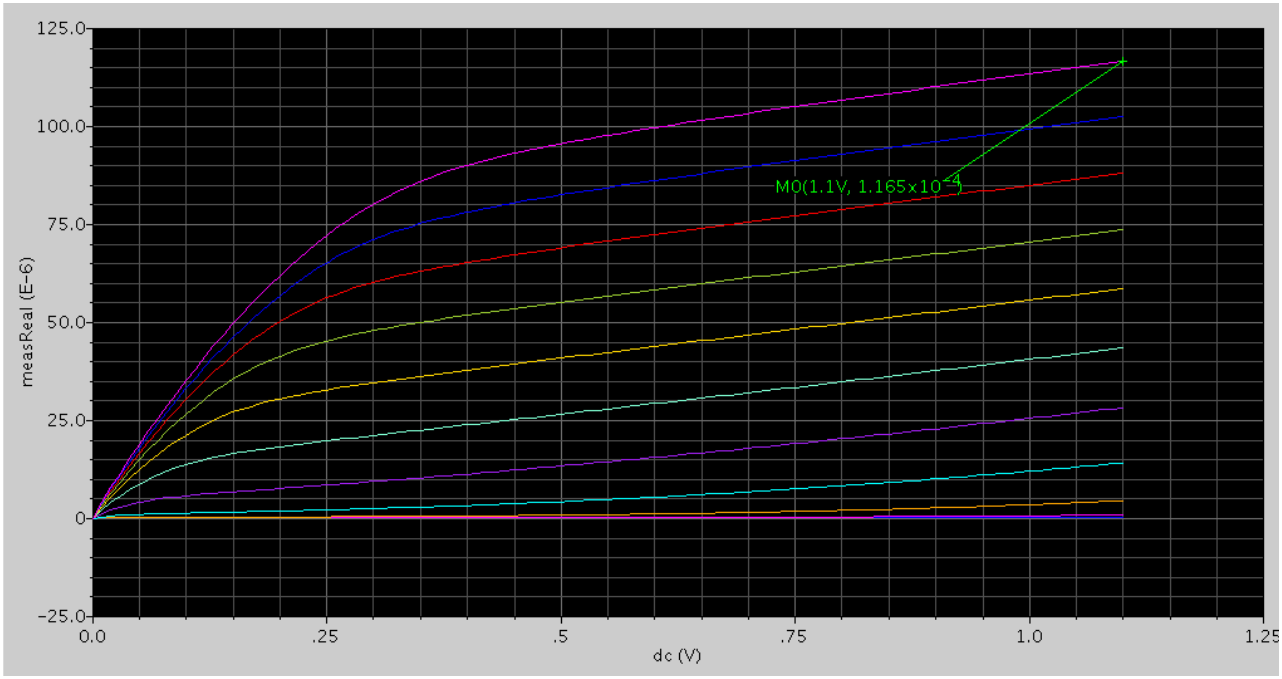


**Figura 5.1 – Caratteristiche MOSFET utilizzati**

Non solo è evidente che le caratteristiche non sono quelle "standard" dei MOSFET a canale lungo, ma anche che il valore di corrente di saturazione previsto con il modello abituale è differente dal valore ottenuto mediante simulazione SPICE.

Per valutare dunque la differenza, quantomeno qualitativa, tra le caratteristiche dei MOSFET a canale lungo e a canale corto, è possibile simulare un MOSFET che abbia lo stesso rapporto d'aspetto del primo, ma con canale largo 1µm e lungo 500nm (figura 5.2).

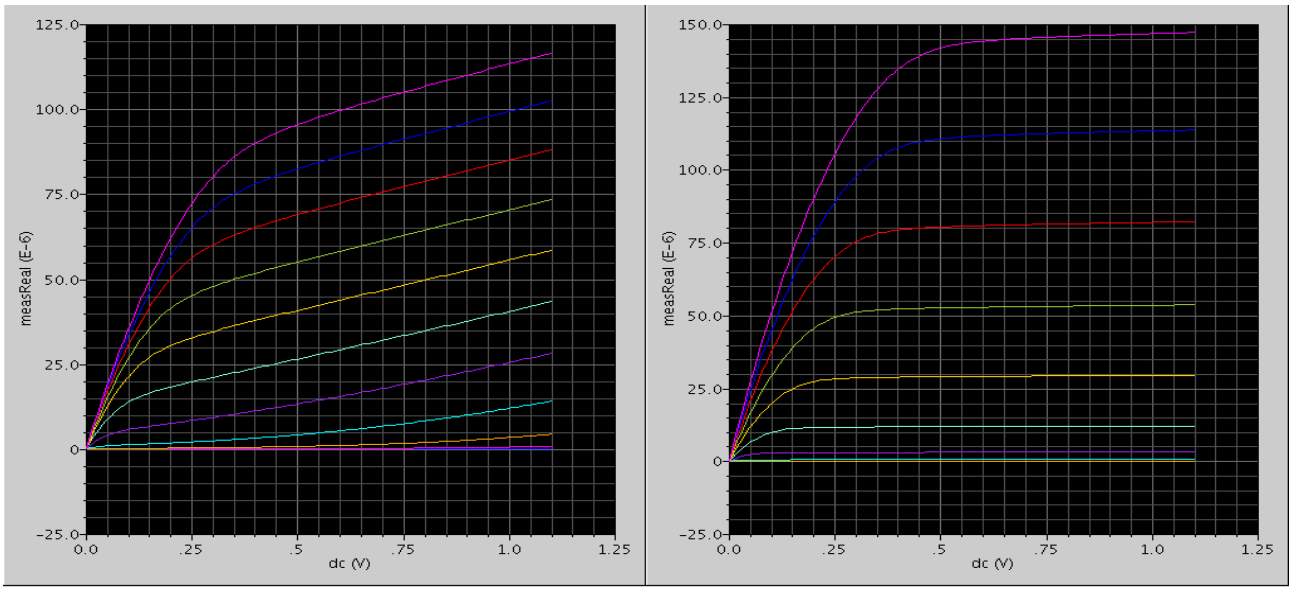


**Figura 5.2 – Confronto MOS a canale corto e a canale lungo**

Anche utilizzando delle stime più accurate per i parametri del modello a canale corto, l'analisi che si basa sul modello semplificato del MOSFET continua ad essere non accuratissima, infatti per una coppia di tensioni $(V_{GS}, V_{DS}) = (1V, 1V)$ il modello semplificato da come risultato 135µA mentre la simulazione 99,2µA, mentre per una coppia di tensioni $(V_{GS}, V_{DS}) = (1V, 0{,}5V)$ il modello semplificato da come risultato 113µA a fronte degli 82,5µA ottenuti in simulazione.
È inoltre possibile, fissata $V_{DS}$, analizzare l'andamento della corrente al variare della tensione $V_{GS}$ (figura 5.3).

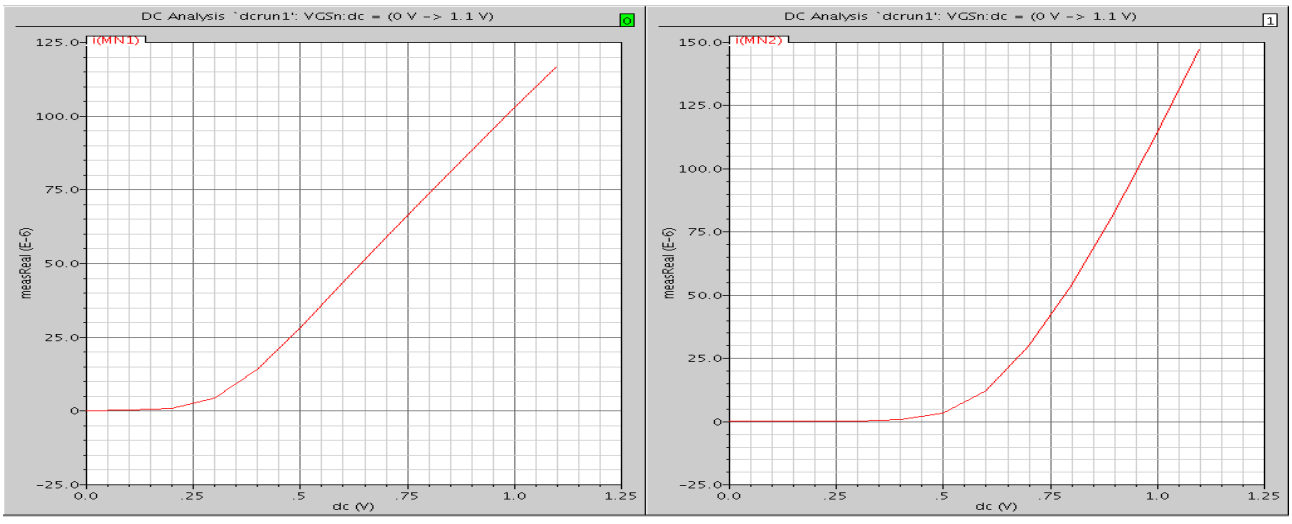


**Figura 5.3 – Transcaratteristica MOSFET**

È evidente che, mentre nel dispositivo a canale lungo la transcaratteristica è pressappoco quadratica, nel dispositivo a canale lungo questa è quasi completamente lineare, a conferma del fatto che il dispositivo non può essere modellato con le equazioni cui siamo abituati.
La corrente di saturazione degli NMOS risulta dunque essere (11,65/4,825)=2,42 volte più grande della corrente di saturazione dei PMOS a parità di dimensioni e di polarizzazione.
Per valutare invece la corrente di leakage, ovvero la corrente che scorre tra source e drain anche quando la tensione sulla gate è nulla, è opportuno eseguire un'ulteriore simulazione eseguendo uno sweep in tensione tra drain e source e collegando la gate a massa. Si esegue la simulazione su un dispositivo con lunghezza di gate di 45nm e larghezza 90nm. Il risultato è riportato in figura 5.4.

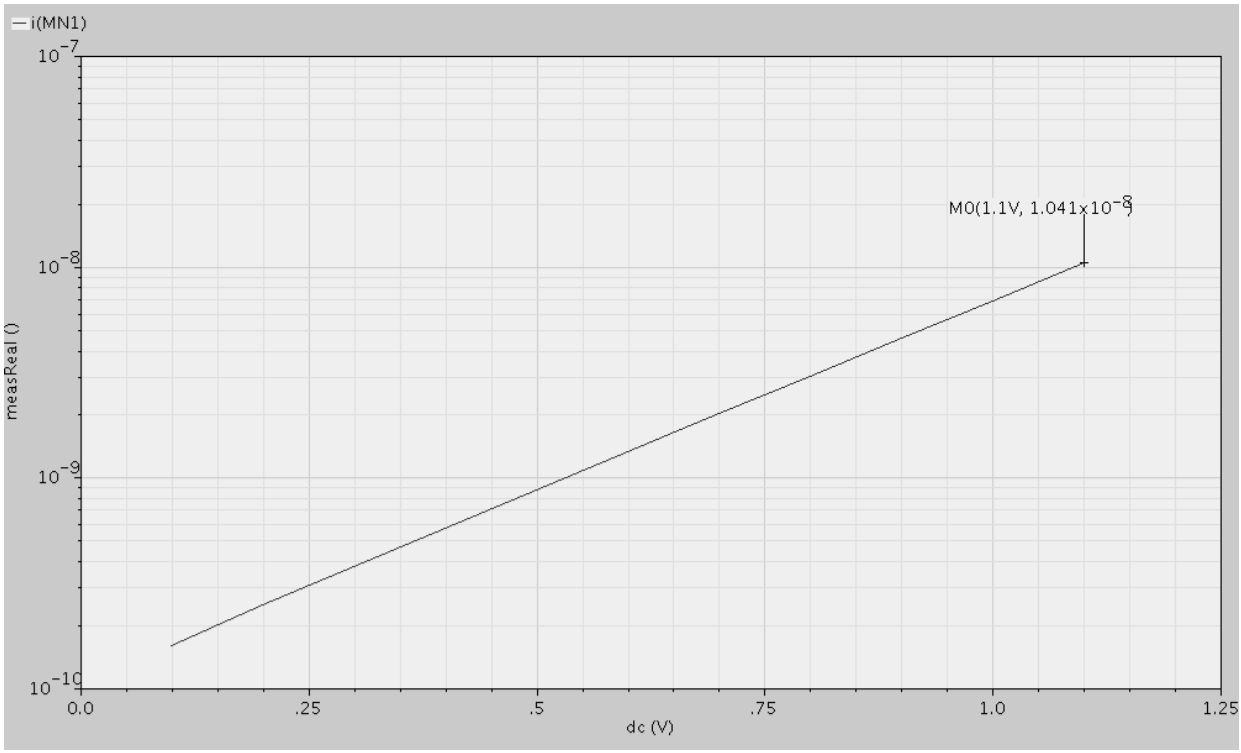


**Figura 5.4 – Corrente di leakage NMOS**

L'andamento è quello atteso poiché la corrente di leakage dei dispositivi NMOS è di tipo diffusivo e dunque ha un andamento esponenziale, dunque lineare se graficato in scala semilogaritmica. Eseguendo la stessa simulazione, ma con un dispositivo la cui lunghezza di canale sia raddoppiata (ovvero 90nm), si può verificare che effetti di canale corto, come proprio la corrente di leakage, vengono drasticamente ridotti. La corrente di leakage si riduce, infatti, di due ordini di grandezza, passando da 1,041E-8 A a 5,532E-10 A. È inoltre possibile valutare l'elevata dipendenza della corrente di leakage dalla temperatura, effettuando tre diverse simulazioni a temperature differenti (figura 5.5).

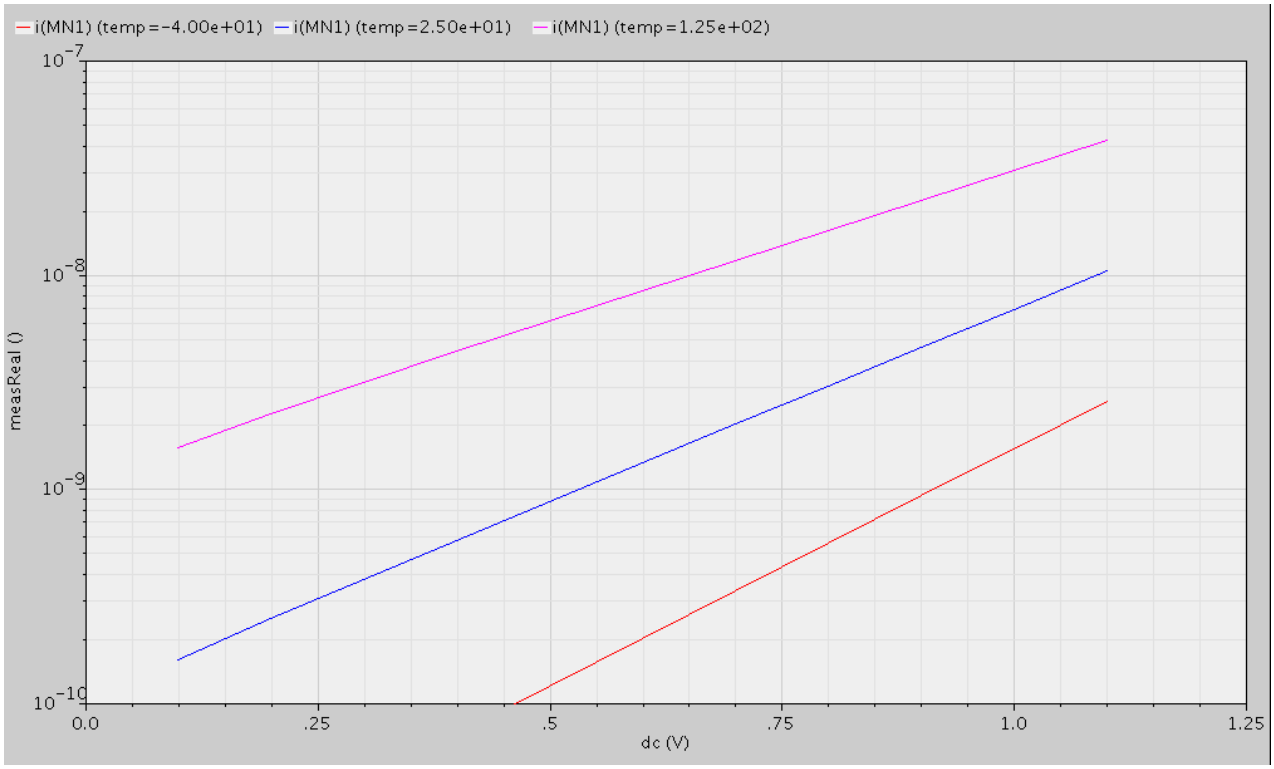


**Figura 5.5 – Variazione della corrente di leakage al variare della temperatura**

È possibile inoltre compiere la simulazione utilizzando, anziché i dispositivi le cui caratteristiche sono descritte nel file .model relativo al caso *normal*, quelli del caso *fast*. In tale caso, la corrente di leakage sarà leggermente più grande poiché si suppongono frequenze operative maggiori. La corrente di leakage, per il caso *fast*, sarà infatti di 6,7E-8 A, a fronte degli 1,04E-8 A del caso *normal*.
Per l'analisi dinamica, si simula il circuito in figura 5.6.

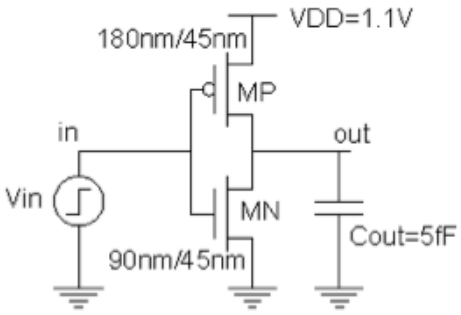


**Figura 5.6 – Circuito utilizzato per la simulazione dinamica.**

È possibile utilizzare wavescan per visualizzare gli andamenti dei segnali di ingresso/uscita, ed ottenere anche una sima dei tempi di propagazione del circuito (figura 5.7).

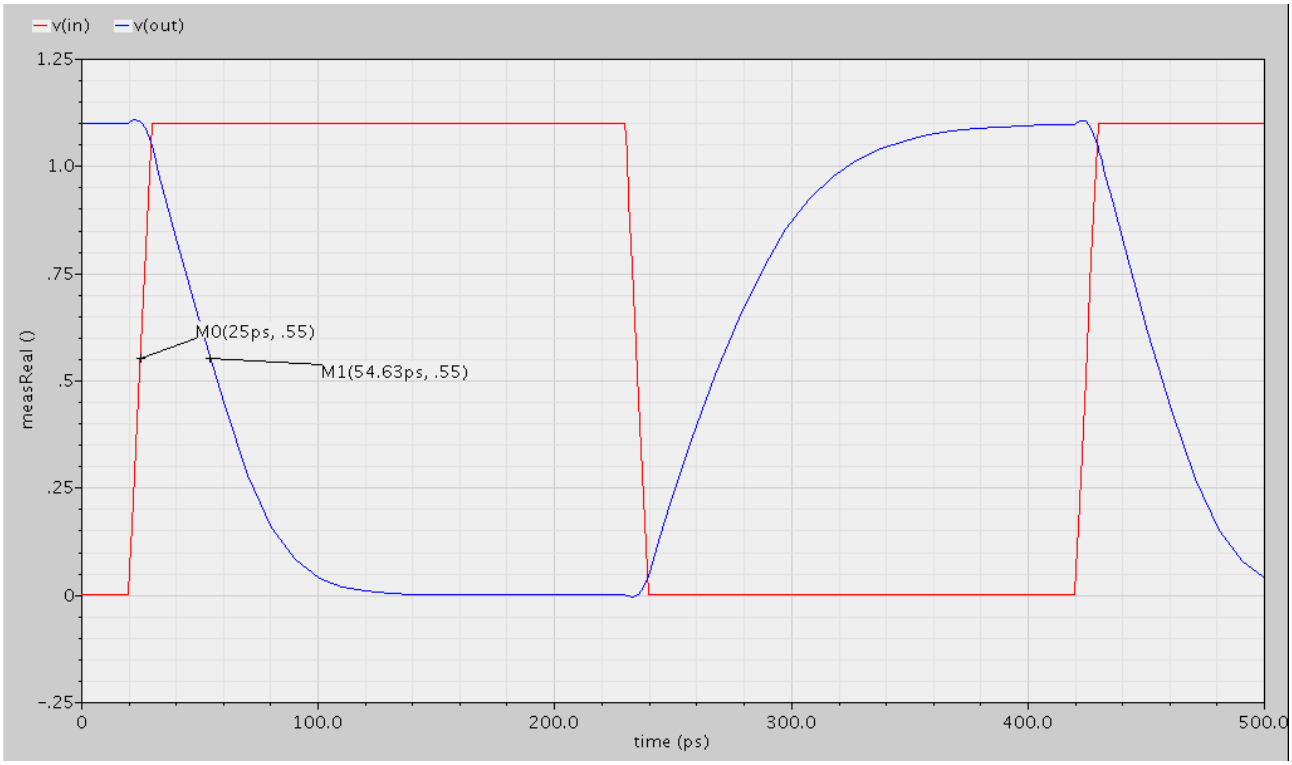


**Figura 5.7 – Segnali in ingresso e in uscita al circuito analizzato.**

Per ottenere una più accurata misura dei tempi di propagazione, è però possibile utilizzare le funzionalità di SPICE attraverso il comando *.measure*. In questo modo, viene calcolato precisamente il valore dei tempi di propagazione; per la precisione si avrà $t_{pHL} = 29{,}53\text{ps}$ e $t_{pLH} = 35{,}76\text{ps}$, risultati in linea con quanto ottenuto dal confronto grafico. Attraverso questi valori, è possibile calcolare la resistenza equivalente dei MOSFET utilizzando il modello semplificato switch-level. Si avrà, infatti, $t_p = 0{,}69 R_{eq} C_L$, sostituendo i valori dei tempi di propagazione calcolati, si trovano come resistenze equivalenti:

$$\begin{cases} R_{Neq} = \dfrac{t_{pHL}}{0{,}69 C_L} = 8{,}559 k\Omega \\ R_{Peq} = \dfrac{t_{pLH}}{0{,}69 C_L} = 10{,}365 k\Omega \end{cases}$$

È possibile valutare la variazione dei tempi di propagazione al variare di parametri quali la capacità di carico o il tempo di salita/discesa dei segnali, attraverso un'analisi SPCIE parametrica. Gli andamenti sono riportati in figura 5.8 e 5.9

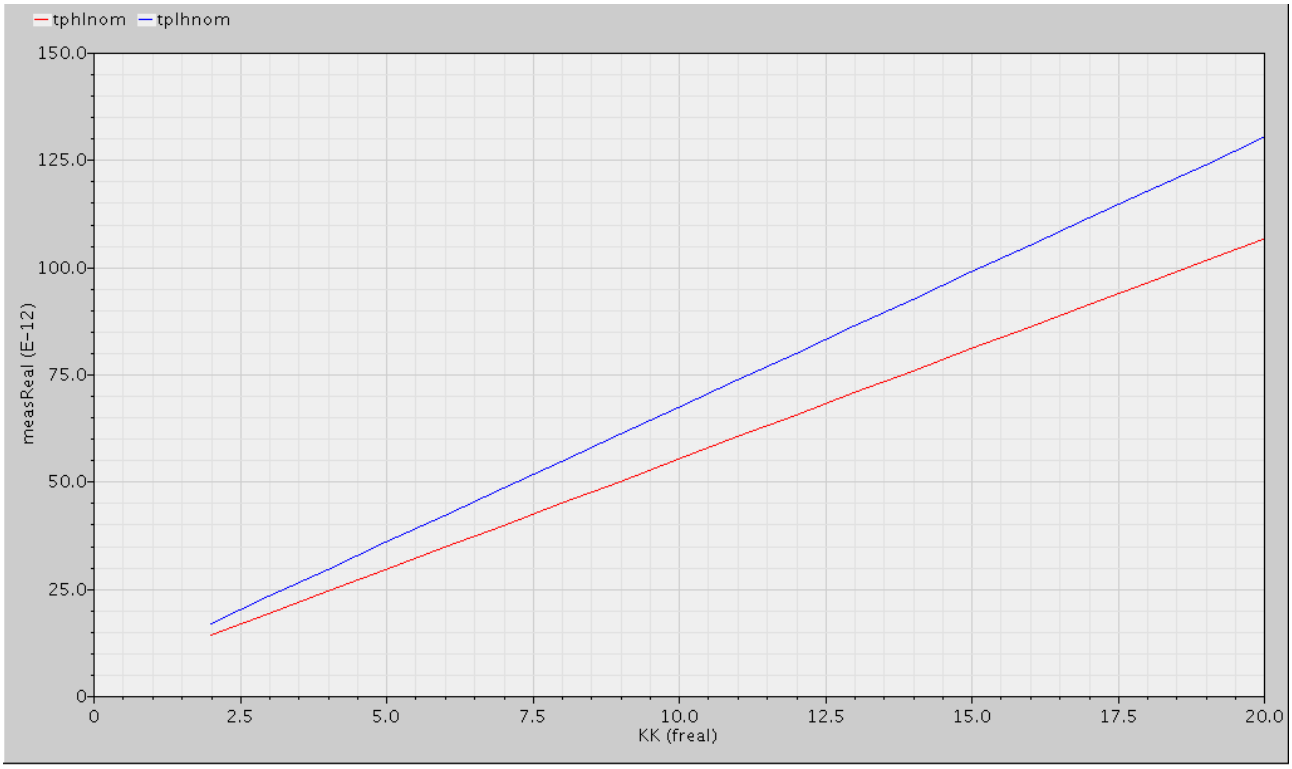


Figura 5.8 – Variazione dei tempi di propagazione al variare della capacità in uscita.

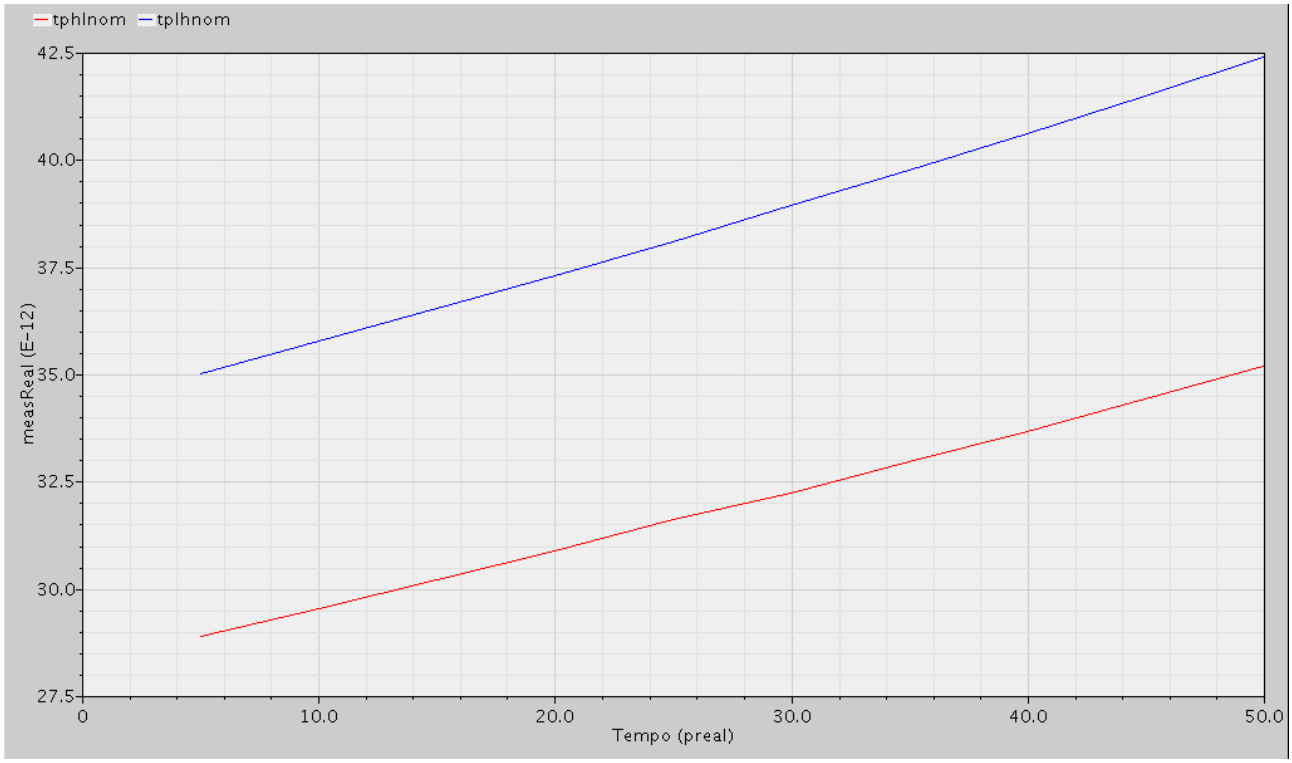


Figura 5.9 – Variazione dei tempi di propagazione al variare dei tempi di salita/discesa dei segnali in ingresso.

In particolare, in riferimento a figura 5.8, l'andamento lineare dei tempi di propagazione risulta essere coerente con la formulazione analitica introdotta precedentemente, ovvero che $t_p = 0{,}69 R_{eq} C_L$, e dunque lineare.

Si passa dunque ad analizzare un circuito più complesso, ovvero quello rappresentato in figura 5.10, e si concentra l'attenzione sul secondo invertitore, ovvero quello che pilota gli altri 4 invertitori. Tale configurazione è stata definita attraverso una netlist SPICE istanziando gerarchicamente il componente di libreria INV_X1.

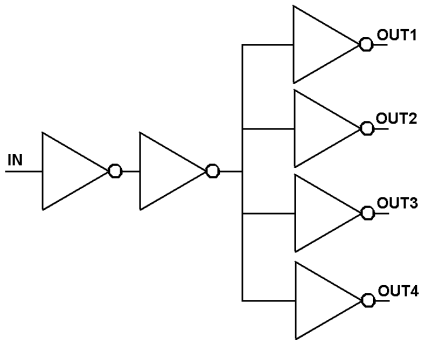


**Figura 5.10 – Circuito preso in esame**

Attraverso il comando *.measure*, è possibile valutare i tempi di propagazione per il secondo invertitore. Si avrà dunque $t_{pHL} = 15\text{ps}$ e $t_{pLH} = 23{,}5\text{ps}$.
Se si volessero valutare analiticamente i valori dei tempi di propagazione, sarebbe necessaria una stima della capacità pilotata dall'invertitore e della resistenza equivalente del modello switch-level.
Riguardo la prima, si può assumere che la capacità di pilotaggio sia data dalla sola capacità di gate dei MOSFET pilotati. In particolare, la capacità sarà data dalla capacità di gate di un singolo invertitore, moltiplicata per 4.
Leggendo il file *INV_X1.pex.netlist* si evince che, nel componente di libreria INV_X1, è presente un NMOS con L=50nm W=90n ed un PMOS con L=50nm e W=135nm.

$$C_{OX} = \frac{\varepsilon_{OX}}{t_{OX}} = \frac{34 \cdot 10^{-12}\, F/m}{1{,}75 \cdot 10^{-9} m} = 19{,}43 \cdot 10^{-3}\, \frac{F}{m^2}$$

Vado allora a calcolare la capacità di carico equivalente:

$$C_L = 4 \cdot C_G = 4 \cdot (A_N + A_P) \cdot C_{OX} = 4C_{OX} \cdot (W_P L_P + W_N L_N)$$
$$C_L = 4 \cdot 19{,}43 \cdot (6750 + 4500) \cdot 10^{-21} F = 0{,}8744 fF$$

Per quanto riguarda invece la resistenza, attraverso opportune manipolazioni algebriche, è possibile ottenere la seguente espressione, valida nell'approssimazione switch-level:

$$R_N = \frac{V_{DD}}{1{,}4 k_N \left[2(V_{DD} - V_{SAT})V_{SAT} - {V_{SAT}}^2\right]\left(1 + \lambda \frac{3}{4} V_{DD}\right)} = 7{,}77 k\Omega$$

Il tempo di propagazione valutato analiticamente, con le approssimazioni effettuate, è $t_{pHL} = 4{,}7\text{ps}$, un valore decisamente piccolo, il che porta a considerare che tutte le approssimazioni utilizzate in questa analisi sono di fatto troppo restrittive e non tengono conto di termini di ordine superiore che sono invece considerati nelle elaborazioni del simulatore SPICE.
Si passa allora alla valutazione dei tempi di hold e di setup dei registri. All'uopo, si utilizza un segnale di dato il quale è fatto variare opportunamente prima o dopo il segnale di clock. Per la valutazione del tempo di setup, si fa variare il dato da 100ps prima del fronte del clock fino a commutare in contemporanea con il clock stesso. Se, attraverso una netlist SPICE, si valuta l'andamento dell'uscita, e in particolare il ritardo *clock to Q* Tcq, è chiaro che l'ultima commutazione del dato in uscita corrisponde al segnale in ingresso che è commutato nel limite del tempo di setup, mentre non vi è commutazione dopo che il segnale è commutato. Utilizzando i comandi SPICE di misura, si può trovare che il tempo di setup risulta essere 26ps. Allo stesso modo, per valutare il tempo di hold, si fa commutare un segnale dai 50ps prima ai 50ps dopo il fonte di clock. La prima uscita osservata sarà quella corrispondente al tempo di hold del flip-flop. In particolare, il componente che si sta caratterizzando risulta avere un tempo di hold negativo: anche se il dato commuta prima del fronte del clock, l'uscita tende lo stesso a portarsi al valore logico alto (figura 5.11); ciò è probabilmente dovuto al fatto che il segnale dato D in ingresso al flip-flop è opportunamente bufferizzato e quindi ritardato. Il tempo di hold è dunque negativo e risulta essere -18,5ps.

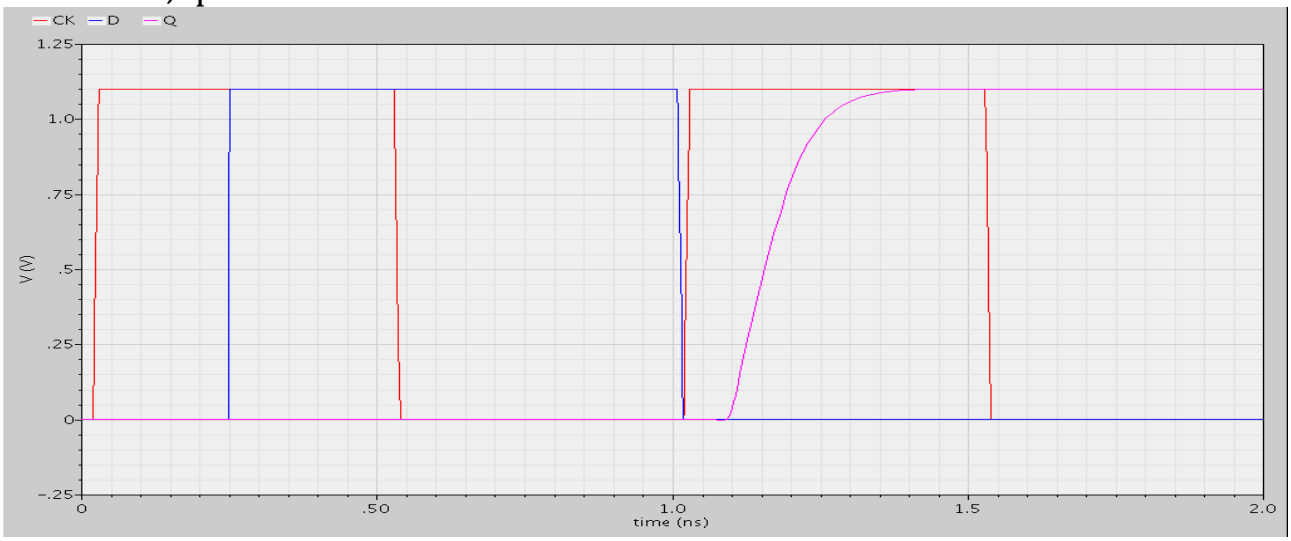


**Figura 5.11 – Valutazione tempo di hold del flip-flop.**

È dunque interessante valutare l'andamento del tempo *clock to q* al variare dei segnali in ingresso, si realizza dunque un grafico in cui si riporta sull'asse delle ordinate il Tcq mentre sull'asse delle ascisse è riportato l'istante di commutazione del segnale D, in cui si assume come istante zero l'istante in cui commuta il clock. Si può dunque notare la presenza di una banda centrale in cui,

il dato è commutato violando i vincoli di setup e di hold, e dunque l'uscita non commuta, ma resta nello stato logico precedente al clock (figura 5.12).

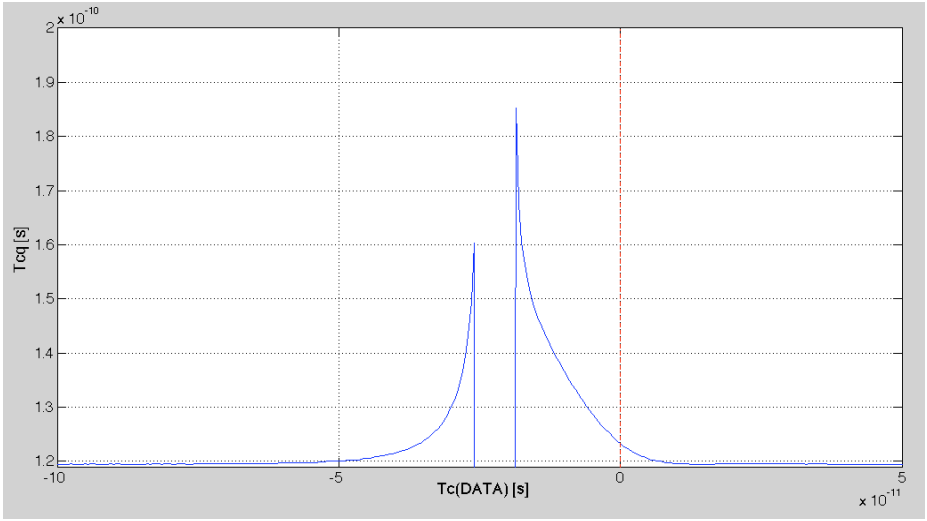


**Figura 5.12 – Variazione uscita in funzione dell'istante di commutazione del segnale D in ingresso.**

Per una simulazione più accurata dei dispositivi, è possibile utilizzare, al posto del software *spectre*, il software *ultrasim*, che consente di avere delle simulazioni con diversi livelli di accuratezza, a discapito del tempo di simulazione. Per valutare ciò, si è simulato il circuito di figura 5.11 con questo nuovo software, riportando i diversi tempi di propagazione misurati con i diversi livelli di accuratezza.

| Simulatore | Tphl [ps] | Tplh[ps] |
|---|---|---|
| spectre | 15,00 | 23,54 |
| Ultrasim – spice | 15,38 | 23,70 |
| Ultrasim – mixed-signal | 14,53 | 23,11 |
| Ultrasim – digital accurate | 14,37 | 23,55 |
| Ultrasim – digital fast | 14,54 | 21,64 |

Il simulatore, inoltre, consente di visualizzare attraverso *ultrasim* le forme d'onda, sia in formato analogico sia digitale. È interessante notare che l'intervallo di tensioni analogiche comprese tra 0.5V e 0.6V è indicato come "indefinito" nel grafico di tensioni digitali (figura 5.13).

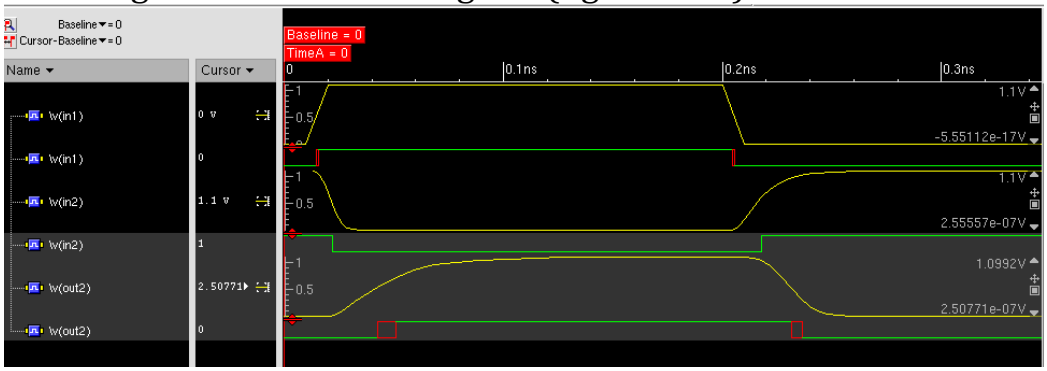


**Figura 5.13 – Segnali particolari in formato analogico e digitale**

## 6. Dal livello layout alla simulazione transistor

31/05/2012 - Esercitazione 7

Per eseguire un'analisi del circuito realizzato, è necessario controllare che le regole di progetto e di composizione siano rispettate, ed eseguire dunque un'analisi che sia la più precisa possibile, considerando anche le capacità e le resistenze parassite. Per prendere mano con il software da utilizzare, *Cadence Design System*, si analizza un circuito più semplice, costituito da una coppia di buffer (figura 6.1).

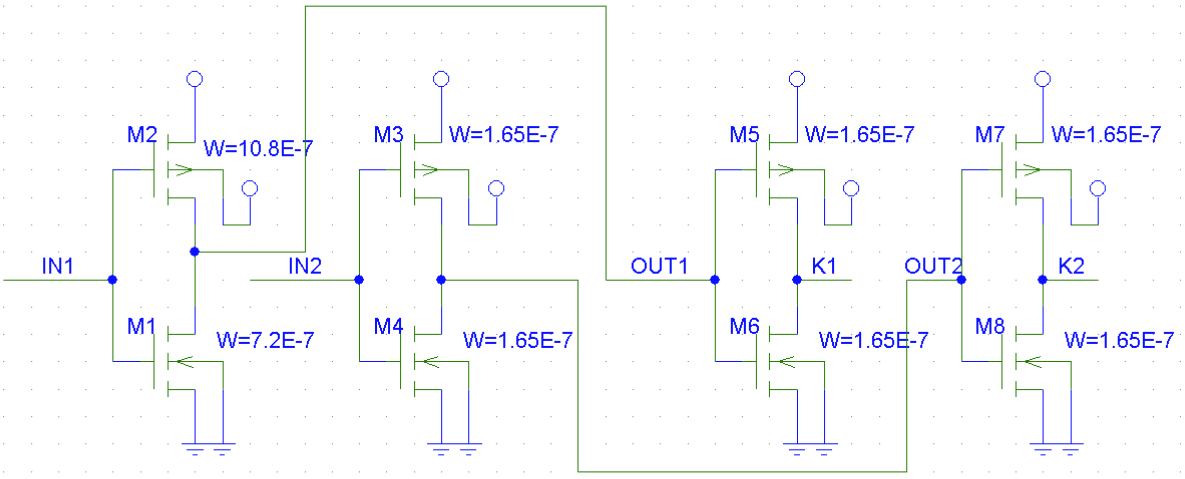


**Figura 6.1 – Circuito preso in analisi**

Visualizzando il layout del circuito, si nota subito che i due dispositivi hanno due linee di metal poste in vicinanza, che daranno dunque vita a un elevato accoppiamento capacitivo. Eseguendo un'analisi RVE del circuito è la pista di Metal1 che collega l'uscita del primo invertitore all'ingresso del secondo l'elemento parassita più resistivo. Il software calcola automaticamente il valore di tale resistenza, che è molto prossimo al valore atteso considerando che il Metal1 ha una resistenza di strato di 0,38Ω e misurando i valori di larghezza e lunghezza con l'apposito *tool* del *Layout Viewer*. Come auspicabile, un'analisi RVE evidenzia anche che il maggior contributo capacitivo è dato dall'accoppiamento delle line di Metal1 *out1* e *out2* (figura 6.2).

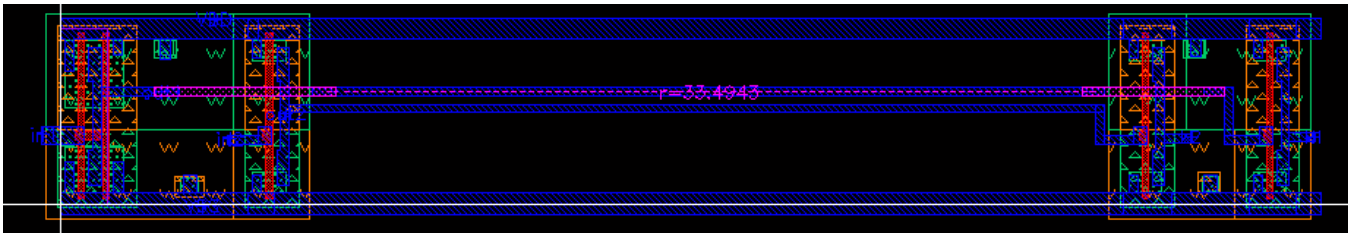

**Figura 6.2 – Circuito in analisi – livello layout.**

La presenza di una capacità parassita così elevata, comporta dei malfunzionamenti del circuito stesso, saranno, infatti, presenti fenomeni di *cross talking*. Infatti, commutando il valore logico del primo buffer, anche lasciando al valore logico basso l'ingresso del secondo buffer, i segnali *OUT2* e *K2* subiranno delle lievi variazioni (figura 6.3), mentre l'effettuare una commutazione su *IN2* lasciando invariato il valore logico di *IN1* comporta una variazione più bassa delle tensioni *OUT1* e *K1*, in virtù del fatto che la *drive strenght* del primo buffer è maggiore di quella del secondo (la W equivalente del primo invertitore è più grande di tutte le altre W equivalenti degli altri invertitori del circuito) (figura 6.4).

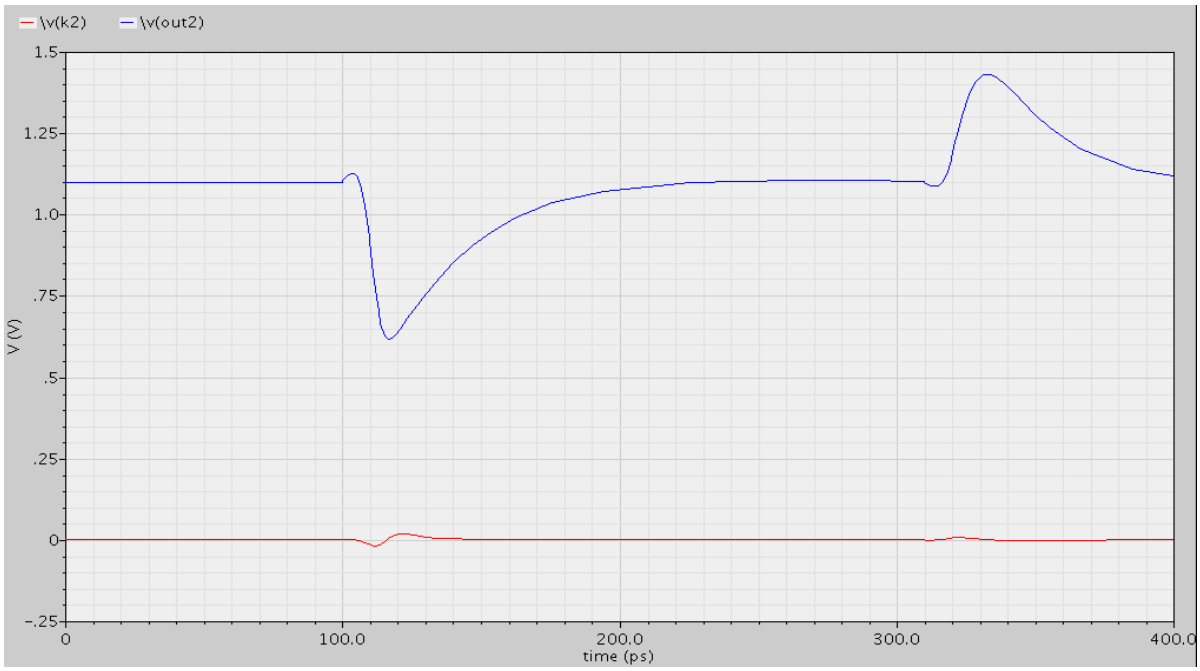


**Figura 6.3 – Commutazione di *IN1* con *IN2* costante**

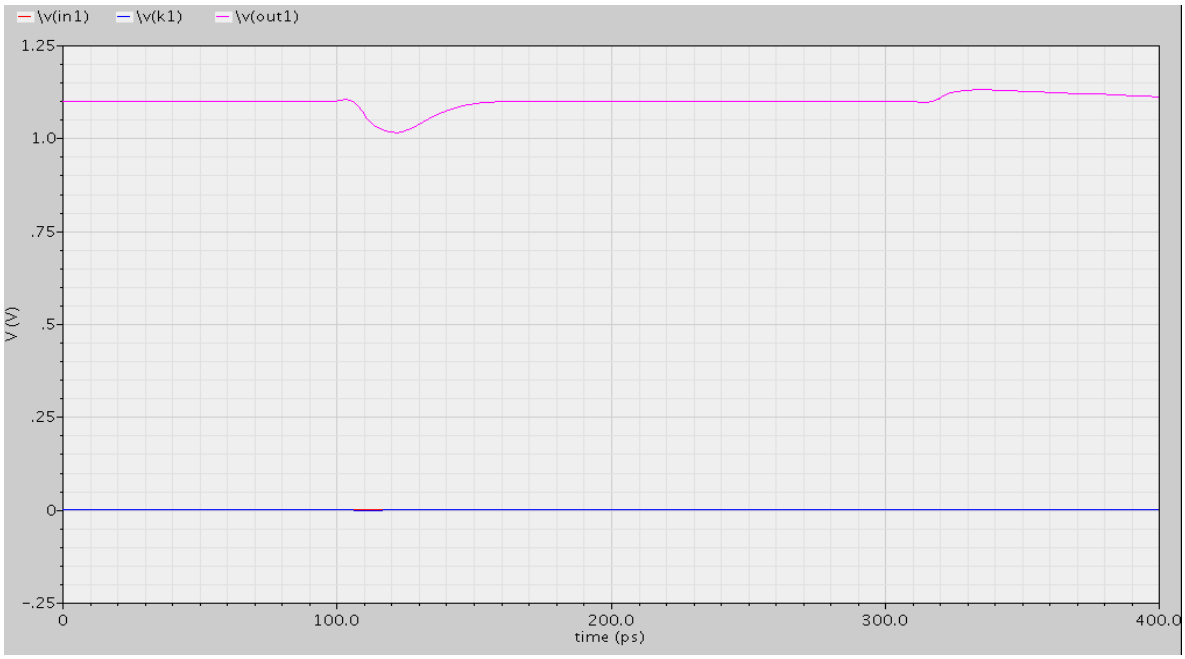


**Figura 6.4 – Commutazione di *IN2* con *IN1* costante**

Attraverso il *Cadence Library Manager* si procede dunque con la creazione di una nuova libreria che racchiuderà tutte le celle utilizzate gerarchicamente nel progetto del datapath di *pico*. Il circuito sintetizzato dalla descrizione VHDL rispetta le regole di progetto e composizione, come testimonia l'analisi DRC effettuata (figura 6.5).

```
--- RULECHECK RESULTS STATISTICS (BY CELL)
---
------------------------------------------------------------------------------
--- SUMMARY
---
TOTAL CPU Time:                 1
TOTAL REAL Time:                2
TOTAL Original Layer Geometries: 13023 (89689)
TOTAL DRC RuleChecks Executed:  156
TOTAL DRC Results Generated:    0 (0)
```

**Figura 6.5 – Analisi DRC del datapath di *pico*.**

Per eseguire una simulazione, si valuterà dunque lo skew, andando a prelevare, con apposite label, il segnale di clock dai punti che nell'analisi effettuata nella fase di P&R del circuito risultavano essere quelli critici, ovvero quelli a clock skew maggiore. In particolar modo, se si pongono tutti i segnali a massa e si analizza il solo andamento del clock, è possibile valutare in maniera precisa lo skew utilizzando gli appositi comandi *.meas* del simulatore SPICE (figura 6.6).

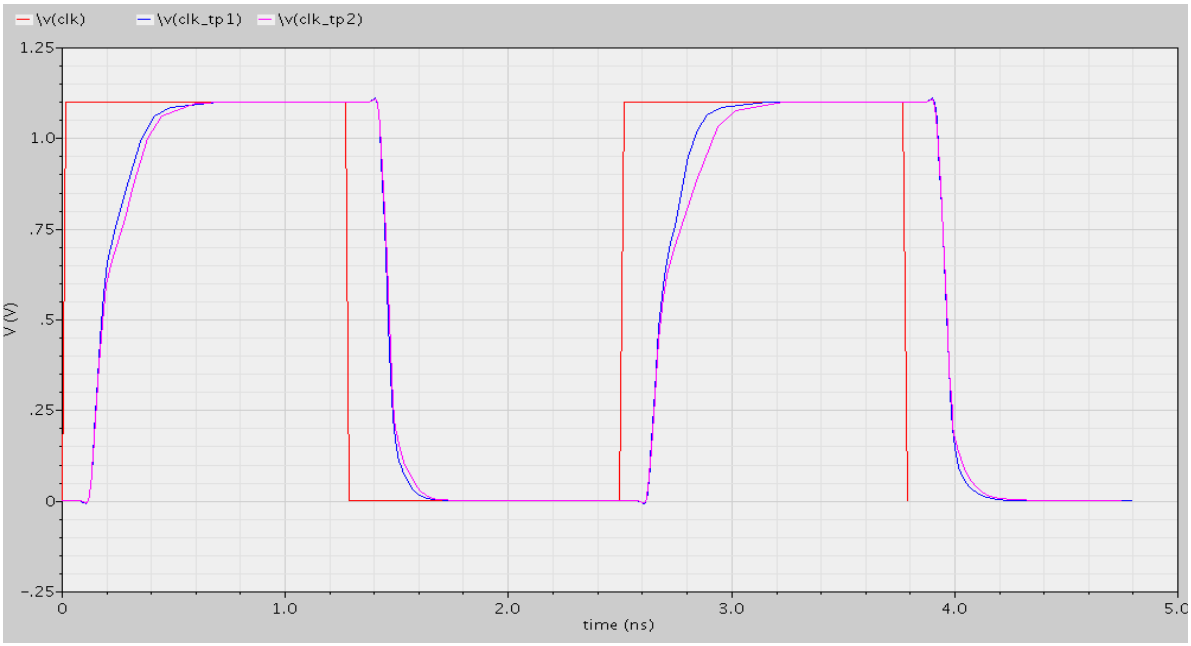


**Figura 6.6 – Segnale di clock nei punti critici del circuito.**

Il risultato della misura riporterà i valori di skew per entrambi i test point (tp_1 e tp_2), e per entrambi le fasi di rise e di fall del segnale di clock. I valori misurati sono i seguenti:

```
potenza          =   -1.5634e-04
delay1_rise      =    1.7584e-10
delay2_rise      =    1.8179e-10
delay1_fall      =    1.8234e-10
delay2_fall      =    1.8462e-10
```

I valori di skew risultano dunque essere 5,95ps per il rise e 2,38ps per il fall, valori non eccessivamente dissimili da quelli ottenuti in fase di P&R utilizzando il *wireload model*.

## 7. Completiamo PICO

07/06/2012 - Esercitazione 8

Resta dunque da completare la descrizione del microporcessore *pico16* in VHDL, descrivendo la parte di controllo dello stesso, simulando il circuito completo ed effettuando la fase di sintesi e di Place & Route. Il circuito completo è mostrato in figura 7.1.

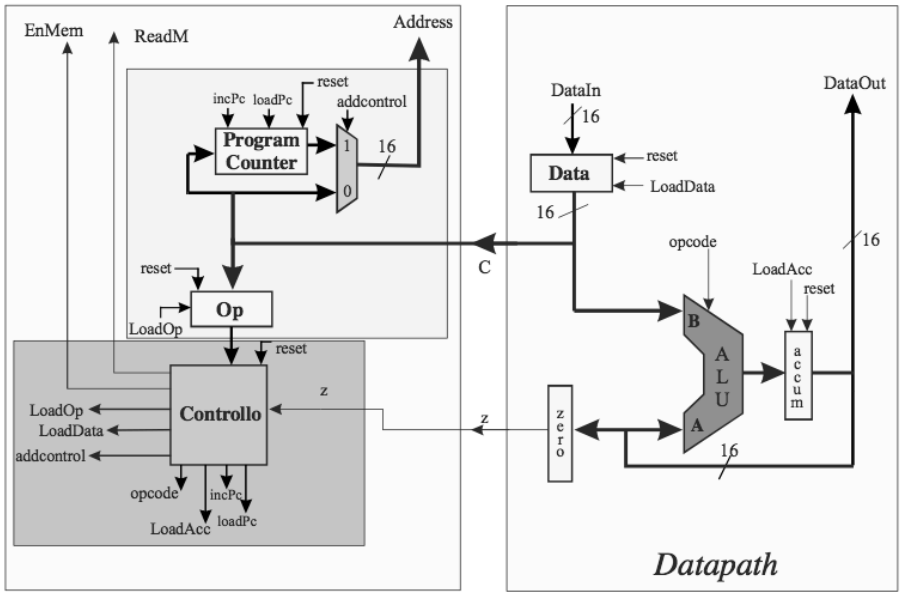


**Figura 7.1 – Schema a blocchi di PICO**

La parte fondamentale del blocco di controllo consiste in una macchina a stati finiti che esegue un ciclo di funzionamento infinito, mostrato in figura 7.2. Secondo il codice operativo immagazzinato nel registro OP, l'unità di controllo eseguirà le diverse funzionalità di pico, le cui operazioni sono codificate in 8 bit.

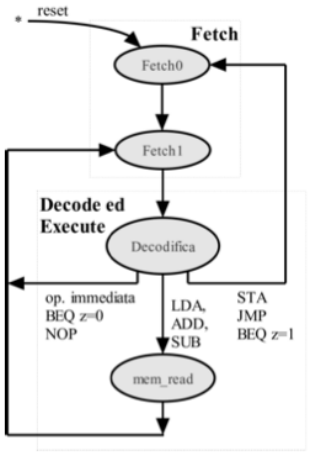


**Figura 7.2 – Funzionalità del controllore**

Per simulare il funzionamento di *pico*, è stata fornita una descrizione VHDL di una memoria RAM, che a sua volta richiama un file.txt che contiene i valori (dati in ingresso e istruzioni) che verranno a mano a mano caricati nel processore per valutarne il funzionamento.

Simulando il circuito, composto dunque dal microprocessore collegato alla memoria RAM, con un apposito test bench, il risultato sarà quello riportato in figura 7.3.

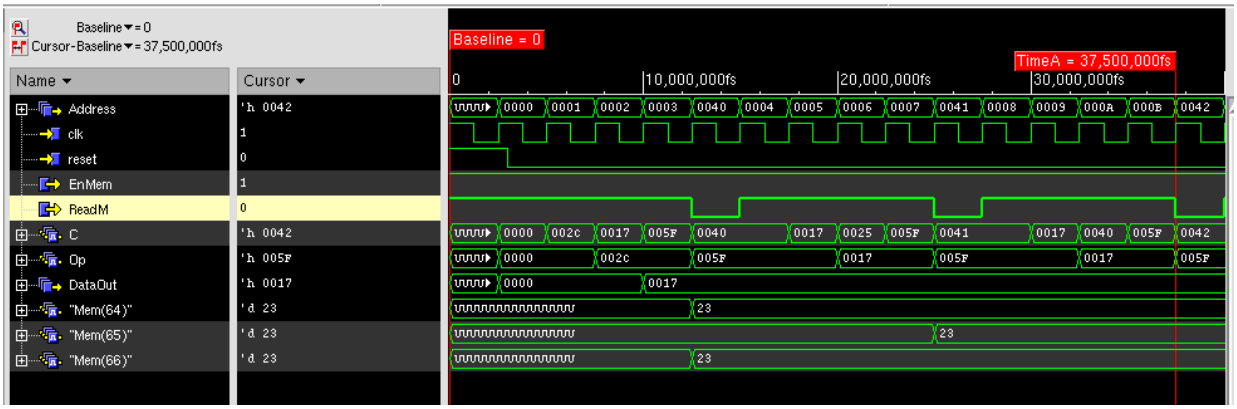


**Figura 7.3 – Test bench di *pico16*.**

Si procede dunque alla sintesi di *pico* utilizzando gli stessi scripts utilizzati nella fase di sintesi del *datapath*, attraverso il sintetizzatore *BGX Shell*.
Si utilizza allora *Encounter* per eseguire il P&R del circuito. In particolare, per le alimentazioni, a differenza di quanto fatto per il solo *datapath*, si preferisce utilizzare dei *ring* che circondano il circuito, per ottimizzare la distribuzione delle alimentazioni sui diversi punti del circuito.
Si procede poi con la fase di placing, eseguita dal software.

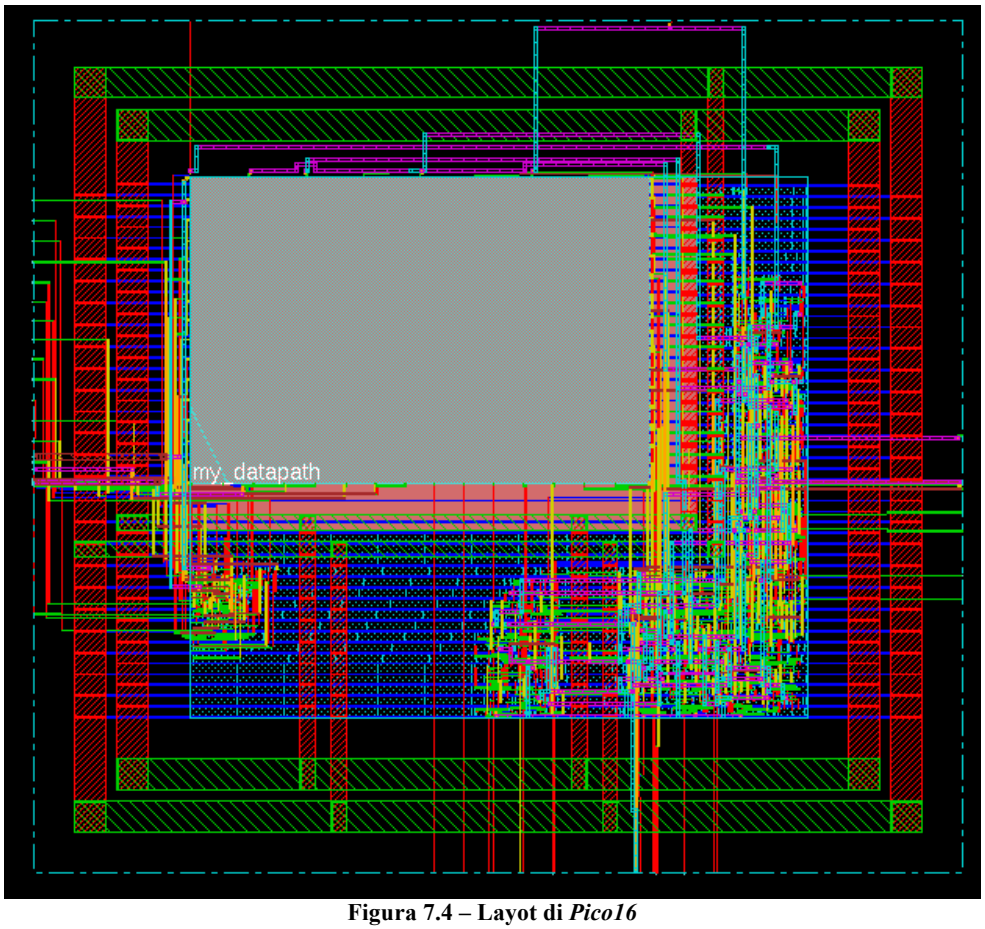


**Figura 7.4 – Layot di *Pico16***

Per ottimizzare la distribuzione del clock, invece, si utilizza lo stesso schema di albero di clock utilizzato in fase di placing del *datapath*, copiando il comando di creazione del *clock tree* generato automaticamente in precedenza.
A questo punto, va dunque eseguita l'ottimizzazione *in-place*, che riordina le celle logiche in modo da ottimizzare la distribuzione del clock e delle alimentazioni utilizzando buffer e celle di *feedthrough*.
Si esegue infine la fase di routing, seguita dalla fase di analisi, per verificare che le operazioni siano state tutte eseguite correttamente, e si salva dunque il risultato con un apposito script di output.
È possibile controllare i report per verificare che non ci siano errori di temporizzazione:

**Clock Skew Summary**

| Clock Name | Nr. Sink | Nr. Buffer | Rise Phase Delay | Fall Phase Delay | Trig. Edge Skew | Rise Skew | Fall Skew | Max. Rise Buffer Tran | Max. Fall Buffer Tran | Max. Rise Sink Tran | Max. Fall Sink Tran | Max. Local Skew | Min. Rise Buffer Tran | Min. Fall Buffer Tran | Min. Rise Sink Tran | Min. Fall Sink Tran | Detail Report |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| clk | 35 | 9 | 225.4(ps) ~ 236.1(ps) | 203.1(ps) ~ 228.6(ps) | 10.7(ps) | 10.7(ps) | 25.5(ps) | 51.1(ps) | 26.2(ps) | 48.8(ps) | 24.3(ps) | 6(ps) | 6(ps) | 38.7(ps) | 20.3(ps) | - | report |

**Figura 7.5 – Clock Skew Summary**

```
# generated on Thu Jun  7 16:04:18
# Top Cell: pico

------------------------------------------------------------
          timeDesign Summary
------------------------------------------------------------

+--------------------+---------+---------+---------+---------+---------+---------+
|     Setup mode     |   all   | reg2reg | in2reg  | reg2out | in2out  | clkgate |
+--------------------+---------+---------+---------+---------+---------+---------+
|           WNS (ns):| -0.000  | -0.000  |  1.315  |  0.310  |   N/A   |   N/A   |
|           TNS (ns):|  0.000  |  0.000  |  0.000  |  0.000  |   N/A   |   N/A   |
|    Violating Paths:|    0    |    0    |    0    |    0    |   N/A   |   N/A   |
|          All Paths:|   92    |   41    |   51    |   34    |   N/A   |   N/A   |
+--------------------+---------+---------+---------+---------+---------+---------+

Density: 129.426%
Real DRV (fanout, cap, tran): (0, 9, 0)
Total DRV (fanout, cap, tran): (0, 12, 0)
------------------------------------------------------------
```

La realizzazione del microprocessore *pico*16 può dunque ritenersi conclusa con successo.