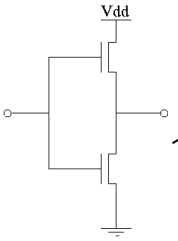
Flusso di progetto circuiti digitali

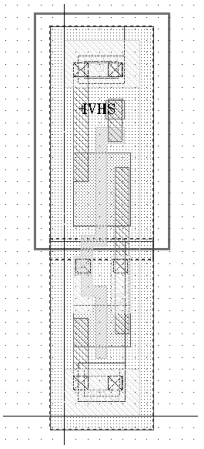
(anni 80/90)

1) Definizione Algoritmica : y=-x;

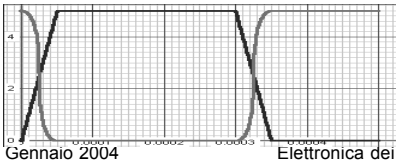


2) Schematic Entry

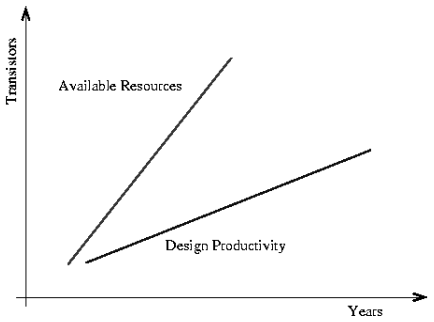
3) Custom Layout (place & route)



4) Parasitic extraction & Backannotation

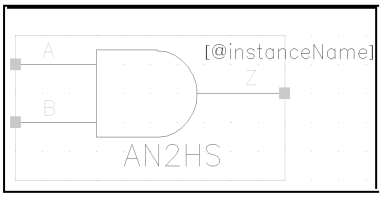


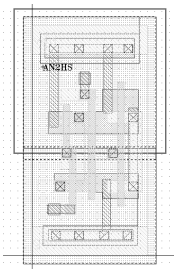
II Design Productivity Gap



Lo sviluppo della tecnologia offre una quantita' di risorse di Calcolo che supera la capacita' del progettista di utilizzarle.

Tecnologia Standard Cells





Per velocizzare il tempo di progetto (time-to-market) di un prodotto vengono realizzano **librerie** formate da celle logiche elementari

Libreria Standard cells

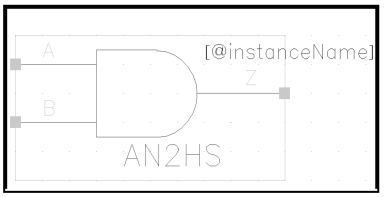
Esempio: Libreria AMS 0.35 mm (Austria Micro Systems)

270 Celle Elementari:

IO Pads	10 Inout 12 Inputs 10 Outputs 6 Power
Celle Combinatorie	5 Inverters 14 Buffers 8 3state buffers 21 and 21 or 12 xor 21 nand 21 nor 6 mux 28 Blocchi misti
Celle Sequenziali	8 FF JK 32 FF D 10 Latches

Libreria Standard Cells: Cella AND2

```
cell(AN2) {  area : 0.64
             cell_footprint : "AND2"
 pin (A B) { direction   : input
             fanout_load :   5;
            capacitance : 0.05 }
 pin(Q) { direction  : output
          max_fanout :  95;
          max_capacitance : 0.9494
          function   : "(A*B)“ }
 timing() { intrinsic_rise  : 0.22
            intrinsic_fall  : 0.12
            rise_resistance : 3.16
            fall_resistance : 2.00
            slope_rise      : 0.00
            slope_fall      : 0.00
            related_pin     : "A B"}}
```



HDL

Hardware Description Languages

- Linguaggi standard internazionale per la descrizione di circuiti integrati digitali
- Strumento convenzionale per il progetto e per la documentazione di blocchi digitali
- Permettono la rappresentazione di istanze hardware da system level fino a gate level
- Linguaggi HDL piu' comuni: VHDL , Verilog (Standard IEEE)

VHDL

Very High speed circuits Hardware Description Language

- Nato nel 1985, presso il dipartimento di difesa degli USA
- Reso pubblico nel 1987
- Formalmente ridefinito e reso standard IEEE nel 1993
- Tuttora piu' sviluppato in Europa, mentre Verilog e' considerato standard di uso comune negli USA

Strumento di DESCRIZIONE, non di PROGETTO

Stili di Descrizione Hardware

- **BEHAVIORAL**
- **Register Transfer Level**
- **Gate Level**

VHDL: Applicazioni

1) **SIMULAZIONE LOGICA**
2) **SYSTEM LEVEL DESCRIPTION**
3) **SINTESI LOGICA**

Sintesi Logica

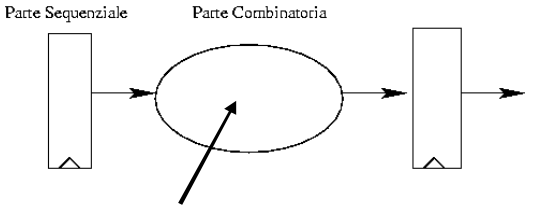
" SYNTHESIS => The process of deriving EFFICIENT results from CLEAR specifications "

Il processo AUTOMATIZZATO di sintesi logica trasforma una descrizione HDL (Behavioral) in una NETLIST di gates elementari che mantengono la stessa funzionalita'

LA sintesi logica puo' essere eseguita su un SOTTOINSIEME RTL del linguaggio VHDL detto "VHDL Sintetizzabile".
Molti costrutti VHDL NON SONO SINTETIZZABILI.
Per avere sintesi efficiente, gli elementi sequenziali (Registri, F/Fs) devono essere descritti ESPLICITAMENTE.
Inoltre, i meccanismi di sintesi NON SONO STANDARDIZZATI, quindi diversi strumenti (software) di sintesi possono dare risultati anche MOLTO diversi.
Infine, I risultati della Sintesi dipendono FORTEMENTE dalla libreria di GATES ELEMENTARI (Standard Cells) su cui e' eseguita

Sintesi logica: Da RTL ai Gates

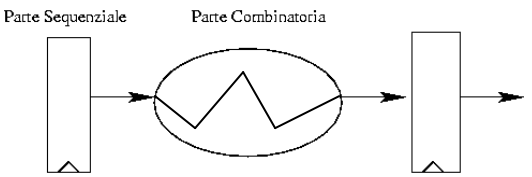
Codice RTL: rigorosa separazione tra logica Combinatoria (sintetizzabile) e logica sequenziale



SINTESI LOGICA: Il software interpreta la funzionalita' del circuito e la realizza attraverso celle elementari (STD_CELLS)
-> Mappe di Karnaugh

Gli elementi sequenziali (F/F) del circuito non vengono interpretati ma instanziati esplicitamente!

Sintesi logica: Timing Driven

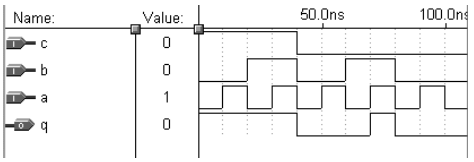


Il processo di sintesi e' **dettato dalle temporizzazioni**, si tenta di Minimizzare il piu' lungo percorso combinatorio **(critical path)** interno al design

Flusso di Sviluppo di circuiti digitali : FRONT END

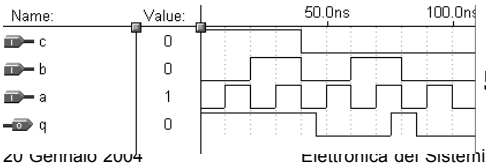
1) Definizione Algoritmica (linguaggio C): q=a * b + c;

2) Descrizione VHDL del circuito: q <= a and b or c;



3) Simulazione Funzionale

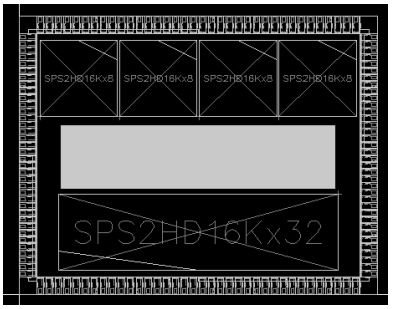
4) Sintesi Logica

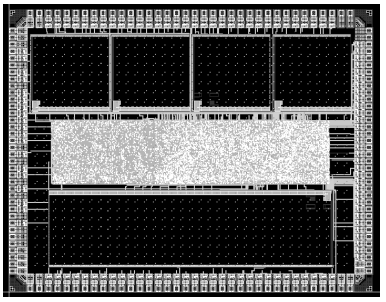


5) Simulazione Post-Sintesi

Flusso di Sviluppo di circuiti digitali: BACK END

6) Floorplanning



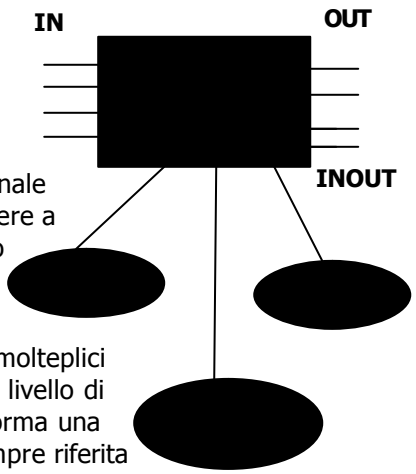


7) Place & Route

8) Parasitic extraction & backannotation

Entity e Architecture

• **ENTITY**= BLACK BOX, "scatola vuota che descrive l'interfaccia I/O del circuito



• **ARCHITECTURE** = Descrizione funzionale del comportamento del circuito. Puo' essere a diversi livelli,e puo' essere sintetizzabile o meno

Ad ogni entity possono corrispondere molteplici architectures, a seconda del tipo e del livello di descrizione voluto: es, la sintesi trasforma una architecture RTL in una a gate-level, sempre riferita alla stessa entity

Esempio

```
entity  NAND is
Port ( a , b : in bit;
         z : out bit
       )

architecture behavioral of NAND is
signal s : bit;
begin
   s <= a and b;
   z <= not s;
end  NAND;
```

VHDL vs C

Linguaggio C:

Linguaggio di programmazione software: genera una *esecuzione SEQUENZIALE* di una serie di istruzioni.

VHDL:

Tool CAD di progettazione hardware: genera una instanziazione (mapping) di risorse fisiche di calcolo CONCORRENTI.

VHDL : Costrutti

• **STATEMENTS**: Costrutto che rappresenta un istanza circuitale. Benche' vengano SIMULATI SEQUENZIALMENTE gli statements vengono ESEGUITI SIMULTANEAMENTE.

Es: A <= B and C;

Es: Z <= A + B;

Es: Z <= '1' when (A=B) else '0';

VHDL : Costrutti

• **INSTANZIAZIONE GERARCHICA**: E' possibile istanziare in un dispositivo descritto in VHDL componenti circuitali di libreria descritti separatamente,purche' la loro interfaccia di I/O sia descritta, come component, nella architecture

```
architecture structural of FullAdder is
component HalfAdder
port ( in1,in2   : in bit ;
        s,co : out bit )
end component
signal a,b,sum,carryout  : bit;
begin
   [….]
   HA1 : HalfAdder port map (in1 => a, in2 => b, s=>sum, co=> carryout);
```


VHDL: Costrutti

• **PROCESSI**: Un processo rappresenta uno "statement espanso" , ovvero una operazione non elementare composta da un insieme di operazioni elementari che si suppongono eseguite nello stesso istante.
All'interno di un processo la esecuzione e' SEQUENZIALE, e possono essere definite delle variabili.

```
process(a,b)
variable c,d: bit_vector(0 to 3);
begin
   c := a + b;
   d := c and "011"
   z <= d;
end process;
```

Sensitivity List

SENSITIVITY LIST : L'uscita di un processo viene ricalcolata ogni volta si ha un evento (cambiamento di valore) su uno dei segnali appartenenti alla sensitivity list

In VHDL sintetizzabile, I processi sono puramente combinatori e tutti gli ingressi del processo devono appartenere alla Sensitivity list.

Tipi Predefiniti

VHDL e' un linguaggio fortemente tipato, non permette Casting.

Boolean	True/False
Bit (*)	0,1
Bit-Vector (*)	Array of Bits
Character	
String	Array of Char
Integer (*)	
Positive	
Time	

(*) = Sintetizzabile

IEEE Std_logic_1164

Si tratta di un package di libreria, SINTETIZZABILE, che permette di definire segnali (o variabili, interne a processi) che descrivano il comportamento elettrico del segnale stesso

```
use ieee.std_logic_1164.all;

[.....]

entity adder is
port ( a,b  : in std_logic_vector(7 downto 0);
       z    : out  std_logic_vector(7 downto 0);
       ovf  : out std_logic );
```

0	Logic 0
1	Logic 1
X	Conflict
U	Unknown
Z	Float (3state)
H	Weak 1
L	Weak 0
-	Don't Care

Operandi Fondamentali

In VHDL un operando fondamentale puo' riferirsi un tipo predefinito, a un tipo strutturato (Array o Struct), o ad una tipologia "custom" definita in una libreria, Sia essa dell'utente o di sistema.

Gli operandi possono appartenere a tre categorie fondamentali

PORTE : Ovvero, gli elementi di interfaccia verso l'esterno del circuito

SEGNALI : Elementi di comunicazione interni al circuito tra processi, statements e components

VARIABILI : Rappresentano operatori temporanei, sono solo INTERNE A PROCESSI

Operatori Fondamentali

<=	Assegnamento
--	Commento
and, or, xor, not	Operatori Logici
+, -, *	Op. matematici sintetizzabili
/, mod, rem, abs, **	Op. matematici non sintetizzabili
>, >=, <, <=,	Operatori relazionali

Alcune Regole

- Un Segnale non puo' essere usato due volte come destinazione (corto circuito) a meno che cio' non venga fatto all'interno di un processo
- Una porta di uscita non puo' essere usata come sorgente (ingresso) da statement / processo / blocco gerarchico
- I costrutti IF/LOOP, intrinsecamente sequenziali, possono essere usati solo all'interno di processi
- LA ESECUZIONE DI DIVERSI STATEMENT / PROCESSI / COMPONENTS E' SEMPRE CONCORRENTE, l'ordine con cui sono scritti NON HA ALCUNA IMPORTANZA

Attributes

ATTRIBUTES => Funzioni di simulazione che sono "collegate" al comportamento di segnali, tipi o array. Possono essere anche definiti dall'utente

S'event	Vero in caso di eventi su S
S'active	Vero in caso di transactions su S
S'driving	Vero se il processo corrente pilota S
S'stable(t)	Vero se non ci sono eventi in S nel tempo t
T'low , T'high	Valore maggiore/minore ? T