Informazioni organizzative

- **Fonti per il materiale didattico per CE2**

1. Queste dispense sono disponibili al centro fotocopie;
2. Saranno distribuiti dei CD con tutto il materiale del corso di CE2;

- Per quanto riguarda il VHDL, nel CD saranno presenti:
  - Tutte le dispense del corso;
  - Un ambiente di sviluppo VHDL free;
  - Un semplice tutorial per l'ambiente di sviluppo;
  - I file con le descrizioni di tutti gli esempi;

3. Lo stesso materiale presente sul CD è presente anche sul sito del prof. Mazzeo - www.docenti.unina.it;

- **Tesine per l esame**
- Sono disponibili dei progetti in VHDL, sia di tipo compilativo che progettuali (saggese@unina.it);

Corso VHDL

- **Materiale didattico a cura di:**

Prof. A. Mazzeo

Ing. G.P. Saggese

Ing. A. Cilardo

Come studiare il VHDL

- Non è necessario conoscere a memoria la sintassi di tutti i comandi, l'importante è aver capito il loro significato;
- Questo non vuol dire non essere capaci di abbozzare una descrizione VHDL senza un manuale di VHDL;
- Il modo migliore di imparare è *by doing*, ovvero provando con l'ambiente di sviluppo una (buona) parte degli esempi presentati con l'ambiente di sviluppo e provando a modificarli;

Evoluzione dei sistemi digitali

- I primi sistemi "digitali" che
  - erano basati su tecnologie elettromeccaniche o elettriche,
  - operavano su informazioni in forma "numerica" (discreta) piuttosto che analogica,

  comparvero attorno al 1850 (macchina di Hollerith per elaborare i dati di un censimento).
- La diffusione della tecnologia digitale divenne rapidissima a partire dal 1970. La rivoluzione digitale è stata resa possibile da due fattori:
  - miglioramenti della tecnologie elettroniche (CMOS) e dei processi microelettronici;
  - introduzione di metodi e strumenti di progettazione automatica capaci di supportare il progetto, la verifica ed il testing di sistemi digitali di elevata complessità.

Livelli di Astrazione (1)

- Per dominare la complessità del progetto, della verifica, del testing di un sistema digitale moderno, bisogna ricorrere al concetto di "livello di astrazione".
- Un **livello di astrazione** (o vista) rappresenta il livello di dettaglio con cui "si guarda" un sistema.
  - Quanto più alto è il livello di astrazione a cui si descrive un sistema, tanto più i dettagli risultano nascosti.
  - Una descrizione di basso livello di astrazione è più vicina all'implementazione.
- **Esempio**: il circuito di controllo del cruscotto di una macchina (un tipico sistema embedded).

1) A livello di astrazione massimo (*livello funzionale*) si vede questo circuito come un sistema che a determinati stimoli (uscita del sensore di velocità delle ruote, interruttore dei fari, temperatura del liquido di raffreddamento del motore, etc.) risponde con altri segnali (pilota la lancetta del tachimetro, pilota i led che segnalano lo stato dei fari, etc.). A questo livello non interessa come è fatto il sistema, ma cosa fa.

Livelli di Astrazione (2)

2) Scendendo di livello (*livello architetturale*), si vede l'architettura del sistema, ovvero si vede che lo stesso sistema è composto da un bus di sistema, a cui è collegato un microprocessore, un insieme di interfacce per i sensori e per gli attuatori, una RAM, etc.
3) Ad un livello di astrazione più basso (*livello gate*), non si distinguono i vari componenti ma si vede che il circuito è composto da un insieme di porte logiche (alcune del microprocessore, altre di circuiti integrati dedicati, etc.), collegate opportunamente fra loro.
4) Scendendo ancora (*livello layout*) si vedono le "maschere" (ovvero delle regioni che hanno caratteristiche elettriche differenti) che permettono di realizzare un circuito integrato.
5) Scendendo ancora è possibile considerare lo stesso circuito come un insieme di elettroni che circolano attraverso le regioni conduttrici, secondo leggi fisiche ben precise.

I sistemi digitali

- Un qualsiasi sistema digitale, indipendentemente dal suo livello di complessità, può essere realizzato utilizzando un insieme di pochissimi componenti-base (*basic building blocks*).
- Questi basic building blocks dipendono dal "livello di astrazione" a cui si descrive il sistema.
  - Ad es. ad un livello RTL (Register Transfer Language) questi componenti-base sono:
    - elementi di memoria: RAM, ROM, flip-flop, shift register, …
    - elementi del data-path: mux, demux, reti combinatorie, sommatori, moltiplicatori, …
  - Ad un livello gate i componenti base sono le semplici porte logiche (NAND, NOR, NOT).

Classificazione dei sistemi digitali (1)

- Esistono 2 famiglie di circuiti digitali dal punto di vista della programmabilità:
  - circuiti general-purpose;
  - circuiti special-purpose;

1. I circuiti **general-purpose** (ad es. microprocessori e microcontrollori):
   - hanno un *set di istruzioni* base (RISC o CISC) che possono essere sequenziate (attraverso il software) per implementare "qualsiasi" algoritmo;
   - non sfruttano tutto il parallelismo di un algoritmo, overhead del ciclo di Von Neumann (fetch, decode);
   - possono raggiungere elevate frequenze di clock (2-3 GHz);
   - hanno *elevata dissipazione di potenza* (basso throughput/potenza);
   - hanno un *costo variabile* in funzione della complessità (i6051 ≈ 20 €, P4 ≈ 200 €);
   - *tempi di progetto* della parte software (programmazione) *brevi*;

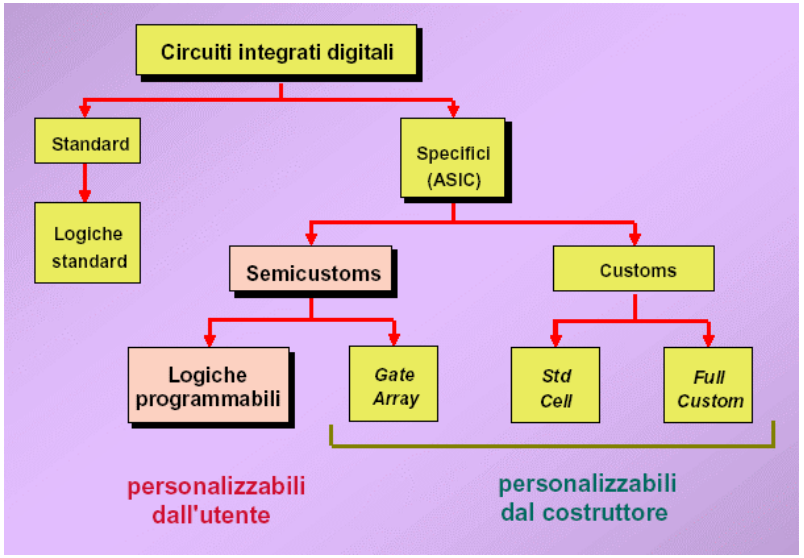
Classificazione dei sistemi digitali (2)

2. i circuiti **special-purpose**:
   - eseguono *un solo* compito;
   - possono sfruttare tutto il parallelismo insito nell'algoritmo;
   - hanno *prestazioni molto più elevate* rispetto ai processori general-purpose a parità di tecnologia (anche 5 ordini di grandezza);
   - possono raggiungere elevate frequenze di clock (in linea di principio anche 2-3 GHz, in pratica molto meno);
   - possono essere progettati in modo da essere *power-efficient*;
   - hanno *costi NRE* (non recurring engineering costs) elevatissimi anche (200 K €) e *costi di produzione bassi*;
   - tempi di progetto della parte hardware *lunghi*;

- Un sistema general-purpose è comunque un sistema che può eseguire un solo compito: il ciclo di Von Neumann;

ASIC ed FPGA

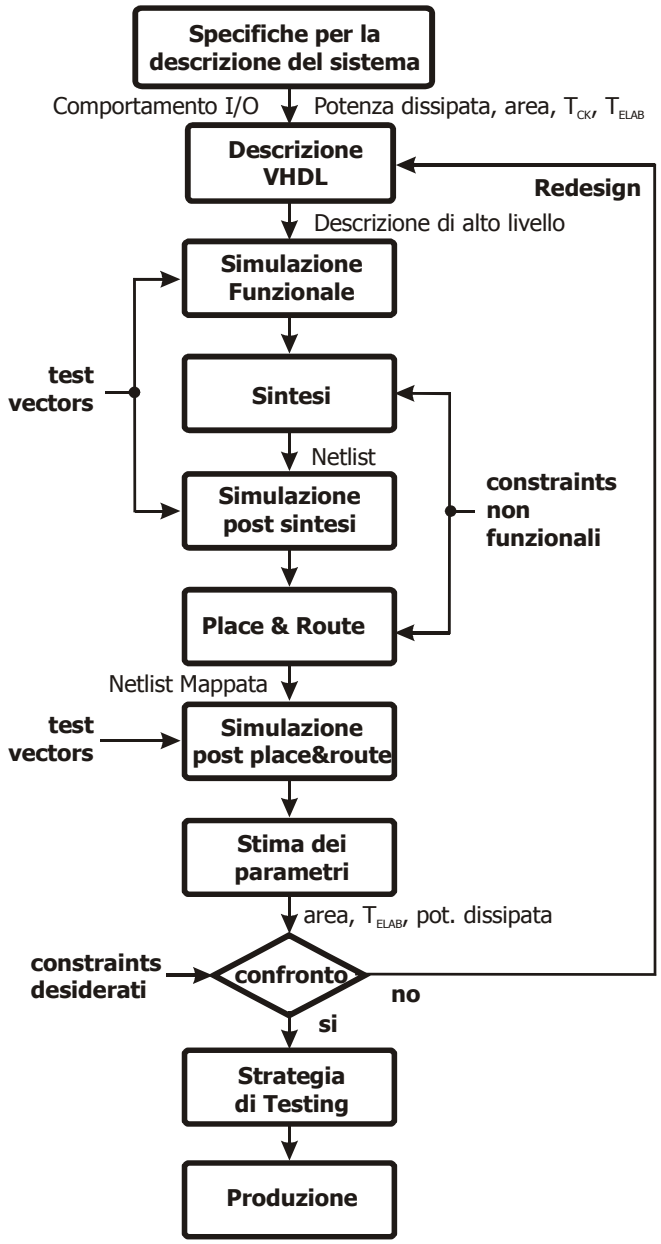
- Dal punto di vista della tecnologia è possibile distinguere:
  - i circuiti a componenti discreti (la serie 74) dai circuiti integrati (VLSI);
  - i circuiti programmabili (dall'utente) da quelli "personalizzabili" dai costruttori (ASIC);



- Oggi tra i circuiti integrati si stanno affermando le logiche programmabili (FPGA, CPLD) anche come sistemi definitivi e non solo come prototipi:
  - non hanno costi NRE;
  - basso time-to-market;
  - sempre maggiore elevata densità, performance, capacità in termini di gate;
  - riprogrammabilità sul campo;
  - frequenze massime di funzionamento basse (< 50-100 MHz);
  - maggiore dissipazione di potenza rispetto agli ASIC;

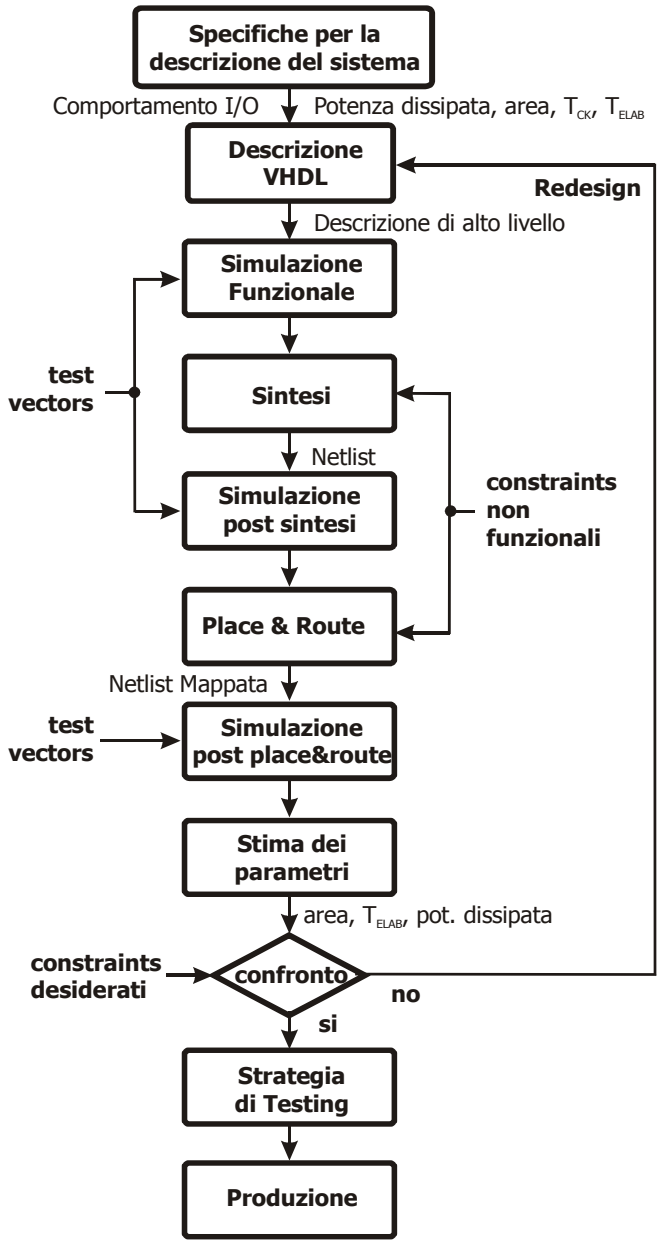
Design flow (1)

- Il flusso di progetto (*design flow*) di un sistema digitale è composto da diverse fasi (*steps*), tramite cui si passa da un insieme di specifiche alla produzione del sistema.
- In generale il processo di progetto è di tipo *top-down*: ovvero si passa da descrizioni ad alto livello di astrazione verso livelli più vicini all'implementazione.
- **Specifiche del sistema**: il committente fornisce delle specifiche sia *in termini funzionali* (comportamento ingresso/uscita, interfaccia con l'esterno, …), che *in termini non funzionali* (throughput, area del circuito, frequenza del clock, dissipazione di potenza, …).
- **Descrizione HDL**: il progettista deduce dalle specifiche uno schema ad alto livello di astrazione (l'architettura del sistema in termini di unità funzionali, percorsi di dati, segnali di controllo, …)



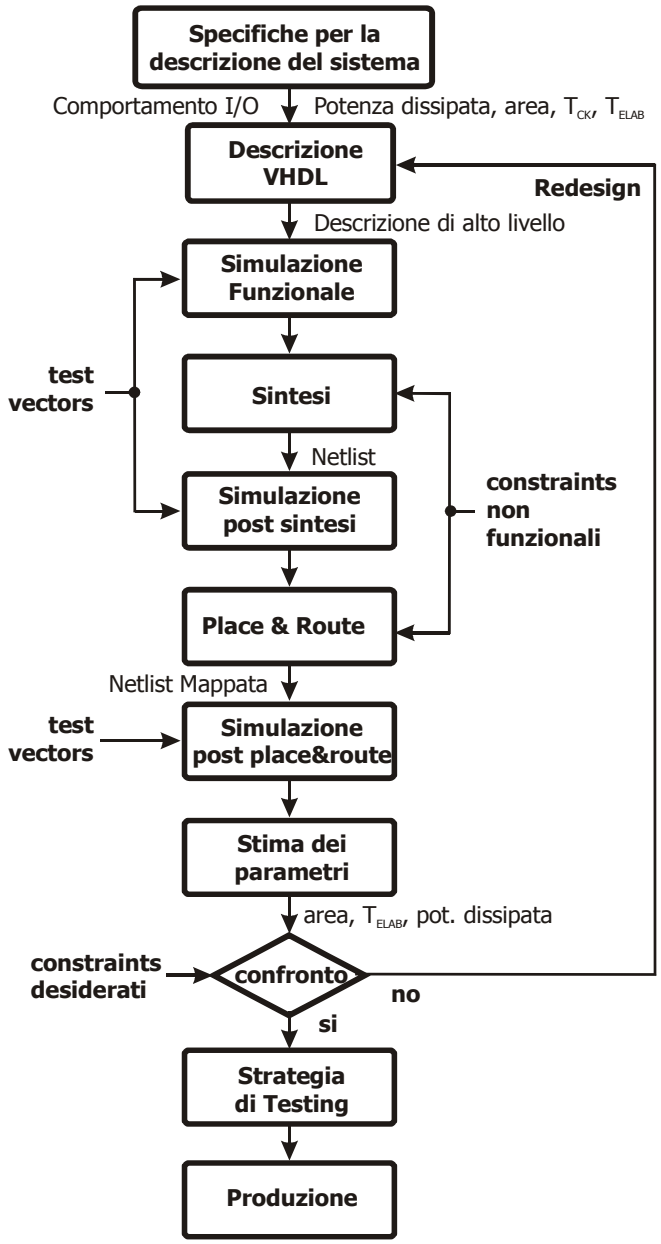
Design flow (2)

- **Simulazione Funzionale**: il sistema viene testato per verificare che il suo comportamento funzionale sia *corretto* (congruente con le specifiche);
- **Sintesi**: consiste nella trasformazione di una descrizione ad alto livello (RTL, strutturale, behavioral, …) ad una di livello inferiore (gate-level) più vicina ad una descrizione dell'implementazione;
  - Questo passo ha come risultato una *netlist* di porte logiche;
  - Si ricorre a strumenti CAD che, usando una libreria di porte logiche, un insieme di constraints e di direttive del progettista, permettono di ottenere una netlist ottimizzata del progetto;
  - Spesso c'è iterazione con il progettista, che "sovraintende" al processo di sintesi.



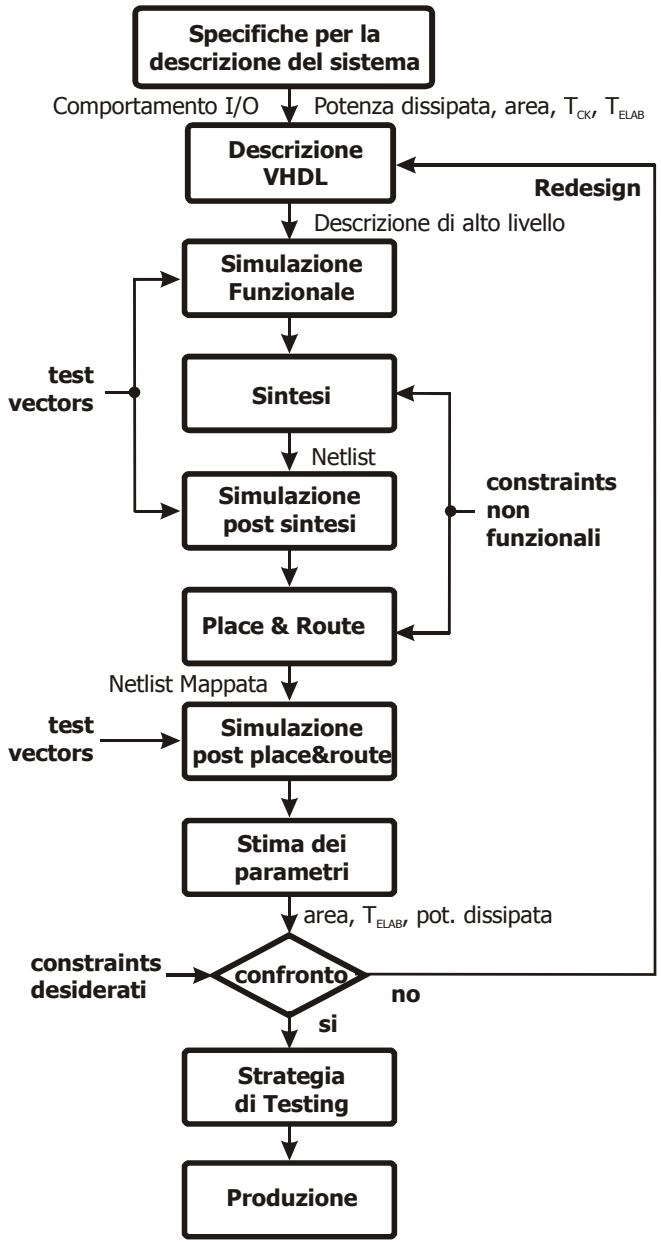
Design flow (3)

- **Simulazione post-sintesi**: evidentemente bisogna verificare che il processo di sintesi sia stato eseguito correttamente
- Bisogna assicurarsi che il comportamento ingresso/uscita del sistema desiderato e di quello sintetizzato sia lo stesso (per questo si usano gli stessi test patterns della simulazione funzionale);



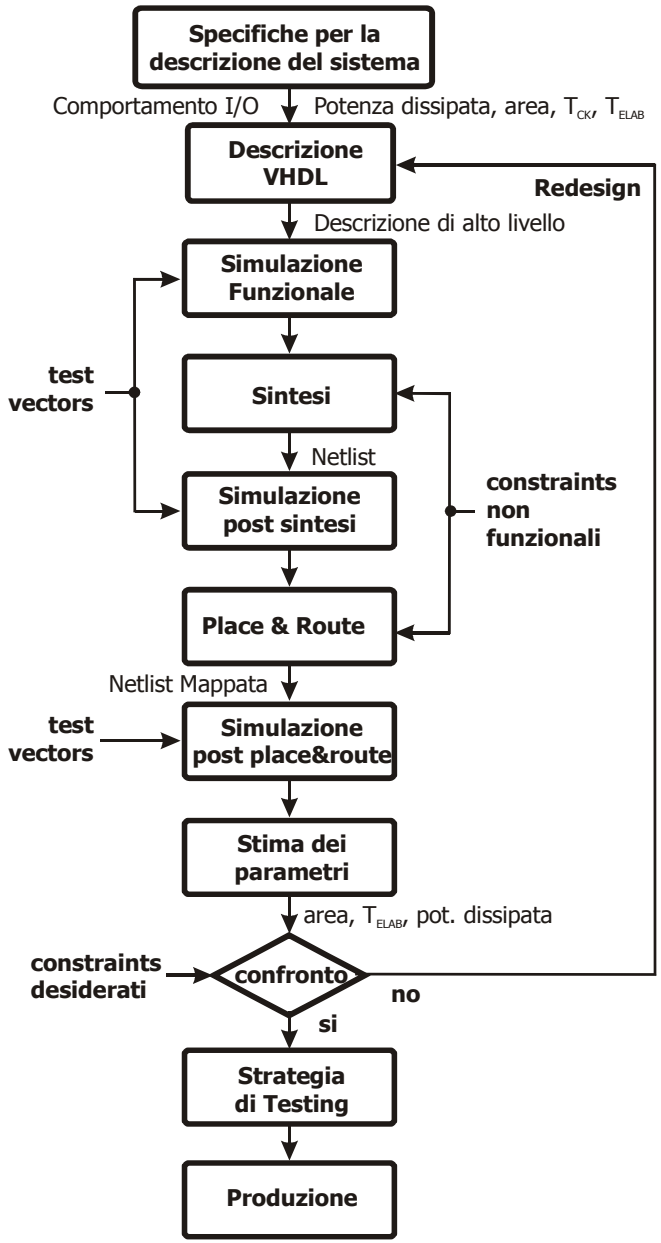
Design flow (4)

- **Place & Route**: per generare una descrizione del sistema che possa essere concretamente realizzata non basta avere la netlist delle porte, ma bisogna scegliere come "disporle" sul silicio.
  - I parametri (frequenza di clock, dissipazione di potenza, …) del circuito dipendono da quanto il processo di P&R è stato effettuato efficacemente;
  - Questo passo dipende da quello che è il dispositivo target: ad esempio per un ASIC (Application Specific Integrated Circuit) si possono decidere dove piazzare le porte logiche, mentre per i dispositivi programmabili è possibile solo scegliere quali componenti elementari, fra quelli disponibili, usare e come collegarli;
  - Come per la sintesi, anche questo passo è realizzato attraverso strumenti automatici CAD, che accettano direttive da parte del progettista.



Design flow (5)

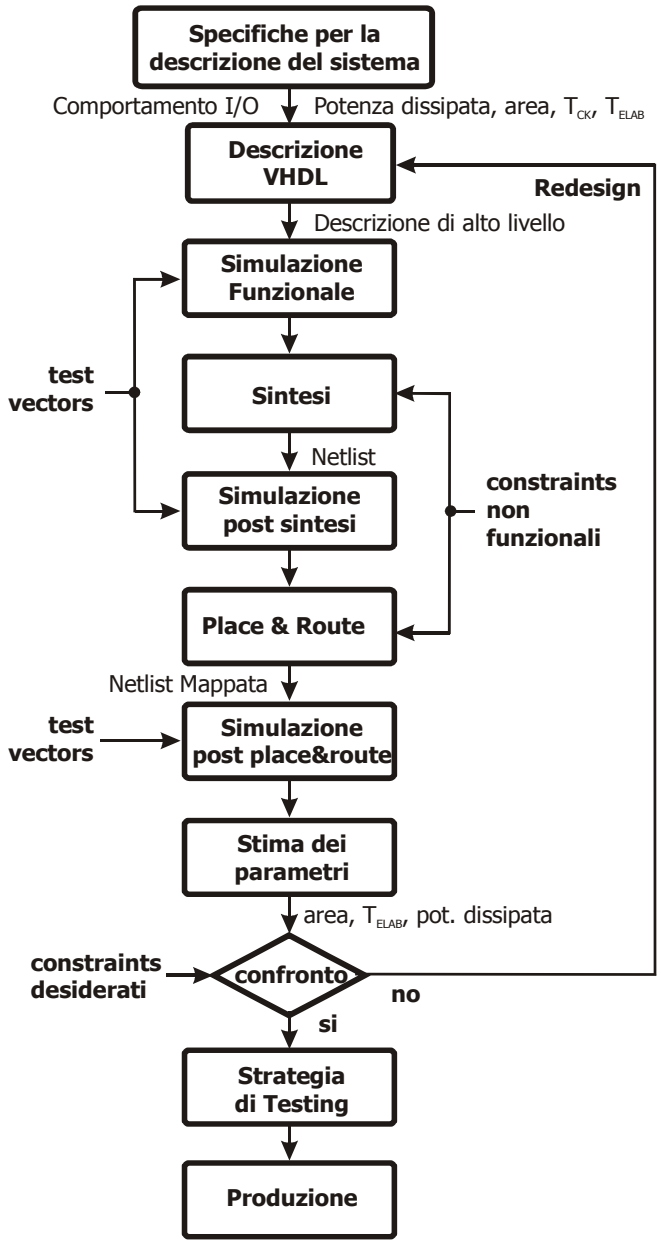
- **Simulazione post Place & Route**: anche in questo caso, dato che c'è una trasformazione fra diverse "rappresentazioni" del sistema con uno strumento automatico, è necessaria una verifica della correttezza del procedimento.



Design Flow (6)

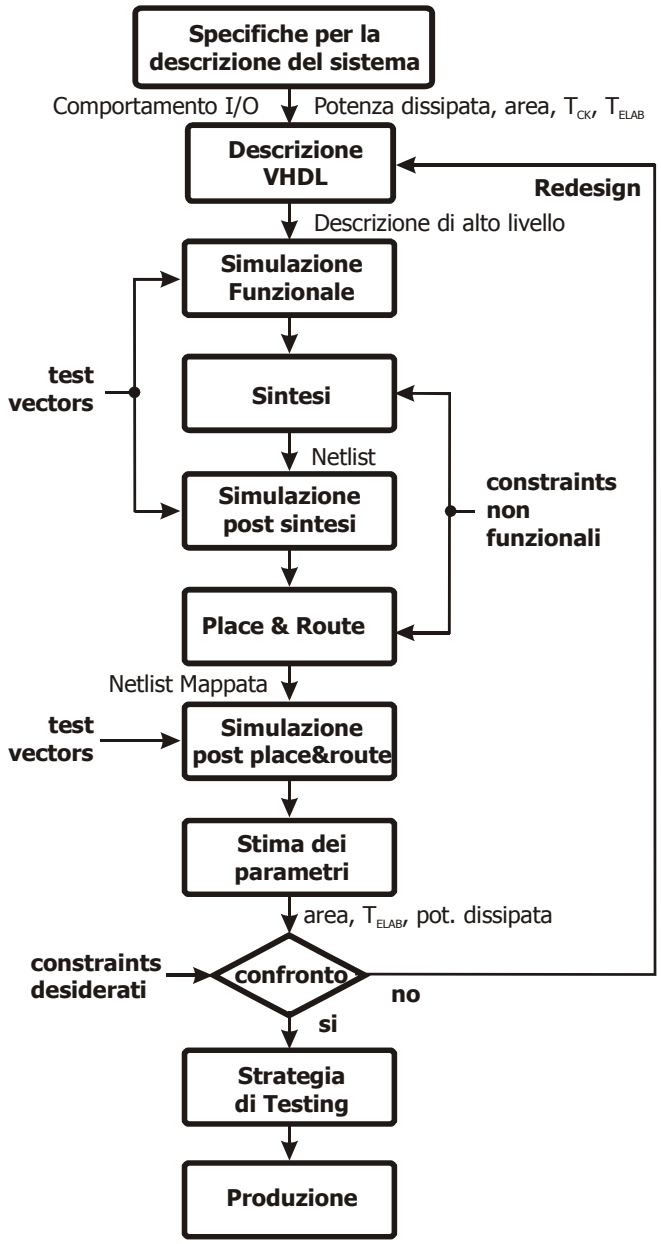
- **Stima dei parametri**: dopo la fase di P&R il sistema è descritto in dettaglio (*livello layout*), e quindi si possono estrarre i parametri *effettivi* delle funzioni di costo (area, throughput, dissipazione di potenza, …).
  - Si usano opportuni strumenti CAD per estrarre questi parametri (ad es. Static Timing Analyzer, Power Estimator, Design Rule Checker, …);

- Il flusso di design è completato con successo se i parametri effettivi del sistema progettato sono *accettabili* secondo le specifiche (constraints "desiderati") e si può procedere con l'implementazione.



Design Flow (7)

- Se le specifiche non funzionali non sono soddisfatte → **Redesign**.
- Bisogna modificare uno dei passi precedenti per cercare di migliorare le performance del sistema (area, throughput, dissipazione di potenza, …).
  - Bisogna tornare indietro nel flusso di progetto, e tanto più indietro quanto più le performance ottenute sono lontane da quelle desiderate.
  - A volte bisogna ripercorrere solo la fase di P&R dei blocchi, altre volte bisogna riprogettare tutto il sistema effettuando scelte architetturali diverse;
  - Il redesign diminuisce la produttività: è quindi estremamente deleterio e deve essere minimizzato.



Gli HDL

- Gli strumenti EDA (Electronic Design Automation) o CAD (Computer Aided Design) sono *indispensabili* per il progetto di un sistema digitale;
  - Le fasi di simulazione, sintesi, place&route e di generazione dei vettori di testing sono impossibili da svolgere manualmente, se non per circuiti con poche porte logiche;
- Il ricorso a strumenti automatici rende evidente la necessità di un linguaggio *formale* per la descrizione dei circuiti digitali;
- Inoltre un formalismo rigoroso risulta utile anche per la documentazione completa, leggibile e comprensibile.
- Per questi motivi sono stati studiati, proposti e standardizzati i Linguaggi di Descrizione dell'Hardware (**HDL**, Hardware Description Language).
  - Gli HDL più diffusi a livello industriale sono VHDL, Verilog e SystemC.
- Gli *obiettivi* di un HDL:
  - Modeling (design entry), simulation, design (sintesi, place&route, …), testing, verification;
  - Documentation;

Requisiti Semantici per un HDL

- Un linguaggio per essere un efficace HDL deve:
  - Supportare adeguatamente la *concorrenza*, infatti in un sistema digitale ci sono entità che operano in maniera essenzialmente concorrente (ad es. le singole porte logiche in un circuito);
  - Permettere di descrivere "ricorsivamente" la strutturazione di un sistema in sotto-componenti (*descrizioni gerarchiche*);
  - Fornire metodi per descrivere un sistema a *diversi livelli di astrazione*;
  - Supportare il *progetto modulare* di sistemi (librerie);
  - Supportare la *tempificazione* accurata di un sistema (ritardi inerziali, ritardi di trasporto, sincronizzazione tramite segnali di clock, …).
- Evidentemente il linguaggio C o C++ non sono buoni candidati come HDL. Infatti è stato esteso dando origine al SystemC.

Il VHDL

- Il **VHDL** è un linguaggio per la descrizione dell'hardware (HDL, Hardware Description Language).
- E' un acronimo di acronimi: VHDL = VHSIC + HDL

  VHSIC = Very High Speed Integrated Circuits

  HDL = Hardware Description Language
- E' un HDL standardizzato dalla IEEE.
  - Il progetto del VHDL incomincia nel 1981 dal DoD (Department of Defense degli U.S.A.);
  - Prima versione nel 1987: VHDL-87;
  - Nel 1993 lo standard IEEE 1076-1993 è stato aggiornato e approvato dalla IEEE: VHDL-93.
- Le due versioni si differenziano in pochi punti.
- La versione VHDL-93 è l'ultima versione dello standard.

Applicazioni del VHDL

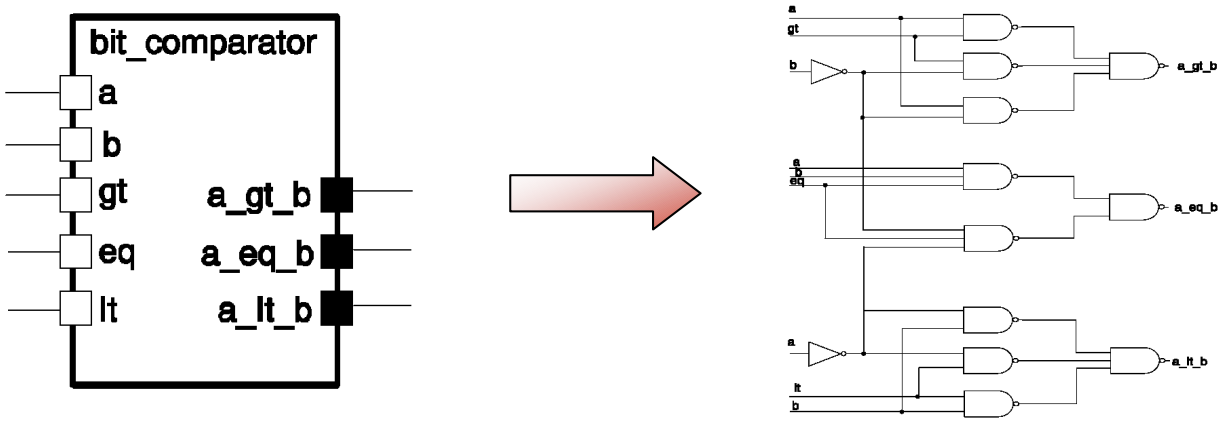
- **Documentazione**: è possibile descrivere in maniera "univoca" il comportamento ingresso/uscita e le interfacce dei sistemi digitali (e quindi le specifiche funzionali di un sistema).
- **Design and modeling**: permette una descrizione accurata di un sistema digitale ai diversi livelli di astrazione.
  - Per una descrizione strutturale si potrebbero anche usare gli *schematics*, ma questa soluzione è efficace solo per circuiti semplici.
- **Simulazione**: una descrizione in VHDL di un sistema è simulabile, ovvero può essere "eseguita" producendo le uscite ad opportuni stimoli in ingresso.
- **Verifica**: la correttezza di una descrizione VHDL è quindi anche testabile tramite simulazione (verifica via simulazione) oppure con metodi formali.
- **Sintesi**: da una descrizione VHDL, con certi limiti e con molta accortezza, può essere anche generata automaticamente tramite programmi CAD una descrizione, che può essere implementata in hardware.
  - In altre parole, una parte del lavoro necessario per la trasformazione della descrizione di un sistema, da un livello di astrazione più alto ad uno più basso che includa maggiori dettagli implementativi, può essere automatizzabile.
- **Testing:** una descrizione VHDL può essere usata per generare automaticamente i vettori di test (generazione dei test patterns).

Caratteristiche Generali

- Progetto gerarchico e modulare
  - Partizionamento di un sistema complesso in sotto-sistemi più semplici (divide et impera)
- Descrizione a diversi livelli di astrazione:
  - Possibilità di una specifica strutturale
  - Disponibilità di costrutti sequenziali (behavioral software-like constructs)
- Entity & Architecture
- Disponibilità di librerie
  - Standard Packages, Cell based design, Modularità, Re-use
- Progetto parametrizzabile
  - Generic Design
- Sottoprogrammi e funzioni
- Supporto per la gestione della tempificazione
  - Ritardi, Concorrenza

Nozioni base di Modelling con il VHDL

- Un sistema digitale può essere descritto come un **blocco** o **modulo** (*design entity*, *design module* nella terminologia VHDL) con pin (*ports*) di input e/o output che compongono la sua interfaccia con il mondo.
- I **valori elettrici** dei pin di output sono una certa funzione dei valori dei pin di ingresso, sia dal punto di vista dei valori logici assunti, che dal punto di vista della tempificazione.



- I **segnali** permettono di collegare fra loro i moduli attraverso i loro pin. I segnali in ogni istante hanno un loro valore elettrico.
- I valori elettrici dei segnali e dei pin sono modellati con un insieme discreto di valori logici ('0', '1', 'U', 'Z', 'X', …).

Design Methodology in VHDL

- La dimensione e la complessità dei sistemi digitali richiede di usare un insieme di "metodologie" diverse:
  - **Abstraction Levels** - permette di descrivere diverse parti di un sistema con un diverso livello di dettaglio, nascondendo i dettagli non essenziali (ad es. moduli che servono solo per la simulazione dovrebbero essere descritti solo a livello comportamentale);
  - **Modularity** - permette al designer di dividere blocchi complessi in blocchi più semplici, di descrivere ciascun blocco e poi di collegarli;
  - **Hierarchy** - permette di descrivere anche i blocchi componenti in termini di altri blocchi e così via ricorsivamente. Ogni livello di gerarchia può contenere livelli a diverso livello di astrazione.

- Queste tecniche/metodologie permettono di dominare la complessità di un sistema digitale (divide et impera, reuse di blocchi già testati e implementati).

- Nelle prossime slide approfondiremo singolarmente queste metodologie.

Livello di Astrazione di una Descrizione in VHDL (1)

- Il VHDL adotta la filosofia dello "**specification-and-body**" che distingue e separa esplicitamente l'interfaccia di un componente (*entity* secondo la terminologia VHDL) ed il corpo del componente (*architecture body*).
- Questo approccio consente di associare alla stessa interfaccia diversi body, che rappresentano diverse descrizioni dello stesso componente a livelli di astrazione differenti.

- Una descrizione VHDL di un sistema digitale può essere realizzata a **diversi livelli di astrazione**:
  - *Behavioral* - si specifica quale è la funzione del blocco ma non come è implementato;
  - *Structural* - si specifica un sistema in termini di sottoblocchi componenti;
  - *Data-flow* - permette la descrizione del sistema come flusso concorrente dei dati e dei segnali di controllo.

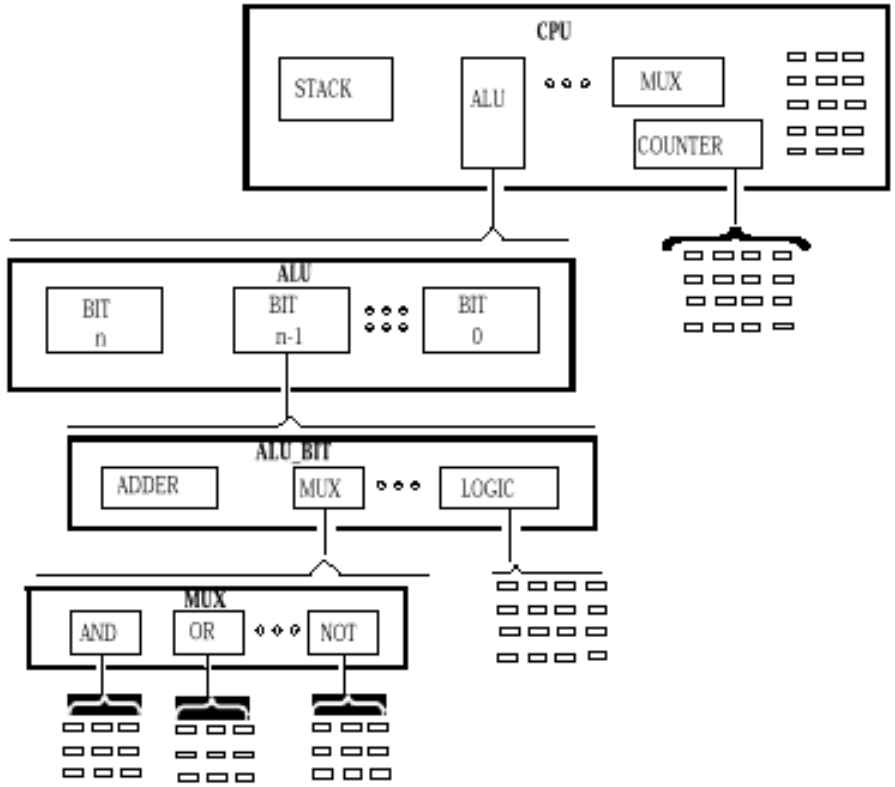
Livello di Astrazione di una Descrizione in VHDL (2)

- Esistono anche dei livelli di astrazione intermedi fra quelli citati:
  - *Register Transfer Level (RTL)* - è una descrizione del sistema in termini di registri (memorizzazione), reti logiche (elaborazione) e di spostamenti e trasformazione dei dati fra i registri.
  - *Gate Level* - è una descrizione strutturale dove gli elementi interconnessi sono delle porte logiche di una libreria di componenti;

- In generale un sistema può essere descritto anche usando *livelli di astrazione differenti per parti diverse*.
  - Ad es. per un sistema con un microprocessore e delle devices, il microprocessore potrebbe essere modellato a livello behavioral, mentre le devices a livello RTL.
- E' anche possibile mantenere diverse descrizioni a *livelli differenti di astrazione per lo stesso modulo*.
  - Ad es. una device può essere modellata a livello behavioral se è utile solo il suo comportamento e non la sua tempificazione, oppure a livello gate se sono importanti entrambi gli aspetti.

Gerarchia

- Un sistema in VHDL può essere specificato in termini:
  - delle sue *funzionalità* (descrizione comportamentale);
  - di *componenti più semplici* (descrizione strutturale → modularità/gerarchica).
- In VHDL un modulo può essere usato come un componente o può essere usato come top level module del design.
- Descrizione **gerarchica** significa che:
  - Un sistema è suddiviso in componenti più semplici (netlist di blocchi);
  - Ciascun blocco viene a sua volta scomposto in altri blocchi;
  - Il livello più basso consiste nell'usare componenti che sono disponibili in una libreria, oppure corrispondono a elementi fisici (ad es. transistor)
- Esempio: una CPU



Modularità

- La **modularità** permette il riuso di blocchi logici, che è fondamentale nel progetto di un sistema digitale, perché permette di:
  - evitare di ri-progettare, verificare, sintetizzare, e testare blocchi già progettati in precedenza, aumentando la *produttività* → uso di librerie;
  - si semplifica e si velocizza la *sintesi* perché la sintesi diventa un processo incrementale;
  - è possibile lavorare *in team* ad uno stesso progetto;
  - si può fare un *budgeting* efficace dello sforzo necessario per il progetto di un sistema, dell'area e degli altri indici di performance.

- La soluzione è quindi quella di avere una libreria (a uno o più diversi livelli gerarchici) di componenti già progettati, verificati, sintetizzati e mappati con un loro set di vettori di test.

Librerie

- Il VHDL è un linguaggio orientato alle **librerie**: quando una descrizione VHDL è compilato senza errori, il risultato viene inserito nella libreria corrente di lavoro (work library).
- Il linguaggio VHDL è organizzato sulla base di *design unit* (unità di progetto) contenute in file e compilabili separatamente.
- Il progettista lavora sulla libreria work facendo riferimento, a seconda della necessità, ad altre librerie che contengono altre design units già compilate.

- Estremamente utile allo scopo di aumentare la produttività di un designer, è avere una libreria (a uno o più livelli gerarchici e a diversi livelli di astrazione) di componenti già progettati, verificati, sintetizzati e mappati con un loro set di vettori di test.