

# Sistemi Embedded

## Flussi di progettazione (Parte 1)

*Ing. Luigi Pomante*  
*Università dell'Aquila – DEWS*  
*luigi.pomante@univaq.it*

Sistemi Embedded  
2010/2011

1

## Sommario

- Flussi di progettazione
  - Modelli
  - Metriche
  - Strumenti

Sistemi Embedded  
2010/2011

2

# Flussi di progettazione

# Flussi di progettazione

- Con il termine flusso di progettazione (*design flow*) si indica l'intero processo che parte dalla concezione e descrizione di un sistema a un livello sufficientemente formale e preciso e giunge fino al passo immediatamente precedente alla sua realizzazione fisica
- Un flusso di progettazione si compone concettualmente dei seguenti tre tipi di entità ben distinte
  - Modelli
  - Metriche
  - Strumenti

# Flussi di progettazione

- **Modelli**
  - Sono rappresentazioni del sistema durante le diverse fasi di sviluppo
    - Dalla descrizione iniziale verso la fabbricazione si hanno modelli più dettagliati/complessi e meno leggibili e/o modificabili a mano
- **Metriche**
  - A ogni passo bisogna valutare la qualità della soluzione ottenuta
    - La valutazione è molto importante se vista come relativa a soluzioni alternative
- **Strumenti**
  - Il flusso si compone di trasformazioni che, sotto la guida di metriche, arricchiscono i modelli di dettagli
    - Tali trasformazioni comportano di solito computazioni al di fuori della portata di un essere umano

Sistemi Embedded  
2010/2011

5

# Flussi di progettazione

- Lo sviluppo di un sistema embedded inizia con la sua concezione e la valutazione di aspetti economici e di mercato e si conclude con la sua fabbricazione
  - Questo processo si svolge attraverso moltissimi passi successivi che possono essere raggruppati per classi nelle seguenti fasi:
    - Ideazione e concezione del sistema
    - Descrizione del sistema e dei requisiti
    - Specifica a livello di sistema
    - Definizione dell'architettura di sistema
    - Progettazione
    - Validazione e verifica
    - Fabbricazione
      - Test post-produzione

Sistemi Embedded  
2010/2011

6

# Flussi di progettazione

## Modelli

# Flussi di progettazione

- Modelli
  - Un modello consiste in una opportuna semplificazione di un'entità originale (fisica oppure un altro modello) realizzata con lo scopo di cogliere solo alcuni aspetti specifici e ben definiti e ignorare altre proprietà che non risultano d'interesse
    - Un modello di un sistema non è qualcosa di assoluto o generale, ma piuttosto una visione relativa e specifica, costruita con l'obiettivo di rispondere a una ben precisa domanda o esigenza
  - Un secondo aspetto importante di un modello è la sua capacità di fornire un adeguato supporto ai processi di elaborazione propri delle diverse fasi di un flusso di progettazione
    - Modelli eseguibili/simulabili

# Flussi di progettazione

- Modelli (obiettivi)
  - Un modello fornisce un supporto per rispondere a una specifica domanda a proposito di un dato sistema o parte di esso e costituisce un supporto alla sua elaborazione e trasformazione
    - Si possono individuare diversi obiettivi perseguibili da un modello
      - Esplorazione architeturale
        - » Analisi rispetto a diverse scelte architetture, costi e tempi di sviluppo
      - Valutazione delle prestazioni
        - » Informazioni rilevanti relative alle "prestazioni" di un sistema (modelli di area, tempo, costo, dissipazione di potenza e altri ancora)
      - Analisi funzionale
        - » Ha lo scopo di catturare la funzionalità di un sistema a prescindere da ogni dettaglio di tipo implementativo (**simulabilità**)

# Flussi di progettazione

- Modelli (obiettivi)
  - ...
  - Sintesi
    - » Un modello per la sintesi deve cogliere tutti gli aspetti funzionali (e a volte non) rilevanti a un livello tale da consentire agli strumenti automatici di procedere al suo raffinamento
  - Validazione e verifica
    - » Queste due attività richiedono prevalentemente modelli simulabili, simili a quelli usati per l'analisi funzionale, ma generalmente più ricchi di dettagli (una eccezione sono i modelli utilizzati per la verifica formale)
  - Mapping
    - » Dato il modello di un sistema e un insieme di modelli di entità di base, il problema del mapping consiste nel determinare come combinare tali entità elementari in modo da realizzare il sistema richiesto
  - Testabilità
    - » L'analisi della testabilità di un sistema è mirata a comprendere quale porzione degli eventuali guasti hardware è rilevabile in fase di testing post-produzione oppure durante il normale funzionamento del sistema stesso (**modello di guasto**)

# Flussi di progettazione

- Modelli (proprietà)
  - Un modello, a prescindere dai suoi obiettivi, è caratterizzato da un insieme di proprietà, ovvero di caratteristiche inerenti e indipendenti dal dominio applicativo
  - Molto spesso le caratteristiche di un modello non sono immediatamente evidenti nel formalismo o nel linguaggio utilizzato per esprimerlo
    - Tali proprietà sono in questi casi da attribuirsi ai **modelli di calcolo** sottostanti, ovvero alla semantica del corrispondente linguaggio
      - Accade spesso, infatti, che linguaggi molto diversi condividano uno stesso modello di calcolo e pertanto siano semanticamente equivalenti
        - » VHDL RT e SystemC 1.0

Sistemi Embedded  
2010/2011

11

# Flussi di progettazione

- Modelli (proprietà)
  - Un aspetto essenziale di un modello di calcolo risiede nel *modello del tempo* cui fa riferimento
    - **Modelli a tempo continuo**
      - Il tempo è una variabile reale
    - **Modelli a tempo discreto**
      - Il tempo è una sequenza discreta e infinita di istanti
  - Nei modelli a tempo discreto si hanno due sotto-casi
    - Istanti sono originati dallo scorrere del tempo: **modelli time-driven**
    - Gli eventi determinano lo scorrere del tempo: **modelli event-driven**
      - Una differenza sostanziale tra i due modelli sta nell'efficienza della simulazione

Sistemi Embedded  
2010/2011

12

# Flussi di progettazione

- Modelli (proprietà)
  - Un'ultima ma importante proprietà dei modelli riguarda il loro determinismo
    - Modelli deterministici, non deterministici, stocastici
  - Nella progettazione dei sistemi embedded è molto raro incontrare sistemi stocastici o non deterministici per cui la maggior parte dei modelli utilizzati in quest'ambito ricade nella classe dei modelli deterministici

# Flussi di progettazione

- Modelli (astrazione e linguaggi)
  - Il livello di astrazione a cui si pone un dato modello determina alcuni aspetti del modello stesso
    - Quantità di dettagli rappresentati, accuratezza, generalità, flessibilità, leggibilità e manutenibilità, possibilità di essere utilizzato da strumenti automatici di sintesi o di simulazione e così via
      - Secondo una classificazione tipica e ampiamente condivisa si possono identificare sostanzialmente cinque livelli

Livello	Hardware	Software
Sistema	Matlab, Simulink, SystemC, SDL, UML, ...	
Algoritmo	Verilog, VHDL, SystemC	C, C++, Java, ...
Dataflow	Verilog, VHDL	C, Fortran, Assembly
Logico	Verilog, VHDL, EDIF, ...	Assembly
Fisico	LEF/DEF, GDSII, HEX	Binary

# Flussi di progettazione

- Modelli (astrazione e linguaggi)
  - Il più astratto è il livello di **sistema** che ha lo scopo di descrivere la struttura generale, non sempre necessariamente in termini funzionali e completi
  - Spesso, questo tipo di modelli ha come obiettivo prevalente quello di fungere da documentazione e da specifica del problema, piuttosto che come soluzione implementativa
    - In alcuni casi tali modelli possono essere simulabili, ma molto raramente sintetizzabili

Matlab, Simulink, SystemC, SDL, UML, ... **SysML**

# Flussi di progettazione

- Modelli (astrazione e linguaggi)
  - Immediatamente al di sotto, si colloca il livello di astrazione **comportamentale** detto anche algoritmico o *behavioral*
  - Lo scopo è cogliere gli aspetti funzionali del sistema, quali, per esempio, gli operatori, i dati e i meccanismi di comunicazione
    - Nel caso HW, una visione algoritmica del sistema, in generale, non contiene informazioni sul modo in cui le funzionalità saranno realizzate
      - » Limita l'utilizzabilità come punto di partenza per la progettazione automatica
    - Nel caso SW, invece, il livello algoritmico è per sua natura il punto di partenza di ogni flusso di progetto

Verilog, VHDL, SystemC C, C++, Java, ...



# Flussi di progettazione

- Modelli (astrazione e linguaggi)
  - Il livello **dataflow** fornisce una visione molto più completa e dettagliata del sistema
    - Nel caso HW, vengono definiti l'architettura d'insieme, l'architettura dei singoli blocchi, natura e dimensione degli operatori
    - Nel mondo del software, questo livello fornisce una descrizione del programma in termini di operazioni elementari
      - Tipicamente coincide con il codice **assembly simbolico**
  - I modelli a questo livello, soprattutto nel dominio hardware, sono quelli più comunemente utilizzati come punto di partenza dei flussi automatici di supporto alla progettazione

Sistemi Embedded  
2010/2011

Verilog, VHDL

C, Fortran, Assembly

17

# Flussi di progettazione

- Modelli (astrazione e linguaggi)
  - A livello **logico**, ogni parte di un sistema è descritta in termini funzionalmente non ambigui
    - Questo comporta che nessuna informazione aggiuntiva può modificare la rappresentazione funzionale data
      - Nel dominio digitale questo significa descrivere il sistema in termini di **netlist** e di **componenti elementari** la cui funzionalità è nota e fissata
      - Nel caso del software, il programma è descritto come una sequenza di istruzioni **assembly** specifiche di un dato microprocessore

Verilog, VHDL, EDIF, ... Assembly

Sistemi Embedded  
2010/2011

18

# Flussi di progettazione

- Modelli (astrazione e linguaggi)
  - A livello **fisico**, la descrizione dei sistemi HW ha lo scopo di fornire indicazioni di carattere **geometrico** (posizione celle, dimensione celle, percorsi delle net) e **fisico/chimico** (drogaggi, resistività, capacità, e molti altri)
    - A tale scopo sono molti i formalismi utilizzati, spesso legati ai fornitori della tecnologia realizzativa finale
  - Nel mondo del software, non esistendo una fase di fabbricazione, il livello fisico coincide con il prodotto finito e, pertanto, è costituito da uno o più file binari

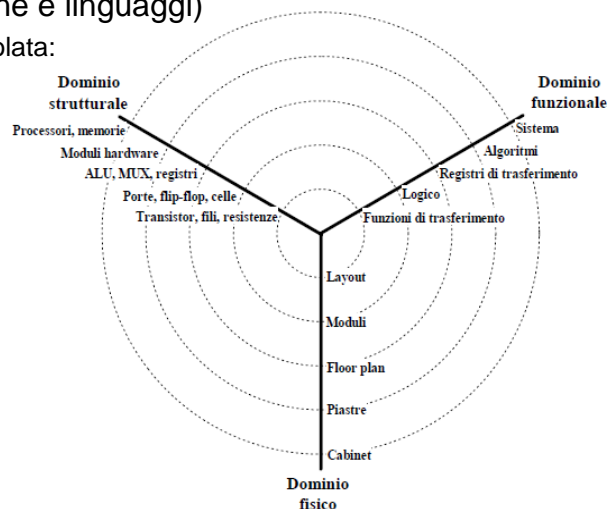
LEF/DEF, GDSII, HEX    Binary

Sistemi Embedded  
2010/2011

19

# Flussi di progettazione

- Modelli (astrazione e linguaggi)
  - Visione più articolata:  
*Diagramma a Y*



Sistemi Embedded  
2010/2011

# Flussi di progettazione

## Metriche

# Flussi di progettazione

- Metriche
  - Nelle varie fasi del flusso di progettazione è necessario valutare quantitativamente la qualità del sistema descritto da un modello
    - Funzione di un insieme di grandezze misurabili del sistema stesso
      - Es: qualità di un sistema digitale =  $f(\text{ritardo}, \text{dimensione}, \text{potenza})$
    - Una metrica, permette di ricavare da un modello una grandezza fisica esprimendola tramite una qualche unità di misura
      - Es: per un sistema digitale, una metrica di ritardo potrebbe essere il numero di livelli di logica ed essere esprimibile con un numero puro
        - » Tale grandezza non è un tempo, ma una rappresentazione indiretta
  - In un flusso di progetto eterogeneo ed articolato come quello per i sistemi embedded sono molte le metriche che si possono incontrare

# Flussi di progettazione

- Metriche
  - Prestazioni
    - Principale metrica di prestazioni: tempo di esecuzione (ritardo)
      - Nel dominio software le grandezze di uso più comune sono il **tempo di esecuzione**, i **dati di profiling**, il **tempo di risposta**, l'**overhead**
        - » A livello più basso (assembly) si usano misure più puntuali quali i **CPI** o **IPC**, il **numero medio di stalli**, l'**hit/miss rate**, ecc...
      - Nel campo dell'HW digitale le metriche sono essenzialmente: il **ritardo combinatorio**, la **frequenza di clock** e la **latenza**
      - Nel caso dei circuiti analogici dedicati, tali metriche dipendono marcatamente dalla tipologia di sistema in esame
        - » Alcune tipiche metriche sono la frequenza di taglio, la banda, il guadagno, il rapporto segnale/rumore, la linearità, e così via

Sistemi Embedded  
2010/2011

23

# Flussi di progettazione

- Metriche
  - Dimensioni (SW)
    - La misura è diversa a seconda che sia di natura statica o dinamica
      - La dimensione statica del codice è comunemente detta **footprint** e indica la quantità di memoria necessaria a contenere tutto il codice più i dati statici
        - » Questa misura vincola la dimensione dei supporti di memoria permanente, mentre non fornisce indicazioni sulla dimensione della memoria RAM
      - Metriche più astratte, utilizzate a livelli di astrazione più alti, sono, per esempio, i **LOC**, i **function point**, il **numero di task** e così via

Sistemi Embedded  
2010/2011

24

# Flussi di progettazione

- Metriche
  - Dimensioni (HW)
    - Per quanto riguarda invece il dominio hardware, le metriche di area sono le più disparate, data la natura estremamente eterogenea dei modelli che si incontrano nel flusso ai diversi livelli di astrazione
      - Nel caso di circuiti analogici: **die size** o il **numero di transistor**
      - Per i circuiti digitali si usano spesso gli **equivalent gate**
      - Nel caso di dispositivi programmabili si preferisce indicare l'area di un circuito in termini di numero di **celle** o numero di **look-up table**
        - » Si tenga però presente che le look-up table e, ancor di più, le celle relative a diverse tecnologie e diversi fornitori possono differire significativamente

Sistemi Embedded  
2010/2011

25

# Flussi di progettazione

- Metriche
  - Potenza
    - La potenza dissipata da un sistema è una misura dinamica, dipendente dai dati e viene in genere fornita come valor medio
      - La potenza è misurata in **Watt**, ma le metriche HW forniscono misure indirette quali la **switching activity** e la **capacità effettiva**
      - Per il SW, la potenza assorbita è spesso misurata come **corrente media per ciclo di clock** o di **corrente media per istruzione**
        - » Metriche meno raffinate usano il valor medio della potenza dissipata dal uP
      - Esistono poi alcune metriche relative di potenza
        - » Le più comuni sono la **potenza media in funzione della frequenza** (mW/MHz) e la **potenza media relativa alla potenza di calcolo di un processore** (mW/MIPS)

Sistemi Embedded  
2010/2011

26

# Flussi di progettazione

- Metriche
  - Testabilità
    - La testabilità è una proprietà delle reti digitali ed è misurata attraverso un'unica metrica: la **copertura**
      - In generale, comunque, è una misura della percentuale dei potenziali guasti rilevabili in fase di test post-produzione
    - Dato che si tratta di una misura utile al testing post produzione, non esiste un analogo nel dominio del software
      - Il concetto esiste solo nelle classiche fasi di verifica e validazione

# Flussi di progettazione

- Metriche
  - Affinché le metriche siano utili e di supporto al flusso di sviluppo, deve essere possibile valutarle a partire da un modello
  - Questo implica che
    - Il modello deve contenere tutte le informazioni necessarie al calcolo
    - Deve esistere uno strumento automatico in grado di effettuare tale calcolo a partire dal modello

# Flussi di progettazione

## Strumenti

# Flussi di progettazione

- Strumenti
  - La complessità dei sistemi embedded è tale che difficilmente modelli e metriche sono sufficienti da soli per ottenere risultati
    - Per un loro concreto uso è indispensabile ricorrere a strumenti software di supporto alla progettazione
      - CAD, CAE, EDA
  - Tali strumenti hanno caratteristiche e funzionalità molto diverse
    - Dominio
      - HW/SW
    - Livello di astrazione
    - Tipo di funzionalità svolta
      - Strumenti di compilazione/sintesi, strumenti di verifica/test, librerie

# Flussi di progettazione

- Strumenti

- Classificazione

	Hardware			Software		
Livello	Sintesi	Verifica	Librerie	Compilazione	Verifica	Librerie
Sistema	Code Generators	Checker Simulatori	Core	Code Generators	—	Sistemi operativi Middleware
Algoritmo	Sintesi behavioral	Cosimulatori	Core	Compilatori	Esecuzione Simbolica	Sistemi operativi Librerie
Dataflow	Sintesi RTL	Simulatori HDL	Macro Componenti	Compilatori Assembler	Macchine virtuali	Librerie standard
Logico	Sintesi logica	Simulatori HDL	Celle	Assembler	ISS	Firmware BSP
Fisico	Place & Route	Simulatori Elettrici, DRC	Transistor	Linker	ISS	—

Sistemi Embedded  
2010/2011

31

# Flussi di progettazione

- Strumenti

- Classificazione

	Hardware			Software		
Livello	Sintesi	Verifica	Librerie	Compilazione	Verifica	Librerie
Sistema	Code Generators	Checker Simulatori	Core	Code Generators	—	Sistemi operativi Middleware
Algoritmo	Sintesi behavioral	Cosimulatori	Core	Compilatori	Esecuzione Simbolica	Sistemi operativi Librerie
Dataflow	Sintesi RTL	Simulatori HDL	Macro Componenti	Compilatori Assembler	Macchine virtuali	Librerie standard
Logico	Sintesi logica	Simulatori HDL	Celle	Assembler	ISS	Firmware BSP
Fisico	Place & Route	Simulatori Elettrici, DRC	Transistor	Linker	ISS	—

Sistemi Embedded  
2010/2011

32



# Flussi di progettazione

- Strumenti
  - Compilazione e sintesi
    - I processi di compilazione e di sintesi, benché molto diversi in quanto inerenti ai due domini dell'hardware e del software, hanno in comune alcune caratteristiche fondamentali
  - Entrambi partono da una specifica, o modello, a un dato livello di astrazione e producono come risultato un nuovo modello, funzionalmente equivalente a quello iniziale, ma a un livello di astrazione più basso
    - Arricchimento del modello con una serie di dettagli aggiuntivi
      - » Tali dettagli provengono da informazioni predefinite contenute nelle librerie di supporto o dalla logica sottostante agli algoritmi di sintesi e di compilazione

Sistemi Embedded  
2010/2011

33

# Flussi di progettazione

- Strumenti
  - Compilazione e sintesi
    - Livello sistema
      - Gli unici strumenti disponibili per la compilazione e per la sintesi a livello di sistema sono i **code generators**
      - Esempi tipici sono sistemi quali Synplify DSP di Synplicity, Matlab e Simulink di The MathWorks, System Generator di Xilinx e altri analoghi
        - » La specifica consiste in uno schema grafico in cui blocchi funzionali predefiniti sono connessi attraverso linee o canali di comunicazione
        - » Il processo di generazione si limita ad accorpare opportunamente porzioni di codice (C, C++, assembly o VHDL/Verilog) e a generare una infrastruttura di controllo derivata dalla topologia del modello originale

Sistemi Embedded  
2010/2011

34

# Flussi di progettazione

- Strumenti
  - Compilazione e sintesi
    - Livello algoritmico
      - A livello algoritmico, si trovano strumenti di generazione automatica del codice e strumenti di sintesi comportamentale (per le sole sezioni HW)
        - » La sintesi comportamentale è scarsamente utilizzata a livello industriale
    - Eccezioni: strumenti specifici di un ristretto campo applicativo
      - » *Protocol Compiler di Synopsys*: a partire da una descrizione grafica simile ai **timing diagrams** genera il codice VHDL/Verilog necessario per le interfacce corrispondenti
      - » *Processor Designer di CoWare e XPRES Compiler / Xtensa Processor Generator di Tensilica*: a partire da una descrizione di un instruction set producono come risultato una descrizione VHDL/Verilog di un microprocessore custom e l'intera catena di sviluppo del software

Sistemi Embedded  
2010/2011

35

# Flussi di progettazione

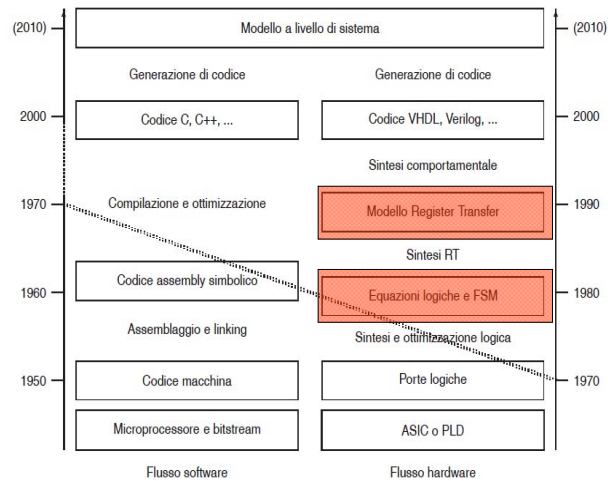
- Strumenti
  - Compilazione e sintesi
    - Livelli inferiori
      - A tutti i livelli inferiori si trova, invece, una moltitudine di strumenti estremamente efficienti, generali e flessibili
        - » Strumenti di sintesi e ottimizzazione logica, per l'hardware
        - » Compilatori e assembler, per il software
    - Dato che per tali strumenti i modelli d'ingresso e di uscita nonché la tipologia di elaborazione effettuata sono praticamente standardizzati, un'analisi più approfondita è rimandata alla analisi dei singoli flussi

Sistemi Embedded  
2010/2011

36

# Flussi di progettazione

- Strumenti
  - Compilazione e sintesi
  - Retrospectiva storica



Sistemi Embedded  
2010/2011

37

# Flussi di progettazione

- Strumenti
  - Classificazione

		Hardware			Software	
Livello	Sintesi	Verifica	Librerie	Compilazione	Verifica	Librerie
Sistema	Code Generators	Checker Simulatori	Core	Code Generators	—	Sistemi operativi Middleware
Algoritmo	Sintesi behavioral	Cosimulatori	Core	Compilatori	Esecuzione Simbolica	Sistemi operativi Librerie
Dataflow	Sintesi RTL	Simulatori HDL	Macro Componenti	Compilatori Assembler	Macchine virtuali	Librerie standard
Logico	Sintesi logica	Simulatori HDL	Celle	Assembler	ISS	Firmware BSP
Fisico	Place & Route	Simulatori Elettrici, DRC	Transistor	Linker	ISS	—

Sistemi Embedded  
2010/2011

38

# Flussi di progettazione

- Strumenti

- Verifica e validazione

- A fianco del flusso di compilazione e sintesi si colloca un flusso parallelo che ha l'obiettivo di verificare che i modelli, a ogni stadio di sviluppo, siano corretti, ovvero che soddisfino tutte le proprietà richieste dalla specifica
    - Da un'analisi delle problematiche legate alla verifica si possono individuare tre tematiche principali
      - Determinare se un modello del sistema soddisfa una o più proprietà richieste dalla specifica
      - Verificare che modelli relativi a due fasi distinte del flusso siano equivalenti rispetto a uno o più criteri predefiniti
      - Test post-produzione dell'hardware

# Flussi di progettazione

- Strumenti

- Verifica e validazione

- Determinare se un modello del sistema soddisfa una o più proprietà richieste dalla specifica
    - Caso SW
      - Esecuzione del codice sul processore target
      - Esecuzione su *Instruction Set Simulator*
        - » Code Composer Studio per i DSP di Texas Instrument, CodeWarrior o MetaWare per lo sviluppo con processori Motorola Freescale, RealView Development Suite per lo sviluppo con processori ARM Ltd
      - Profiler e i tool di code coverage
        - » gprof e gcov di GNU, IBM Rational PurifyPlus

# Flussi di progettazione

- Strumenti
  - Verifica e validazione
    - Determinare se un modello del sistema soddisfa una o più proprietà richieste dalla specifica
  - Caso HW
    - Simulazione logica/digitale: HDL a diversi livelli di astrazione
      - » Simulatori event-driven: Modelsim di Mentor Graphics, Incisive HDL di Cadence e VCS di Synopsys
    - Static Timing Analysis
      - » PrimeTime STA di Synopsys, che costituisce uno standard de facto per il mondo della static timing analysis, ed Encounter Timing System di Cadence
    - Simulazione analogica: SPICE

# Flussi di progettazione

- Strumenti
  - Verifica e validazione
    - Verificare che modelli relativi a due fasi distinte del flusso siano equivalenti rispetto a uno o più criteri predefiniti
  - Equivalence check
    - Formale o in modo empirico/euristico
      - » Formality, per esempio, è un tool di Synopsys che si occupa di equivalence checking; altri esempi sono Encounter Conformal Equivalence Checker di Cadence e FormalPro di Mentor Graphics

# Flussi di progettazione

- Strumenti

- Verifica e validazione

- Test post-produzione dei circuiti integrati

- Il testing post-produzione di un circuito digitale parte dall'assunto che la funzionalità implementata sia corretta e ha l'obiettivo di verificare che il processo di fabbricazione non abbia introdotto errori in grado di portare a un malfunzionamento del sistema

- Modelli di guasto
    - Vettori di test
      - » Automatic Test Pattern Generators
      - » Fault Simulation&Injection
    - ATE (Automatic Test Equipment)

Sistemi Embedded  
2010/2011

43

# Flussi di progettazione

- Strumenti

- Classificazione

	Hardware			Software		
Livello	Sintesi	Verifica	Librerie	Compilazione	Verifica	Librerie
Sistema	Code Generators	Checker Simulatori	Core	Code Generators	—	Sistemi operativi Middleware
Algoritmo	Sintesi behavioral	Cosimulatori	Core	Compilatori	Esecuzione Simbolica	Sistemi operativi Librerie
Dataflow	Sintesi RTL	Simulatori HDL	Macro Componenti	Compilatori Assembler	Macchine virtuali	Librerie standard
Logico	Sintesi logica	Simulatori HDL	Celle	Assembler	ISS	Firmware BSP
Fisico	Place & Route	Simulatori Elettrici, DRC	Transistor	Linker	ISS	—

Sistemi Embedded  
2010/2011

44

# Flussi di progettazione

- Strumenti
  - Librerie
    - Il terzo insieme di strumenti di supporto alla progettazione consiste nelle librerie
      - Il concetto, la dimensione, la struttura e il contenuto delle librerie hanno cambiato forma negli anni, evolvendosi da semplici collezioni di funzioni di base a complessi sistemi di sviluppo per la generazione automatica di funzioni o di macro
        - » Questa evoluzione si è vista sia nel dominio hardware sia in quello software

# Flussi di progettazione

- Strumenti
  - Librerie (HW)
    - Nel dominio hardware digitale negli anni '70, una libreria consisteva in un insieme di porte logiche di base e di qualche funzionalità appena più complessa
      - Le librerie di fatto erano le celle fornite dalle *silicon foundry*
    - Il primo grande passo è dovuto alla diffusione degli HDL
      - Secondo questo paradigma, la specifica di un circuito può contenere descrizioni sintetiche, piuttosto astratte e concise, di componenti complessi e gli strumenti di sintesi sono in grado di riconoscere tali operatori e inferenziare gli opportuni blocchi funzionali
        - » DesignWare di Synopsys

# Flussi di progettazione

- Strumenti

- Librerie (HW)

- Al crescere della dimensione dei sistemi e con la necessità di colmare il *productivity gap*, la complessità dei blocchi forniti dalle librerie è cresciuta costringendo progettisti ed EDA vendor a modificare il loro approccio

- Per componenti complessi il progettista deve istanziare un componente, specificandone tutti i parametri poiché l'inferenziazione automatica non è più una strada percorribile

- Macro, core o IP

- » Per le tecnologie programmabili alcune macro sono fornite come parte del sistema di sviluppo mentre altre, le più complesse e critiche, sono invece fornite a pagamento
        - » Nel caso dei flussi ASIC, invece, la maggior parte dei componenti è fornita separatamente dal design kit e con forme contrattuali spesso complesse

Sistemi Embedded  
2010/2011

47

# Flussi di progettazione

- Strumenti

- Librerie (HW)

- A un livello funzionale ancora più elevato si collocano i core di microprocessori e di microcontrollori

- Per le logiche programmabili, i principali produttori di FPGA forniscono uno o più core, appositamente sviluppati per i loro dispositivi
        - » MicroBlaze e PicoBlaze forniti da Xilinx, Nios fornito da Altera

- Per gli ASIC, esistono società terze che sviluppano core generici e li forniscono unitamente a tutta la documentazione necessaria
        - » ARM sviluppata dalla ARM Limited

Sistemi Embedded  
2010/2011

48



# Flussi di progettazione

- Strumenti

- Librerie (HW)

- Per le macro di fascia alta, sia nel mondo ASIC sia nel mondo delle logiche programmabili, esistono modi diversi di fornire i componenti

- A differenziare gli approcci è la libertà di modifica lasciata al progettista

- Soft macro

- » Componenti HDL a livello RT indipendenti dalla tecnologia

- Firm macro

- » Componenti per una specifica tecnologia sotto forma di netlist HDL

- Hard macro

- » Componenti per una specifica tecnologia sotto forma di netlist tecnologiche in un formato standard (EDIF o GDSII) o proprietario (XNF/UCF di Xilinx)

# Flussi di progettazione

- Strumenti

- Librerie (SW)

- Paradigmi, tecnologie e problemi analoghi esistono anche nel dominio software

- Sempre secondo una prospettiva storica, le prime librerie software erano costituite da collezioni di semplici funzioni, sviluppate in assembly

- » Non portabili

- Con lo sviluppo di compilatori efficienti e con la diffusione dei linguaggi di alto livello le librerie di funzioni hanno cominciato a essere sviluppate usando tali linguaggi per venire poi compilate per macchine specifiche e distribuite sotto forma di codice macchina o binario

# Flussi di progettazione

- Strumenti
  - Librerie (SW)
    - Anche nel mondo del software, la crescente complessità delle applicazioni ha portato a un corrispondente aumento delle dimensioni delle librerie
      - Sia in termini di numero di funzionalità fornite, sia in termini di complessità delle stesse
        - » Librerie standard di supporto ai linguaggi: Standard C Library
        - » Librerie matematiche: GSL (GNU Scientific Library)
    - Si aggiungono poi le librerie per lo sviluppo di interfacce grafiche che hanno vissuto una rapidissima evoluzione negli ultimi anni
      - X11 o Motif per i sistemi Unix/Linux
      - Librerie moderne quali GTK/GTK+ oppure le varie versioni di Qt

Sistemi Embedded  
2010/2011

51

# Flussi di progettazione

- Strumenti
  - Librerie (SW)
    - Una classe completamente diversa di moduli software di supporto, sono costituiti dal software di base per applicazioni embedded
      - BSP (Board Support Package)
        - » Componente software di base che offre il primo e più basso livello di astrazione dell'hardware di un sistema
      - BIOS (Basic Input/Output System)
        - » Collezione di funzioni di base per la gestione dell'interfacciamento di un microprocessore o microcontrollore con le periferiche
      - Sistemi operativi embedded
        - » Generici: Linux Embedded, eCos, VxWorks, QNX, Windows CE
        - » Per specifici campi applicativi (es. WSN): TinyOS, Contiki, Mantis

Sistemi Embedded  
2010/2011

52

# Flussi di progettazione

- Strumenti
  - Librerie (SW)
    - Anche per il software, come già visto per le librerie hardware digitali, esistono diverse modalità di distribuzione
      - Formato sorgente
        - » Offrono la possibilità allo sviluppatore dell'applicazione di apportare modifiche al codice e di usarle in diversi ambienti
      - Formato binario
        - » Pre-compilata per una specifica famiglia di microprocessori e distribuite sotto forma di codice macchina eseguibile. Tali librerie non sono modificabili dall'utente finale, il quale conosce unicamente l'interfaccia (API)
    - Si tenga presente che anche per il software, quando la complessità dei componenti di libreria cresce, la nozione di libreria statica, cioè di raccolta di funzioni, sfuma in quella di ambiente di sviluppo e di generatore di codice