Embedded & IoT Systems Modelling

Franco Fummi

UNIVERSITÀ di VERONA
Dipartimento di INFORMATICA

Version 1.6

Contents

- Hw/Sw Systems Description
  - embedded systems
- Architecture
- Simulation problem
- Modeling problem
- Current solution

Introduction

- Electronic systems consist of:
  - HW platform
  - SW application layers → SW per una specifica app, interfacce e S.O. dedicati
  - Interfaces (S.O.; HAL = Hw Abstraction Layer)
  - Analog components
  - Sensors and transducers
- Main trends:
  - Migration from analog to digital processing
  - Broader system-level integration to support System-On-a-Chip (SOC) approach

potrebbe anche non essere presente => ultimamente con linux un S.O. c'è ovunque

MEMS = Micro mechanical electronic system
Parti meccaniche implementate sul silicio => ACCELEROMETRO

5% dei progetti usano SOC, ma sono più economici se venduti pochi

SOC → un solo chip
≠
FPGA → SOC implementati su una board con altre componenti (chips)
95%

Spostare un progetto da FPGA a SOC è un progetto dispendioso e pericoloso => TAPE-OUT, cioè la prima stampa costa ½ o 1 mln di $ e se non funzionano sono da BUTTARE

Challenges in the design of embedded systems

- increasing application complexity even in standard and large volume products non c'è mai la possibilità di partire da zero in un progetto!
  - large systems with legacy functions
  - mixture of event driven and data flow tasks
  - flexiblity requirements
  - examples: multimedia, automotive, mobile communication
- increasing target system complexity
  - mixture of different technologies, processor types, and design styles
  - large systems-on-a-chip combining components from different sources (IP market)
- numerous constraints and design objectives
- reduced and overlapping design cycles

⇒ IP-cores
chip preesistenti
o codice già esistente o librerie preesistenti di board FPGA

TTM = Tempo massimo in cui lanciare sul mercato il prodotto

tempo di progettare un nuovo prodotto

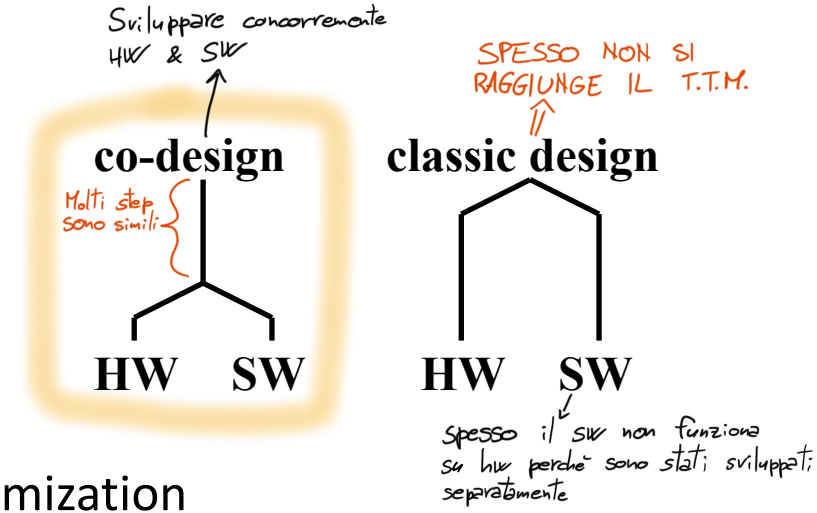
SE SFORO IL TTM PERDO GLI EARLY ADOPTERS E QUINDI NON DIVENTA POPOLARE

EISD IL MIO PRODOTTO E PERDO MOLTO GUADAGNO

Hardware/software codesign

- Hardware/software co-design:
  - Combined design of hardware and software
- Goals
  - design process optimization
    - Increased design productivity
  - design optimization
    - Improved product quality
- Tasks
  - co-specification and co-modeling
  - co-verification
  - co-design process integration and optimization
  - design optimization and co-synthesis

METODO MIGLIORE



SystemC → linguaggio per descrivere sia sw che hw che poi divide sw in c++ e hw in VHDL (per esempio)

SysML → Framework derivato da UML per descrivere hw e sw

Co-design advantages

- Explore different design alternatives in the architectural design space
- Tune HW to SW and vice-versa
- Reduce the system design time
- Support coherent design specification at the system-level
- Facilitate the re-use of HW and SW parts
- Provide integrated environment for the synthesis and validation of HW and SW components

Digital system classification

APPLICATION DOMAIN
- Computer
- Telecom
- Automotive

SYSTEM TYPE
- General-Purpose Systems
- Embedded Systems

I nuovi sistemi embedded sono così complessi che di fatto sono general purpose

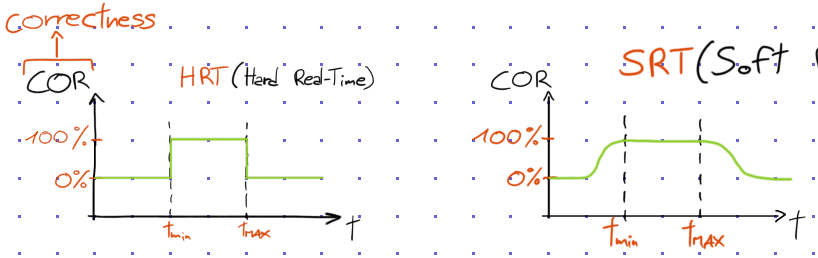
PROGRAMMABILITY
- Application-level
- Instruction-level
- Hardware-level

Dipende dai limiti e requisiti
JAVA potrebbe non andare bene

Il linguaggio base è C ⇐

IMPLEMENTATION
- Technology
- Design style
- Level of integration

REAL-TIME: il sistema deve funzionare in un preciso intervallo di tempo $[t_{min}, t_{MAX}]$



Se una funzione è CRITICA e/o viene eseguita spesso posso decidere di codificarla su hw in modo che sia più veloce

Co-design of embedded systems

- Design of dedicated computing and control systems
- Embedded controllers
  - On-line control of manufacturing process
  - Robots guidance and control
  - Aircraft, automobile and ship control
- Data processing and communication systems
  - Telecom
  - Radio-navigation

Co-design of embedded systems

- Design of dedicated HW parts
  - Different design styles:
    - Co-processors, embedded cores, ASIPs, ...
  - Widely varying design scale
- Design of dedicated SW parts
  - Special-purpose operating systems
  - Drivers of peripheral devices

Application Specific Instruction Processor ⇒ Microprocessore con ISA standard ma con istruzioni aggiuntive specifiche per un certo ambito

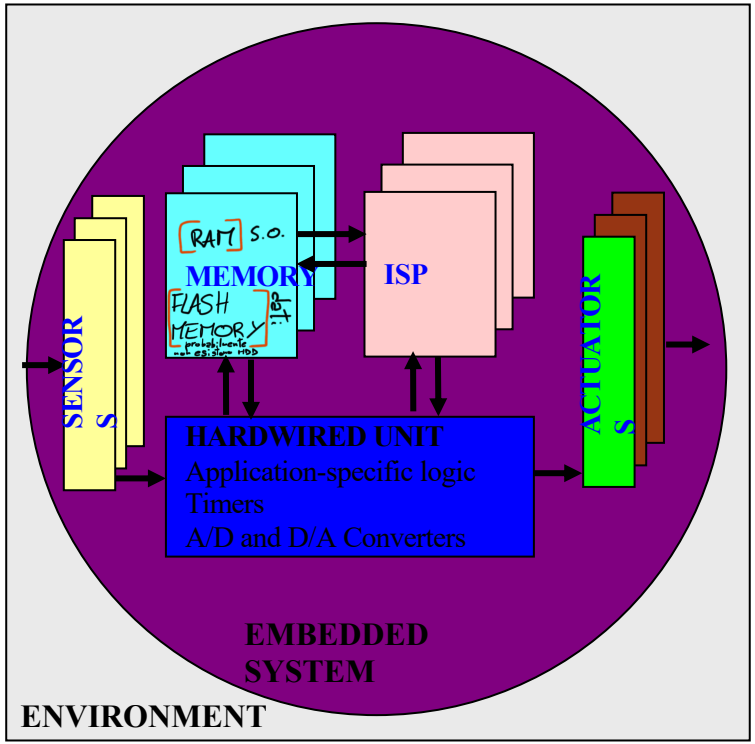
Micro processori specializzati per certe funzioni

Tool per personalizzare s.o. standard togliendo i moduli che non servono

Se uso un s.o. chiuso, mi affido al distributore ma mi assumo la responsabilità in caso di problemi

Embedded systems

Cyber-Physical System



PLATFORM-BASED DESIGN

Cerco una piattaforma precsistente che mi potrebbe andare bene e la personalizzo in base a cosa mi serve

Array-based design

- Pre-Diffused Array or Mask Programmable Gate Array (MPGA)

- Pre-Wired Array or Field Programmable Gate Array (FPGA)
  - Soft-Programmable (Memory based)
  - Hard-Programmable (Anti-fuse based)

Integration level

- Single chip systems (SOC approach):
  - ASICs with embedded cores and memories
  - Cores (microprocessor, microcontroller, DSP, ...)
- Multiple chip systems:
  - ASICs, FPGAs, …
  - Memories
  - Programmable components such as processors, DSPs or controllers
    - Off-the-shelf or proprietary components
- Distributed systems

Comparison

HW: 

hw è 10^2 o anche 10^3 volte più veloce di sw

SW: ANDL %EAX, %EBX

Se è possibile soddisfare i requisiti di performance e non sforare il consumo massimo con la soluzione sw, allora usare quell'approccio!

	IP (SW)	ASIP	ASIC
Performance	+	++	+++
Power	+++ (Consumo alto di corrente)	++	+
Reuse	+++	++	
HW design effort		++	+++
SW design effort	+	++	

Embedded system requirements

- Reactive systems:
  - The system never stops
  - The system responds to signals produced by the environment
- Real-time systems:
  - Timing constraints on task evolution
  - Hard and soft constraints

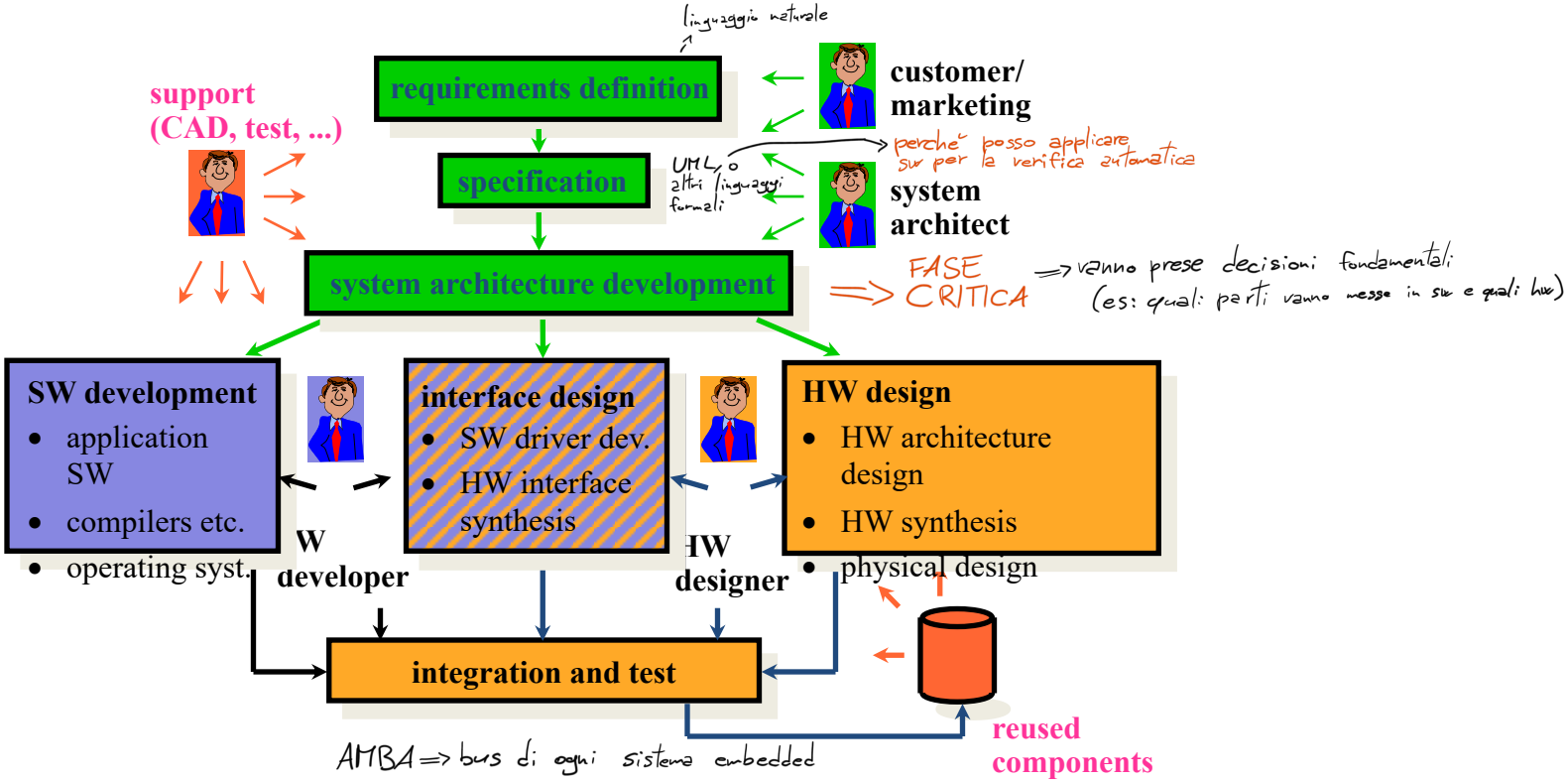
Specific steps in embedded control design

- Architecture selection:
  - Standard microcontroller or microprocessor
  - ASIC
  - ASIC with embedded core or co-processor
- Technology selection for HW resources
- Design dedicated HW, SW and interfaces

Co-design flow of embedded systems

- Modeling, validation and synthesis
  - System-level simulation
- Homogeneous modeling
  - HW/SW partitioning
  - HW/SW or SW/HW migration
- Heterogeneous modeling
  - Direct implementation and re-targeting
- Co-synthesis
  - HW and interface synthesis
  - SW compilation and code generation
- Co-simulation

Embedded systems design process



Observations on the design process

- Increasingly concurrent design of hardware and software with partially incomplete or variable specification
  - tight and permanent cooperation of hardware and software designers, system architects and customer/marketing required
- Narrow time-to-market windows require a safe "first-time-right" design process
  - early detection of systematic design flaws is crucial
  - reliable design times and precisely predictable product data are more important than design time minimization
    prerequisite: reliable estimations - today: designer experience

Observations on the design process

- Increased productivity through reuse of components and functions
  - function and component libraries required
  - problem: function migration between different technologies, between hardware and software

State of the practice

- Co-simulation as a support of design (process) integration
  - extension of simulation techniques to combined simulation of hardware and software components
  - allows permanent control of hardware and software component consistency
  - supports early validation of reused component integration
- Integration validation more costly with increasing level of detail
  - current focus on co-simulation for lower levels of a design
  - simulation with models of specific processors, memories, busses, ...
  - reduction of accuracy mainly to improve simulation performance
  - examples: Mentor Seamless CVS, Viewlogic Eagle

State of the practice

- "Executable" co-specification used as a basis for system validation
- Virtual prototyping
  - simulation based validation
  - many commercial examples for different applications Statemate (i-Logix), MatrixX (ISI), MATLAB (TheMathWorks)
  - RASSP program (DARPA)
- Rapid prototyping with "hardware-in-the-loop"
  - hardware supported system emulation Þ real environment
  - often custom design

State of the practice

- Executable co-specification problems
  - combination of domain specific languages and semantics
  - integration of reused functions and components in abstract model
  - inclusion of non-functional constraints

Specification languages

- Different communities
  - VLSI system design — VHDL, Verilog, SystemVerilog…
  - DSP — COSSAP, SPW, …
  - Continuous design — MATLAB, MATRIXX, ….
  - Synchronous system design — Esterel, Lustre, Statechart
  - Classical programming — C, C++, Java, ….
  - Functional and algebraic — VDM, Z, B, Funmath, ….
  - System design methods — SystemC, ….

Concepts for system level specification

- CONCURRENCY
  - different levels (bit, operation, statement, process, system)
  - two types: data-driven, control-driven
- HIERARCHY
  - needed for structured design methodologies
  - Two types: behavior, structure
- COMMUNICATION
  - data exchange between concurrent subsystems
  - two types: message passing, shared memory
- SYNCHRONIZATION
  - Two models: synchronous, asynchronous

Example of Specification Language

- SDL
  - well-suited for control-intensive, real-time systems
  - flow chart FSM, both graphics and text
  - abstract data types
  - dynamic process creation
  - synchronization via blocking, RPC
  - can monitor performance constraints

Example of Specification Language

- StateCharts, UML Charts
  - graphical FSM of states and transitions
  - addition of hierarchical states for modeling complex reactive behaviors
  - SpecCharts adds
    - behavioral completion
    - exceptions
  - may attach VHDL code to states and transitions arcs
  - extended with arithmetic
  - Easy to use for control-dominated systems

Simulation and debugging requirements

- Embedded controllers:
  - ASICs plus SW running on a processor
  - VHDL or Verilog plus C programs
  - Weakly heterogeneous systems
- Embedded data processing and communication systems
  - ASICs plus SW running on a processor or ASIP
  - Environmental modeling (e.g. telephone lines)
  - Strongly heterogeneous systems

Co-simulation

Vari simulatori effettuano una simulazione del sistema => Considera che il sistema andrà più lento perché devono comunicare tra loro

- Simulate at the same time both hardware and software
- Two conflicting requirements:
  - execute the software as fast as possible
  - keep hardware and software simulations synchronized so they interact as they will in the target system

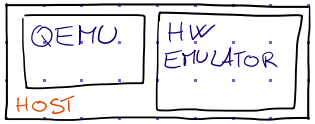
SW → codice binario

HW

Host (fisico) ← processore → Target (virtualizzato)

QEMU: simulatore che simula le istruzioni di un altro processore

Il problema è che il pc host non ha tutto l'hw del target ⇒ QEMU va co-simulato con un simulatore di hw; questa soluzione si chiama VIRTUAL PLATFORM



Co-simulation

- Desired features:
  - Level of timing accuracy
  - Speed of simulation runs
  - Visibility of internal states
- Potential problems:
  - Meaningful results are obtained with large SW programs
  - Model availability
  - Strong heterogeneity requires specialized environment

Co-simulation paradigms

- Homogeneous modeling:
  - HW models in HDL
  - Processor model in HDL
  - SW in assembly code
- Usage of HDL simulator for the whole system including the processor model
- Simple method but quite inefficient

Co-simulation paradigms

- Weakly heterogeneous systems
  - a) HDL simulators with processor model
  - b) Compiled SW
  - c) HW emulation
- Strongly heterogeneous systems
  - Require specialized simulation environments (e.g. Ptolemy)
  - Communication mechanisms among domains and their corresponding schedulers

HDL processor modeling

- Precise timing model
  - Accurate timing and complete functionality
    - Event-driven simulation
- Zero-Delay Model (ZDM) for timing
  - Correct transitions at clock edges
    - Cycle-based simulation
- Instruction-set simulator
  - Model emulates processor while insuring correct register and memory values

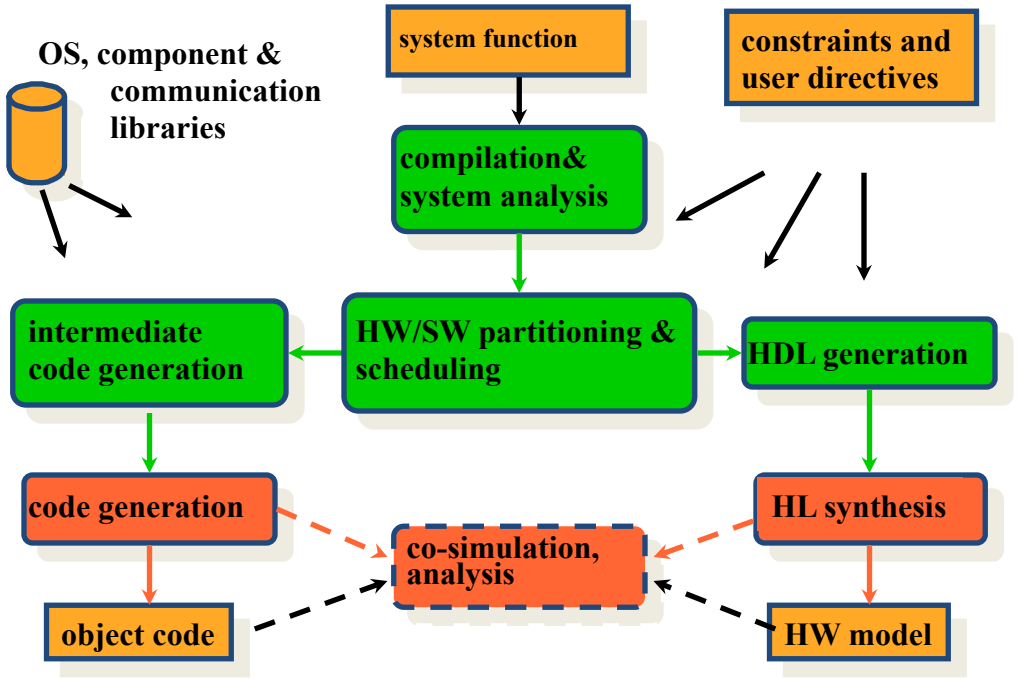
Compiled SW

- Basic assumption:
  - HW/SW communication protocol such that communication delay has no effect on functionality
- SW is compiled and linked to simulator
- HW/SW communication is replaced by handshake
- Simulation speed is limited by HW simulation speed

HW emulation

- HW mapped onto programmable HW
  - One order of magnitude loss in speed
- Programmable HW boards connected to workstations
- Limited visibility of internal states

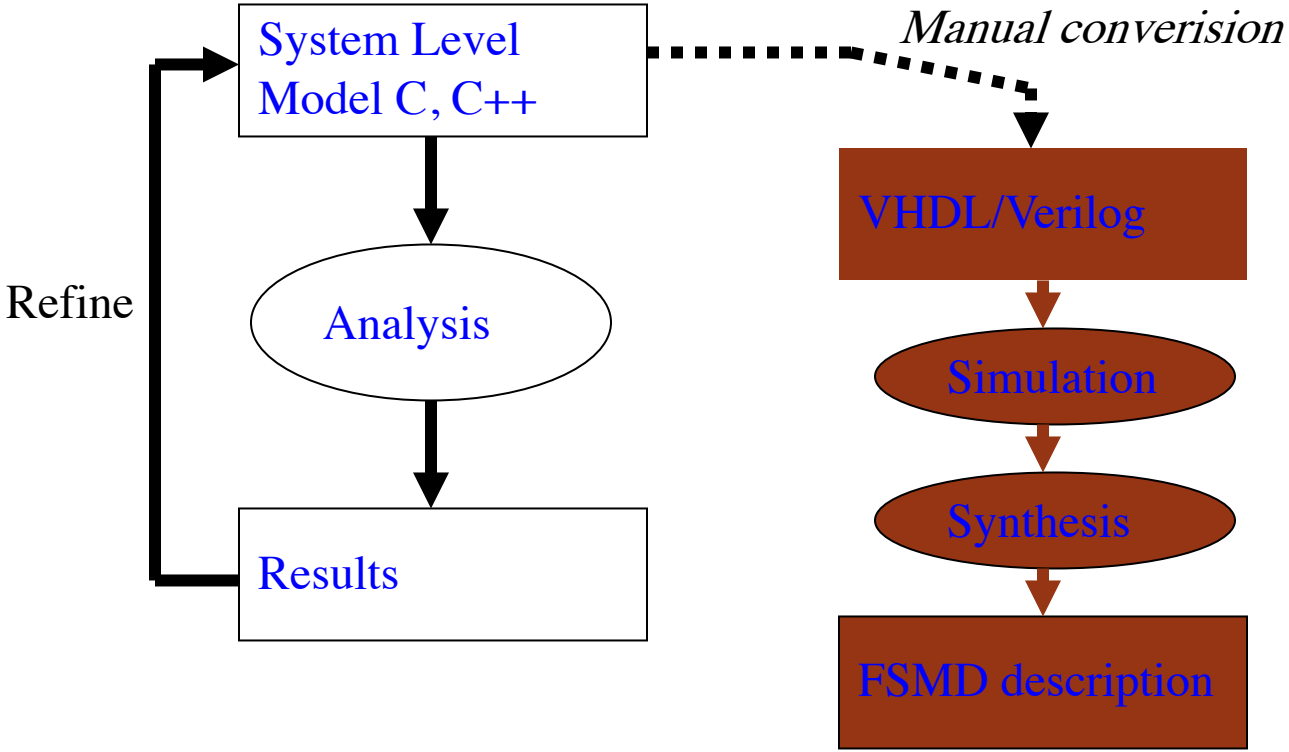
Co-synthesis design flow-Principle



Modeling Current Solution

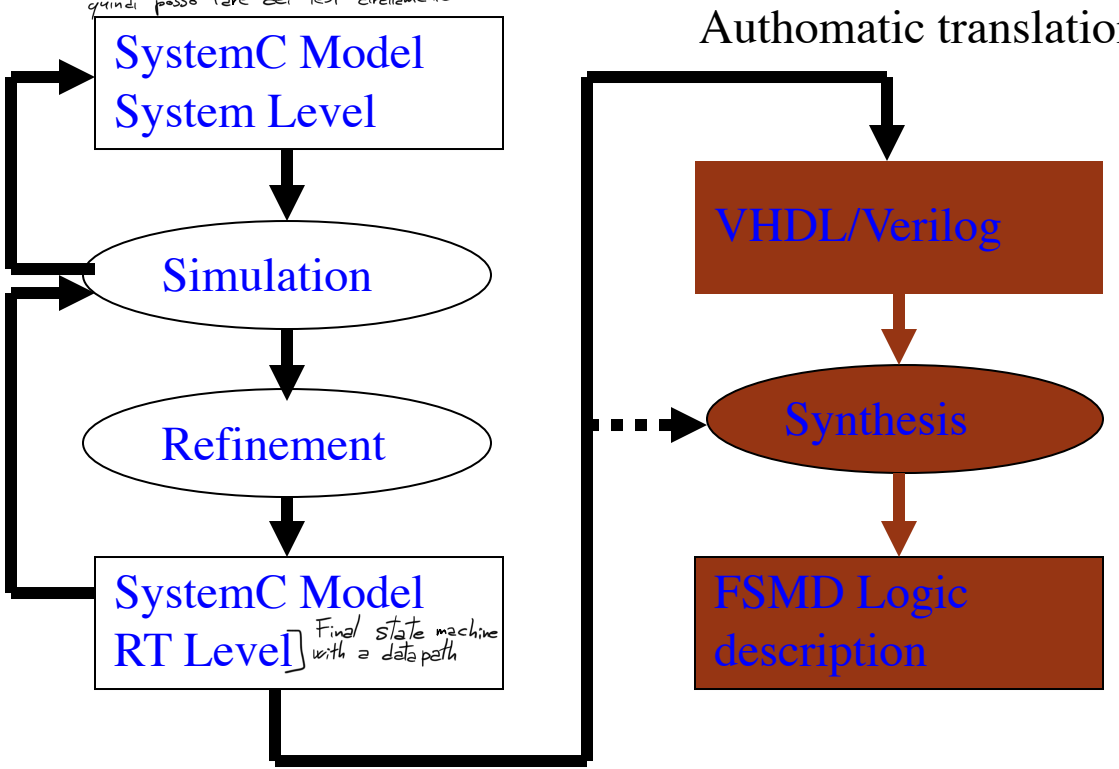
- Some C++ dialects have been proposed
- General idea:
  - new classes are defined to model hardware characteristics
  - no standardization for new classes
- SystemC 1.0 (end of 1999)
  - proposal of standardization
  - continuous extensions (2.0 ... 2.2)

Standard C-based Design Flow



SystemC-based Design Flow

In SystemC posso descrivere sia hw che sw quindi posso fare dei test direttamente



A livello RT la traduzione è diretta ma è molto a basso livello, quindi c'è un tool di sintesi che trasforma un algoritmo ad alto livello in RT

A Unifying Systems Language