

Hardware/Software Codesign

Why, What, and How?

Per Gunnar Kjeldsberg
Department of Electronics and Telecommunications
Norwegian University of Science and Technology

EMAIL: Per.Gunnar.Kjeldsberg@iet.ntnu.no
WWW: <http://www.iet.ntnu.no/groups/krets/people/kjeldsberg.html>

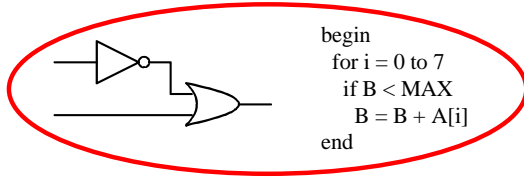
TFE02 HW/SW Codesign med innvevde systemer
August 27, 2008



- The title more or less defines the outline of my talk
- Even though I will start with a little what
- then cover why
- before I return to some more details regarding what
- The main part of the talk will be focused on how hardware/software codesign is being performed, both in industry and in the research community
- Finally I will give some examples of codesign environments
- and then conclude with a short summary and some thoughts about open topics

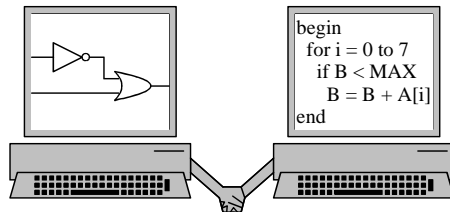
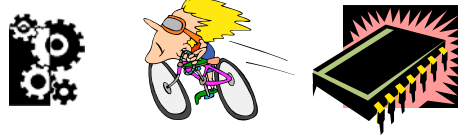
A Definition

Hardware / Software **Codesign**



means meeting
system level objectives

by exploiting the **synergism**
of hardware and software
through their
concurrent design



[De Micheli and Gupta 97]

Norwegian University of Science and Technology
Department of Electronics and Telecommunications



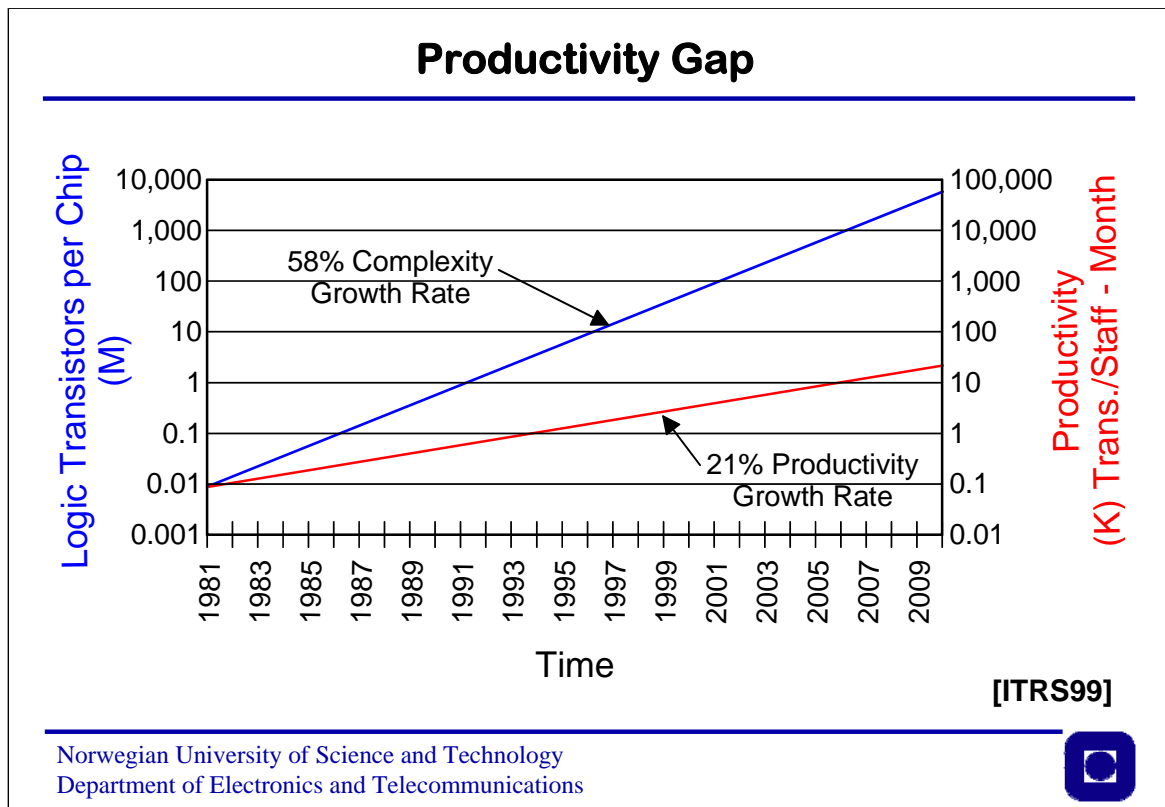
Embedded Systems



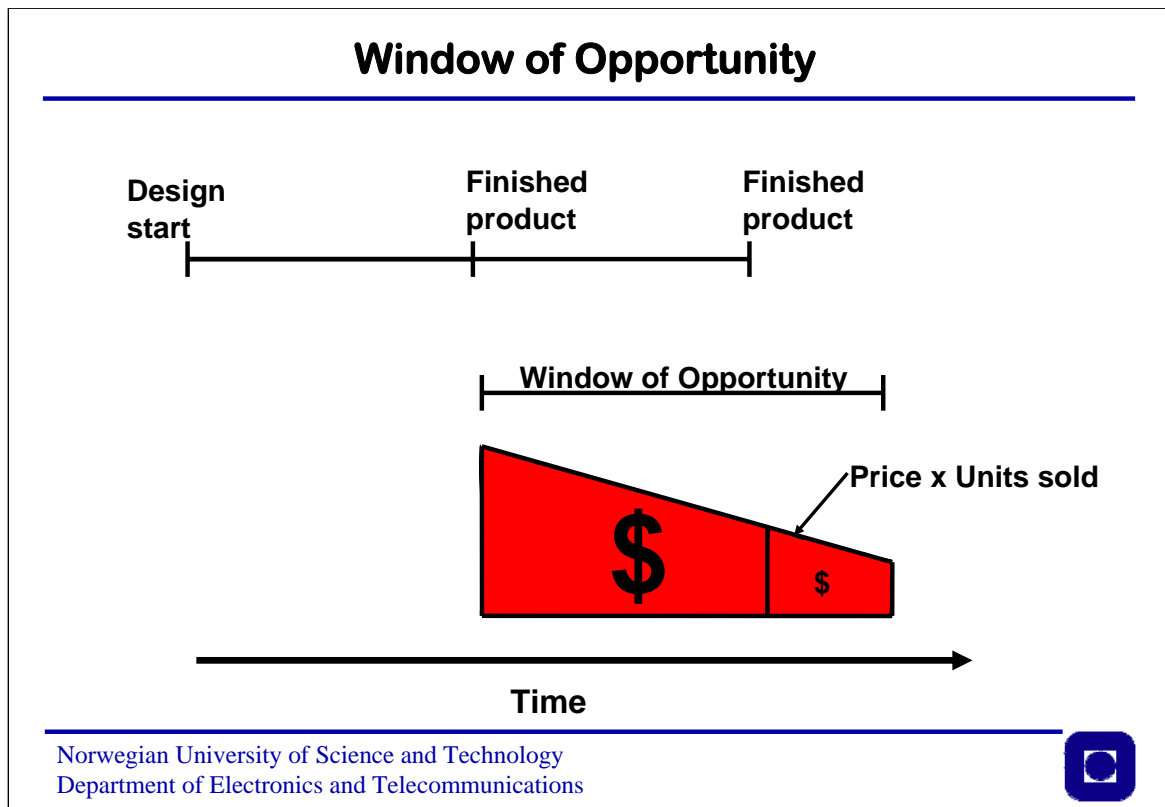
Norwegian University of Science and Technology
Department of Electronics and Telecommunications



- Codesign is mostly used during the design of so-called embedded systems.
- An embedded system is an electronic part of a bigger system, and processes data from and/or controls this bigger system.
- As such, it performs a dedicated task or a number of dedicated tasks.
- Even though it usually contains both hardware and software, the software running on an embedded system is not changed very often as it is in a general purpose computer.
- You can find embedded systems in almost every device around you. A modern car for instance, contains more than 100 microprocessors. Each is a part of an embedded system handling everything from fuel injection, the lowering of the windows, and taking continuous decision of whether to activate the airbag or not.
- Other examples are medical equipment, mobile telephones, and microwave ovens.
- A PC may also be considered an embedded system. It normally does not perform a dedicated task, however, and new Microsoft software is continuously added. I will therefore not count it as an embedded system, even though many people disagree with me.

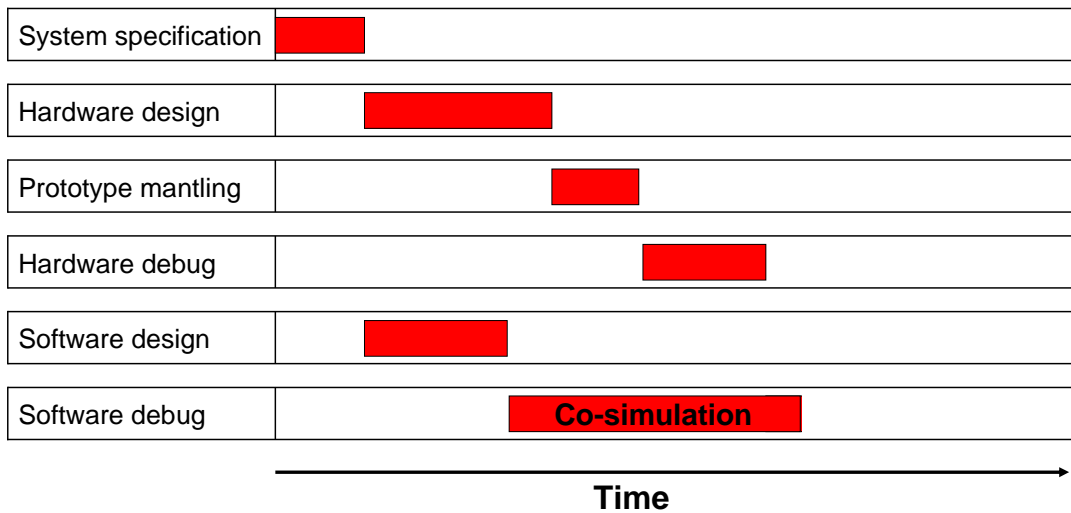


- And now comes the next question: why do we need codesign while designing embedded systems?
- International Technology roadmap for Semiconductors
- The available number of transistors per chip increases much faster than the designer's productivity
- We don't manage to fill the chips with useful functionality the way we could.
- A chip could have approximately 10,000 transistors at the beginning of the 1980
- At the same time, a designer managed to produce, or design, per month functionality requiring 100 transistors.
- A team of 10 people could hence design a chip in less than one year.
- In year 2006, with approximately 1 billion transistors on a chip, the same team, each member now designing 10,000 transistors a month, would require over 800 years to complete one chip.
- The chip technology is available. If we gave the designer better tools and methodologies, so that his or her productivity would increase, we could solve much bigger problems.
- The resulting complex designs and large design teams also induce increased error probability, which the methodology must tackle.
- Hardware/software codesign is such a methodology.



- Instead of 800 years, a company usually has very limited time available if they are going to meet the product's window of opportunity
- Assume a design start and a window as shown here
- If our design methodology is efficient enough, our product is finished in time, and we earn a lot of money.
- If not, someone else will already have flooded the market with this kind of products, and we earn next to nothing.
- As we will see later, codesign can dramatically reduce the design time, both through parallel development of hardware and software and by specifying an efficient overall design methodology

What is Codesign?

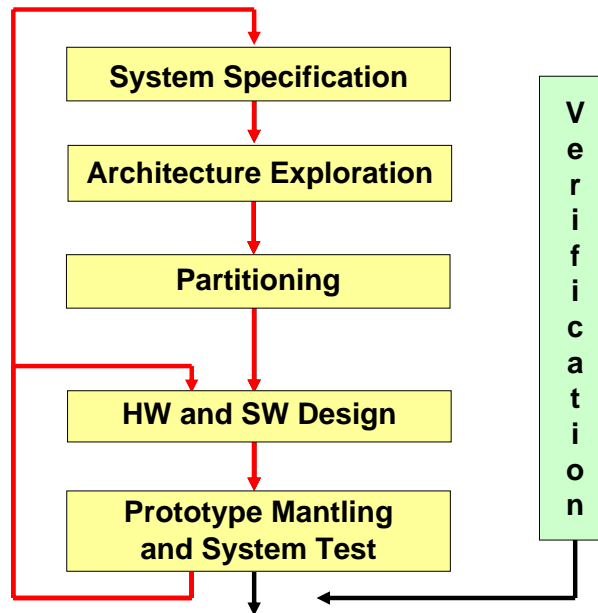


Norwegian University of Science and Technology
Department of Electronics and Telecommunications



- A typical time schedule for traditional design
- A system is specified including partitioning between HW and SW
- Two separate teams start working on hardware and software design respectively
- After some time the software people need a working hardware prototype to continue
- They have to wait for the hw people to finish designing, building the prototype and make it work
- Then they can finish their design and start debugging
- This time schedule can already be dramatically improved by what is available in mature commercial tools
- They perform co-simulation
- The hardware is modeled on the computer before it is implemented as a prototype
- The software can be run on this hardware model
- The overall design time is reduced
- Loops in the hardware design is avoided
- Better solution since bugs can be corrected in HW instead of in SW.

What is Codesign?

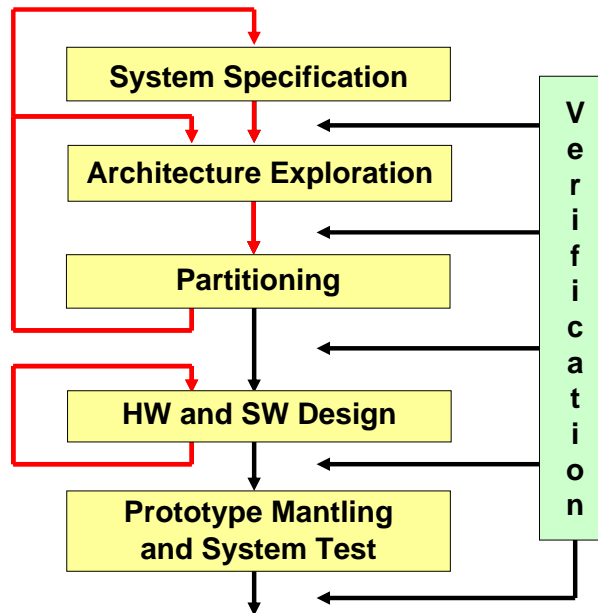


Norwegian University of Science and Technology
Department of Electronics and Telecommunications



- This is more or less the typical design steps used by designers traditionally
- at least for designs being performed top down, which I will focus on
- Except that architecture exploration and partitioning is often done in rather ad-hoc ways based on previous experience and without any real exploration
- They may therefore be included in the system specification step
- Real verification and debug is only possible when a running prototype is available
- This gives rise to very expensive and time consuming loops in the design trajectory
- Problems that are only discovered here may need a full redesign to fix
- At least a number of loops like this is typically necessary before the software and hardware runs smoothly together

What is Codesign?

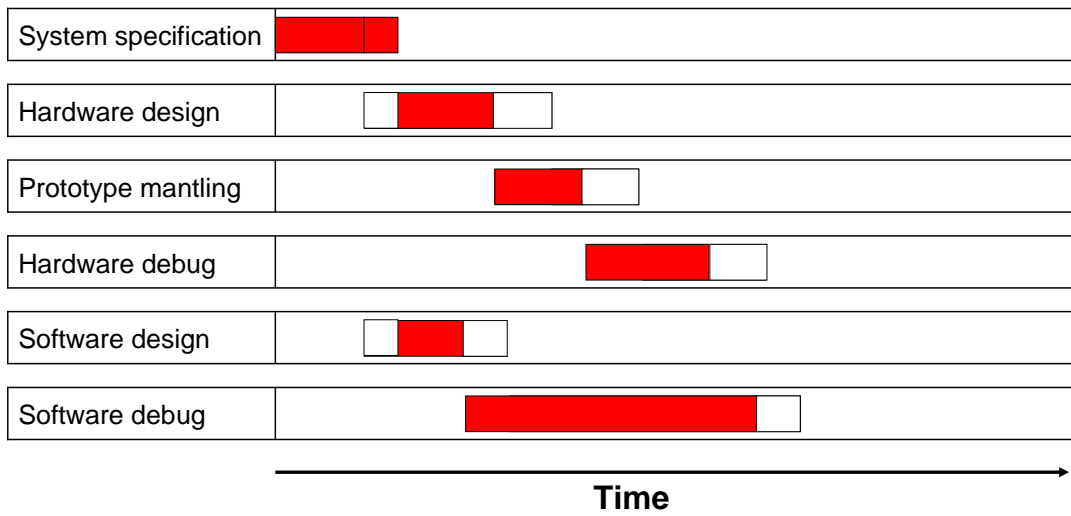


Norwegian University of Science and Technology
Department of Electronics and Telecommunications



- Now let's compare this with a true codesign methodology
- New arrows on the verification box
- Executable specification
- Real exploration of architecture
- Estimates gives feedback
- Architecture exploration is tightly connected to partitioning
- The main design loop is therefore now between these steps
- Sometimes also necessary to change specification (all the way up to the top)
- Much easier to correct errors at this high level of abstraction
- The next major loop is now only around HW and SW design
- Co-simulation avoids building of prototype

What is Codesign?

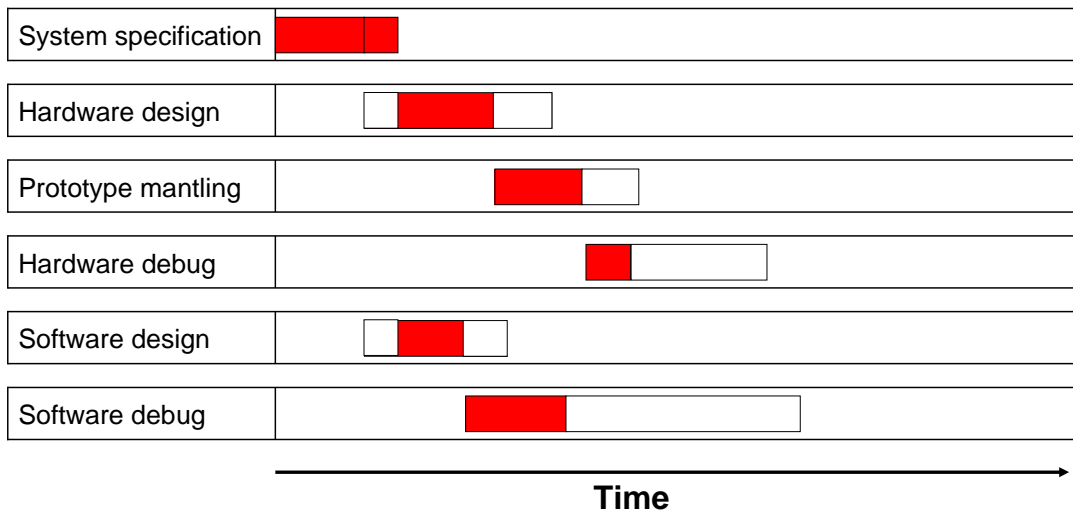


Norwegian University of Science and Technology
Department of Electronics and Telecommunications



- Back to previous time schedule
- System design increased due to exploration of architecture and partitioning
- Hardware and software design time reduced by automatic synthesis tools

What is Codesign?

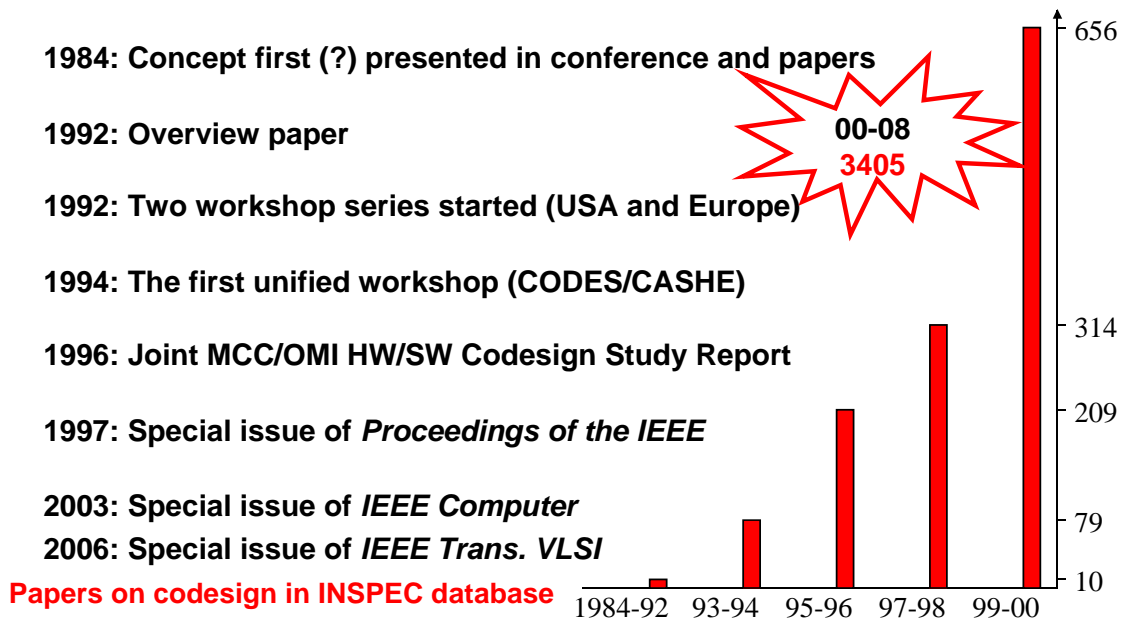


Norwegian University of Science and Technology
Department of Electronics and Telecommunications



- Verification strategy gives reduced need for hardware and software debug

History

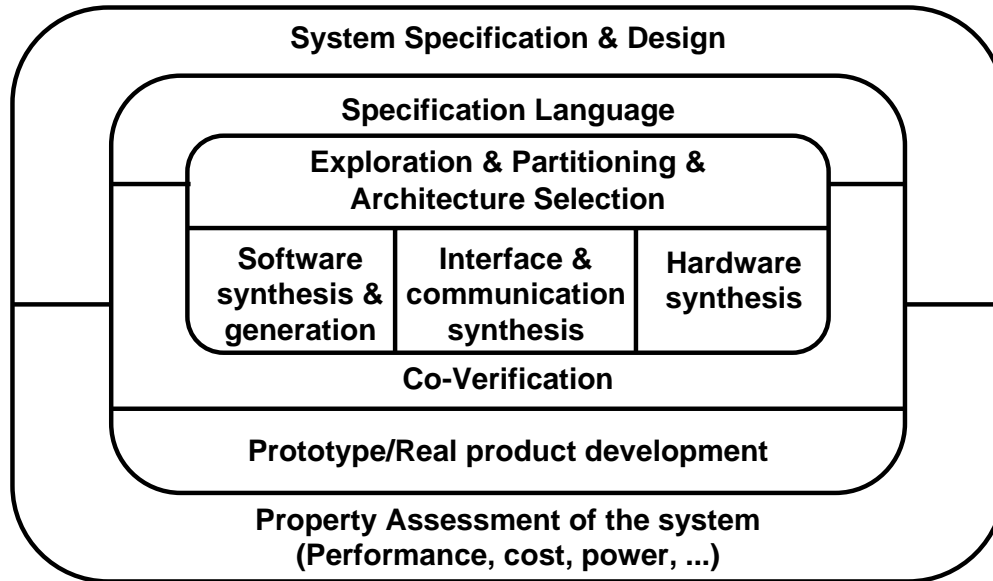


Norwegian University of Science and Technology
Department of Electronics and Telecommunications



- Covered why and what.
- Let us have a closer look at how this is done, both in the research community and in commercial tools
- I would first like to give a brief presentation of the history of codesign however
- Heterosystems at University of Utah
- Design for Tactical Avionics Maintainability (AGARD conference)
- The Architecture Design and Assessment System (ADAS)
- Research Triangle Institute, North Carolina, USA
- Second half of 1980s, mainly further development of ADAS, but also some other systems, the SARA system for instance
- In 1992, the first overview paper was published, indicating a somewhat maturing field
- The same year, two series of workshops on codesign was started, one in Europe, and one in USA.
- In 1994 they were merged into the annual International Workshop on hardware/software codesign.
- In 1996 the Microelectronics and computer Technology Corp. in Austin, Texas, and the Open Microprocessor Initiative in Brussels, Belgium, sponsored a codesign study with participating companies and universities from North America and Europe
- It gave a detailed presentation of the then current state of the art in research, the state of the practice in a number of companies, and the state of the market by key EDA vendors.
- It showed a maturing field, but with still much room for further development.
- Major scientific journals focused a complete issue on the topic in 1997, 2003, and 2006
- A quick look at the numbers of papers registered in the INSPEC publication database, also shows increasing interest.

MCC/OMI Research Focus Model



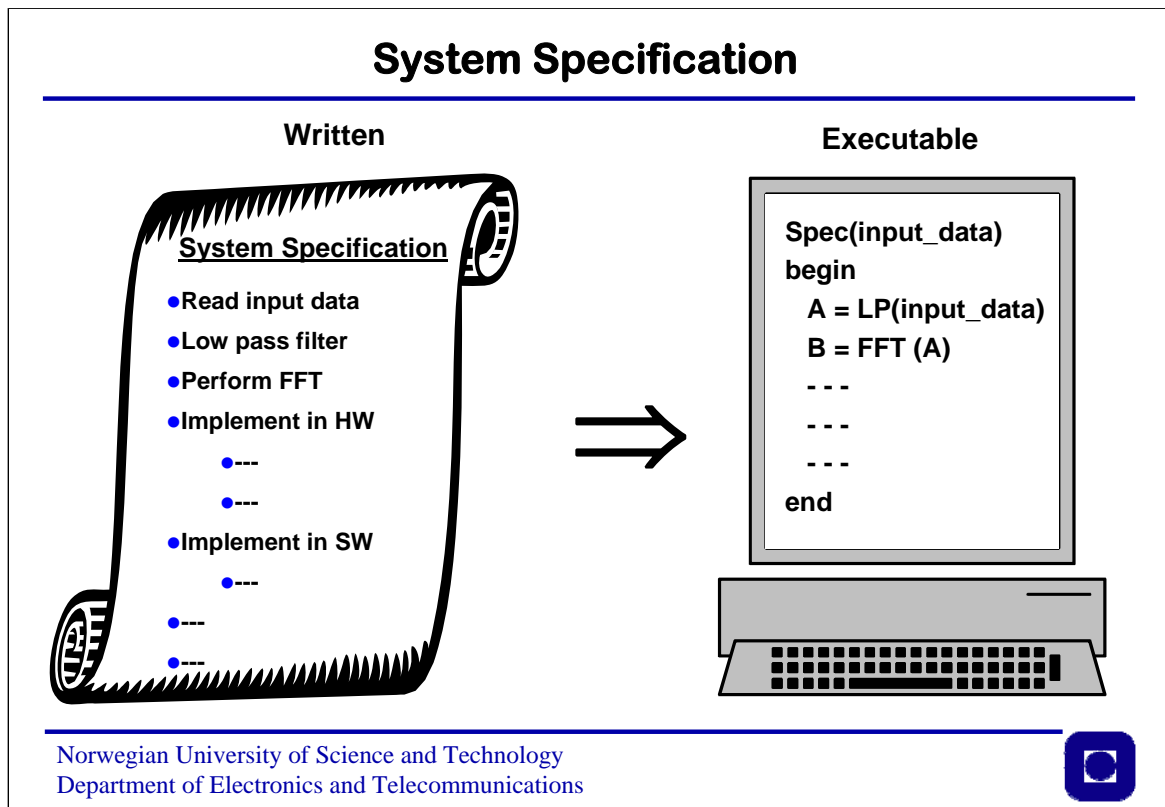
Norwegian University of Science and Technology
Department of Electronics and Telecommunications



- Describes the important tasks in codesign and how they interact.
- Each system development divided into three phases
 - system specification
 - product development
 - property assessment
- Product development further divided into
 - selection or development of specification language
 - architecture design
 - co-verification of design on different levels of abstraction
- Architecture design further divided into
 - exploration and architecture selection
 - component design
- Finally, the design of the components consist of
 - software synthesis
 - communication and interface design
 - hardware synthesis

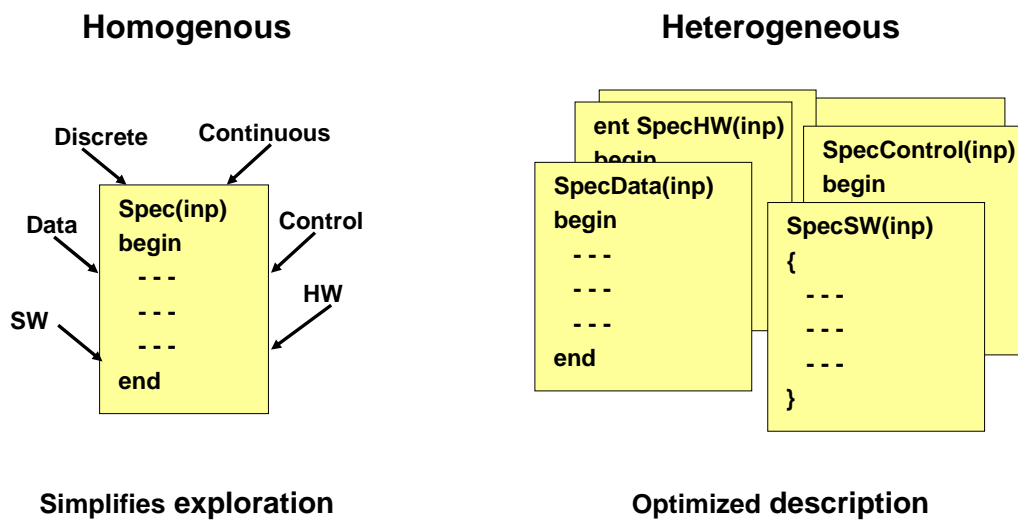
- We will now go into some of these topics in somewhat more detail
- Due to the tight timing constraint on this talk, I will not go into topics like virtual component IP reuse or design for testability.
- These are important parts of a codesign flow, but also somewhat orthogonal to it.

System Specification



- We will now start at the top and work our way downwards in the design trajectory
- In traditional design, a system specification is a document written in natural language, typically English
- This includes quite a lot of implementation details, such as the overall system architecture, hw/sw partitioning, and timing constraints for different parts of the design.
- Using a true codesign, or system design methodology, an executable specification is needed very early in the design trajectory.
- Executable in this context means that it is specified in such a way that it is possible to run it on a computer
- It should only contain information regarding what the system should do, not implementation details such as architecture, hw/sw partitioning etc.

System Specification



Norwegian University of Science and Technology
Department of Electronics and Telecommunications



- Several approaches
- homogenous: everything described using one language
- heterogeneous: optimized languages for each domain
- homogenous:
 - easier to explore design space since no translations required
 - less optimized description since no language is optimal for every domain
- heterogeneous:
 - Optimized description in each domain possible
 - Requires translations to change from one domain to another
 - Easier to include legacy code
- language:
 - homogenous, possible to develop new optimized language. Might be difficult to get accepted
 - heterogeneous: uses usually traditional languages like C, Java, SDL, VHDL. Easier to get accepted and compilers and tools exist.

Modeling at Different Levels of Abstraction

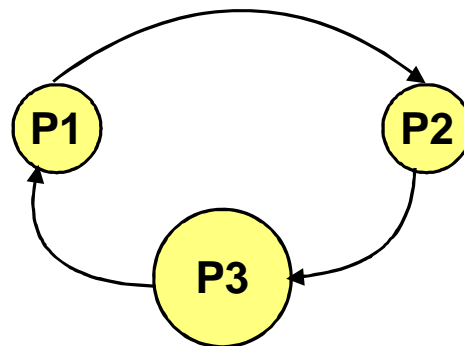
System Level: Network of Communicating Processes

Models:

- Hierarchy
- Concurrency
- Communication
- Synchronization

Used for:

- Architecture exploration
- Partitioning
- Communication synthesis



Norwegian University of Science and Technology
Department of Electronics and Telecommunications



- It is necessary to be able to model the system at different levels of abstraction
- At each level the model is used while exploring different aspects of the design space for a solution that fulfills the design requirements.
- Some typical examples at different levels of abstraction

Modeling at Different Levels of Abstraction

Algorithmic Level: Control and Data-flow Graph

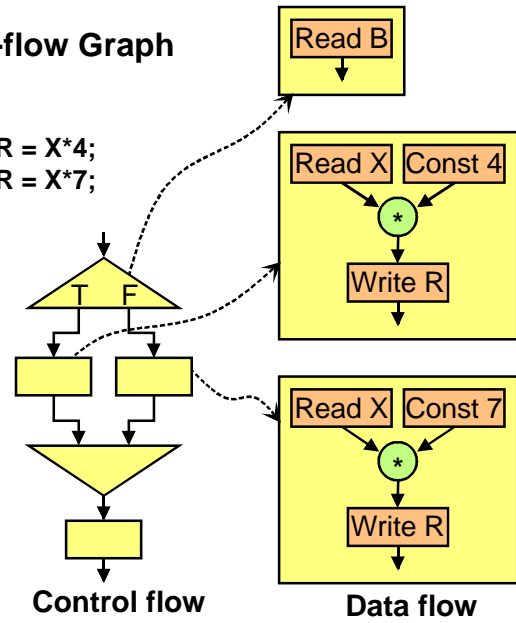
Models:

- Complex operations
- Shared operators
- Sequential operation

Used for:

- Scheduling
- Allocation
- Generation of
 - Control Unit
 - Data Path

case B is
when T \Rightarrow R = X*4;
when F \Rightarrow R = X*7;
end case;



Norwegian University of Science and Technology
Department of Electronics and Telecommunications



Modeling at Different Levels of Abstraction

Register Transfer Level: State machines and Boolean equations

Models:

- Clock cycle accurate behavior

$$A = B + C$$

$$D = !A \& E$$

Used for:

- Synthesis of gates and registers

Other models:

- Petri net
- Hierarchical Concurrent Finite State Machine with Datapath (HCFSMD)
- Instruction level and clock cycle accurate software models

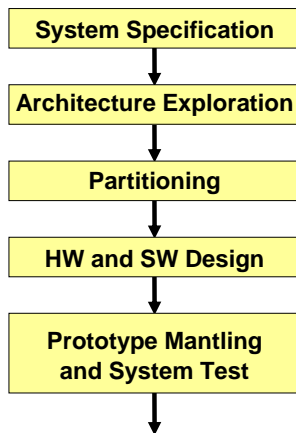
Norwegian University of Science and Technology
Department of Electronics and Telecommunications



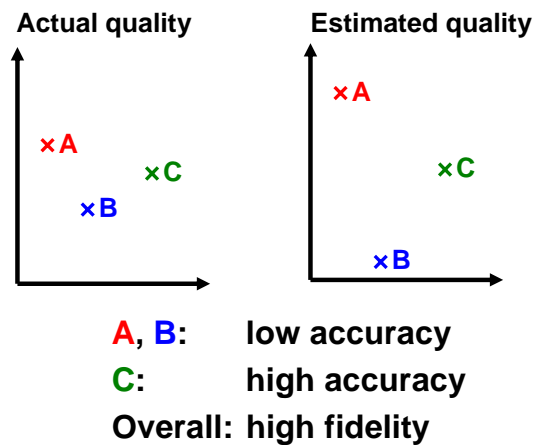
Estimation

Essential during exploration for evaluation of alternative solutions

Speed vs. Accuracy



Accuracy vs. Fidelity



Norwegian University of Science and Technology
Department of Electronics and Telecommunications



- Estimation is essential...
- impossible to implement each alternative in detail
- accuracy enough to make correct decisions
- fast enough to evaluate many solutions
- Architecture exploration often based on a library of components for which we know chip size, performance, and power consumption
- For comparison fidelity is usually more important than accuracy
- An estimate has high fidelity if it orders the quality of the solutions correctly
- A, B, and C is the actual quality (for instance power consumption) of three alternative solutions
- a given estimation technique gives the following result ...
- A and B has low accuracy since it is big differences between actual and estimated quality
- C has high accuracy
- The overall estimate has high fidelity, since the quality of A, B, and C is ordered correctly. A highest, C, in the middle and B lowest
- Still much research needed to find good estimation techniques at the higher levels of abstraction

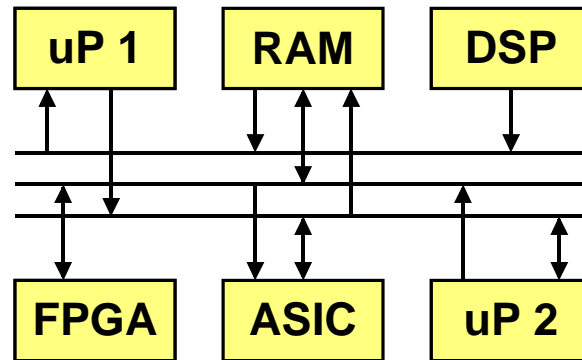
Architecture Exploration

Allocate necessary

- uPs
- “ASICs”
- Memory
- Buses

to perform system tasks

Estimate solution quality



Norwegian University of Science and Technology
Department of Electronics and Telecommunications



- System architecture: allocation of the number of processors, ASICs, DSPs, memories and buses needed to perform the system tasks
- Note the use of quotation marks around ASIC. Everything may be included on chip, so ASIC is really a specially designed and fixed hardware module
- Automatic tools are not available
- Tools for interactive exploration exists
- A|RT from Frontier Design and ArchiMate from Arexsys

Partitioning

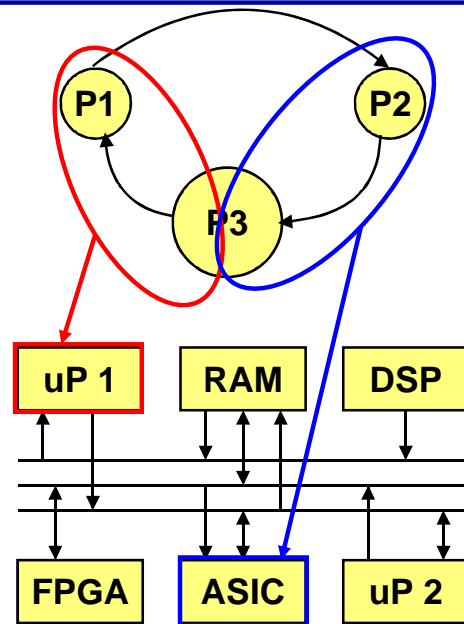
Group tightly coupled processes for implementation together

High performance requirement



Hardware

Map groups to architecture modules

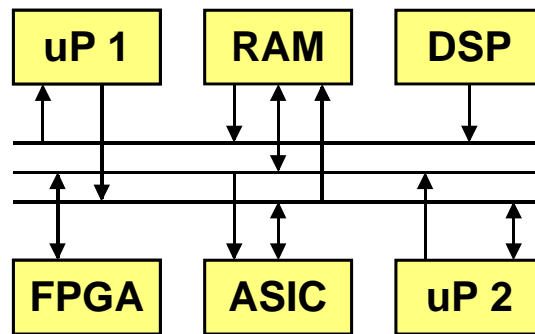


Norwegian University of Science and Technology
Department of Electronics and Telecommunications



- Starting with the system level process graph
- group tightly coupled processes for implementation in the same module
- If P1 and this part of P3 interchange much data, they should be implemented in the same module to avoid overloading the bus
- Modules with high performance requirements should be implemented in hardware
- Less performance critical processes or processes requiring large hardware units should be implemented in software
- Finally the grouped processes must be mapped to architecture modules
- Often iteration with architecture allocation

HW, SW, and Interface Synthesis

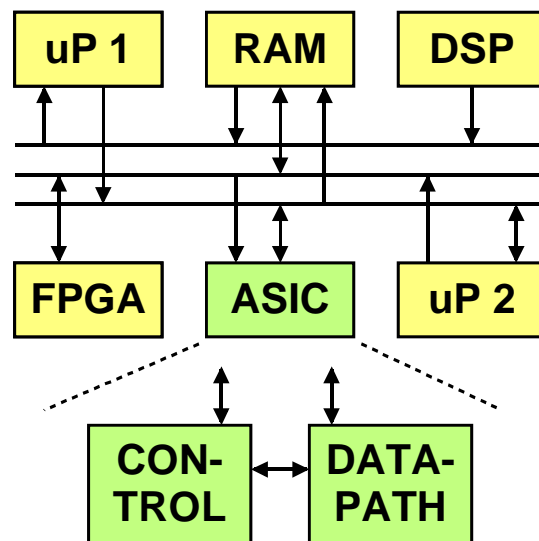


Norwegian University of Science and Technology
Department of Electronics and Telecommunications



- After architecture exploration, partitioning, and mapping, each module and the interface between them needs to be generated
- A stepwise refinement, adding the amount of implementation detail
- manual or interactive at the higher levels, automatic at lower levels

HW, SW, and Interface Synthesis

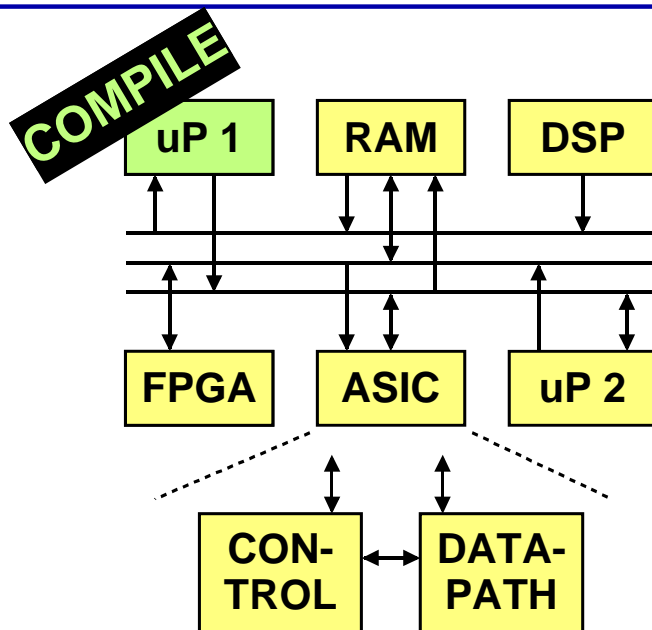


Norwegian University of Science and Technology
Department of Electronics and Telecommunications



- Using a control data-flow graph, a control and data-path can be generated for each hardware module
- These are refined further down to register transfer level,
- from which automatic synthesis tools exist

HW, SW, and Interface Synthesis

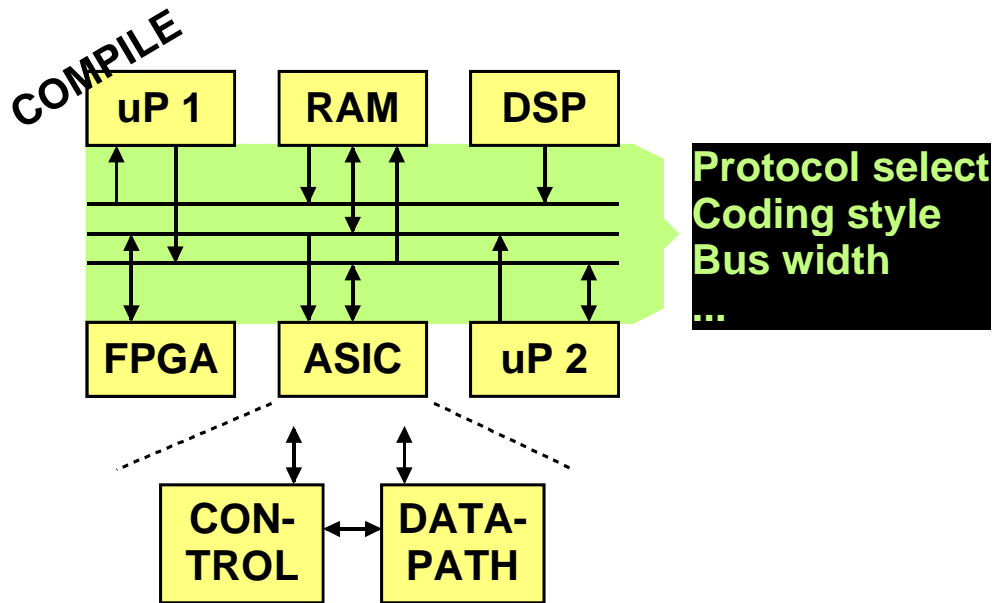


Norwegian University of Science and Technology
Department of Electronics and Telecommunications



- The software side is somewhat simpler
- If standard uPs are used, they usually come with compilers
- In many cases the input code to the compilers needs to be fine tuned to achieve optimality, for instance regarding how data is stored in and transferred to and from memory
- sometimes customized uPs are generated to achieve optimality for a given design
- It is then also necessary to generate a compiler for the uP
- for example the PEAS projects at the Imai laboratories in Japan

HW, SW, and Interface Synthesis



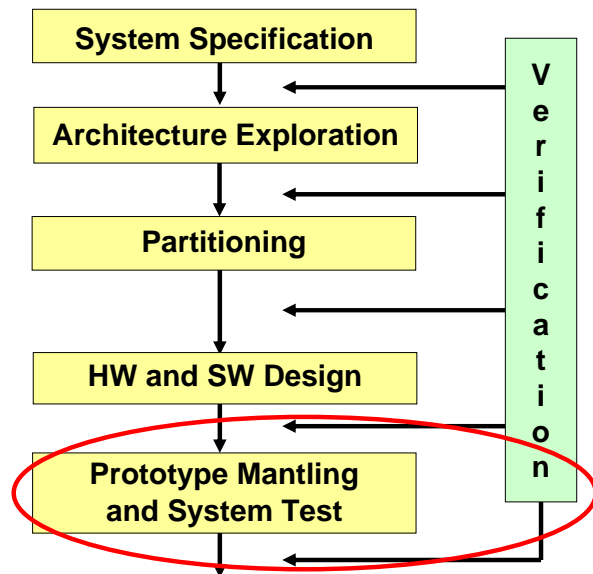
Norwegian University of Science and Technology
Department of Electronics and Telecommunications



- An important part of the architecture is the interface between the modules.
- At the system level they are represented as abstract channels
- Now the actual interfaces must be synthesized
- Bus, FIFO, multiplexers
- Protocols must be selected
- as well as data and address coding style
- bus width
- A number of commercial tools performs interactive interface synthesis based on a library of modules.
- CoWare, Archimate

Co-Verification

- Verification of correctness at each design step
- Reuse of simulation results from higher levels
- Emulators used at lowest level



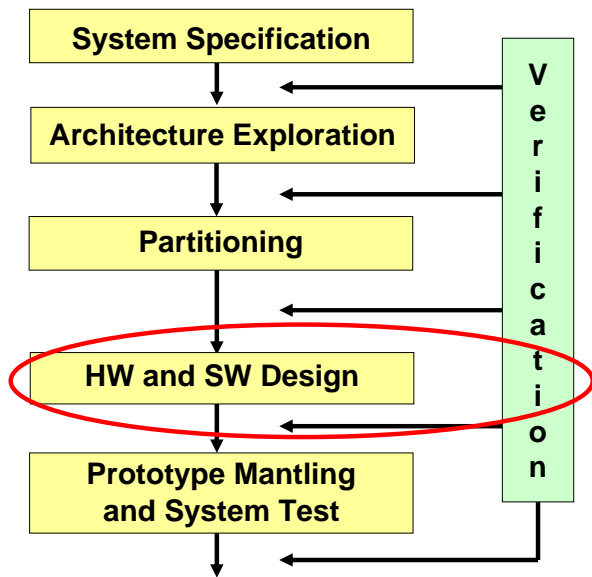
Norwegian University of Science and Technology
Department of Electronics and Telecommunications



- The importance of a consistent verification strategy can not be stressed often enough
- At each step, the correctness of the added details must be verified
- The most widely used technique is simulation
- The verification coverage is then very much decided by the stimuli
- To avoid introduction of errors in the testbench, it is important to reuse stimuli and results from higher levels for comparison at lower levels.
- As the amount of implementation detail increase, the simulation speed drops dramatically
- Several orders of magnitude slower than the real system.
- Rapid prototyping necessary
- FPGA based hw emulators and up emulators can be used to speed up verification and to avoid the production of a full scale prototype

Co-Verification

- Verification of correctness at each design step
- Reuse of simulation results from higher levels
- Emulators used at lowest level
- Mature co-simulation for designs at RT-level



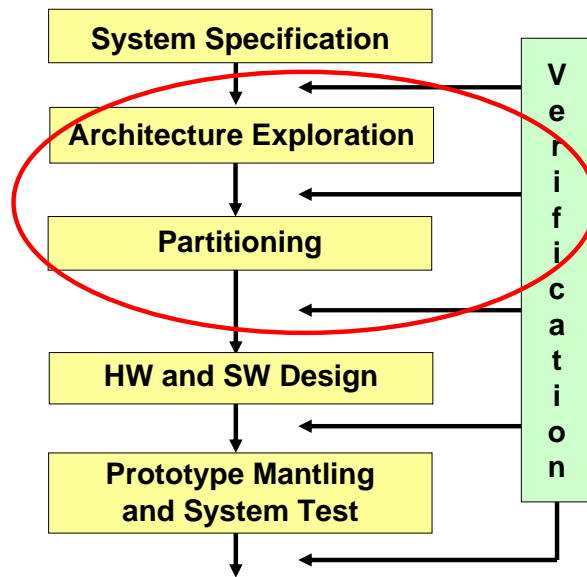
Norwegian University of Science and Technology
Department of Electronics and Telecommunications



- At the register transfer level, mature co-simulation tools are available, such as Seamless from Mentor Graphics

Co-Verification

- Verification of correctness at each design step
- Reuse of simulation results from higher levels
- Emulators used at lowest level
- Mature co-simulation for designs at RT-level
- Simulation based on library modules at higher levels



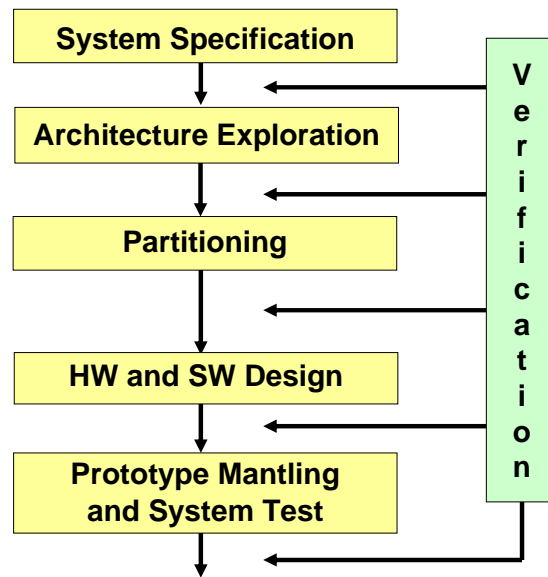
Norwegian University of Science and Technology
Department of Electronics and Telecommunications



- At the higher levels of abstraction, the typical approach is again simulation through the use of library modules,
- especially for the interface between modules
- If each module is written in a high-level language, like C++ or Java, it is possible to simulate these using off-the-self or special made debuggers or software simulators

Co-Verification

- Verification of correctness at each design step
- Reuse of simulation results from higher levels
- Emulators used at lowest level
- Mature co-simulation for designs at RT-level
- Simulation based on library modules at higher levels
- Formal verification



Norwegian University of Science and Technology
Department of Electronics and Telecommunications

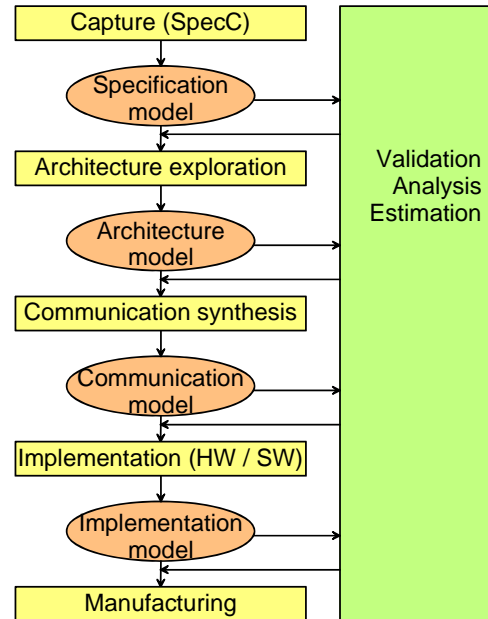


- Due to the low test coverage reachable by simulation,
- many research teams are working on techniques for formal verification.
- Using mathematical techniques they try to prove that the synthesis from one level to a lower level is correct
- Some commercial tools also exist, such as VERIFYer from Avant!
- Due to the complexity of the techniques, the tools can usually only be used for smaller or restricted types of designs
- It is a very interesting approach however, but more research is still needed.

Homogenous Design Environment

SpecC: [Gajski et al.]

- Superset of ANSI-C
 - Program State Machine model
 - Behavioral and structural hierarchy
 - Time
 - Concurrency
 - +++
- Exploration
- Verification
- Code generation (C, VHDL)
- Others:
 - CoCentric, A|RT, JavaTime
 - SystemC



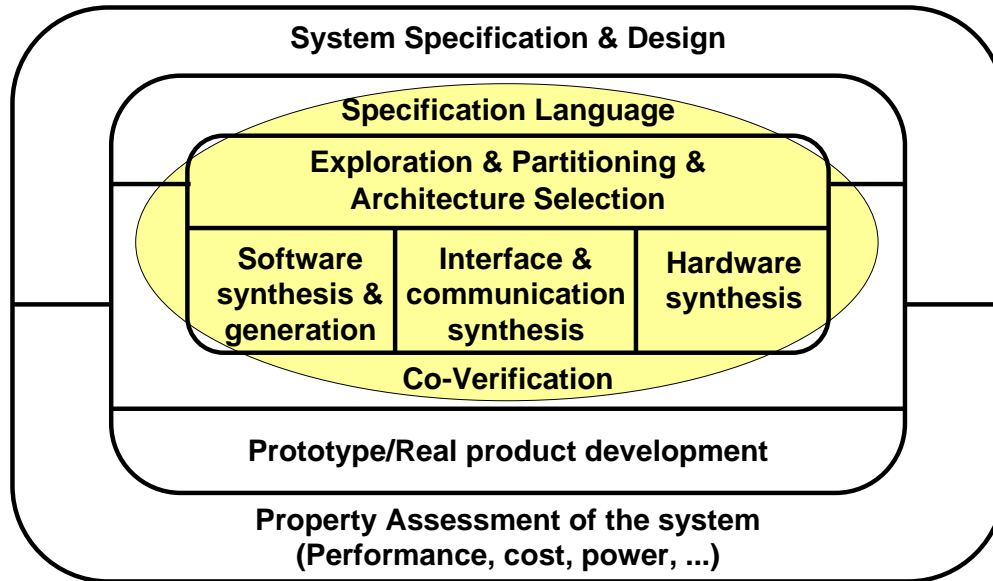
Norwegian University of Science and Technology
Department of Electronics and Telecommunications



•SpecC

- Captures the system using a new language, SpecC
- Uses a Program State Machine model of computation
- A variant of an Hierarchical Concurrent Finite State Machine with Datapath
- Properties such as ...
- The specification model is validated, analyzed and estimated using simulation
- Input to architecture exploration with allocation and mapping
- Includes HW/SW partitioning
- Simulation to
 - evaluate satisfaction of design constraints
 - validate correctness compared to specification model
- Communication synthesis between modules
 - protocol selection and synthesis
- The communication model is input to standard CAD tools for hardware and software implementation (C and VHDL)
- Other tools are JavaTime from Berkley, and commercial ones like CoCentric from Synopsys and A|RT from Frontier Design
- SystemC is the language that is used today in these approaches.

SpecC

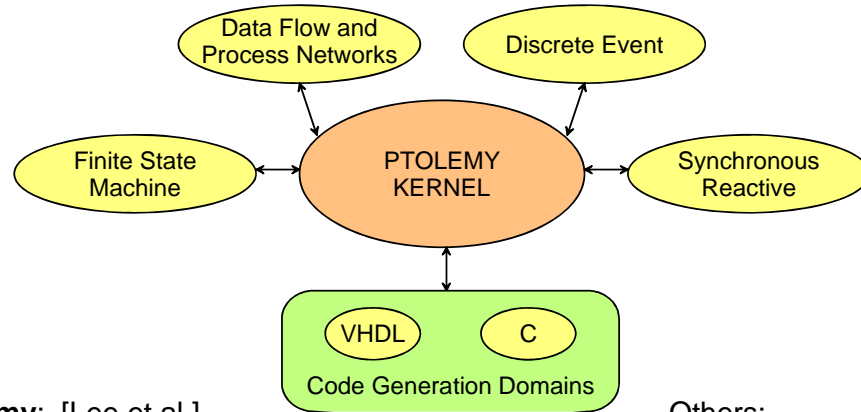


Norwegian University of Science and Technology
Department of Electronics and Telecommunications



- SpecC aim to cover the full design trajectory
- Due to the homogeneous specification it is possible to perform exploration

Heterogeneous Design Environment



Ptolemy: [Lee et al.]

- Multiple models of computation domains
- Kernel connects system blocks of different domains
- Simulation and code generation

Others:

- CoWare, VCC, Seamless, CosiMate/(ArchiMate)

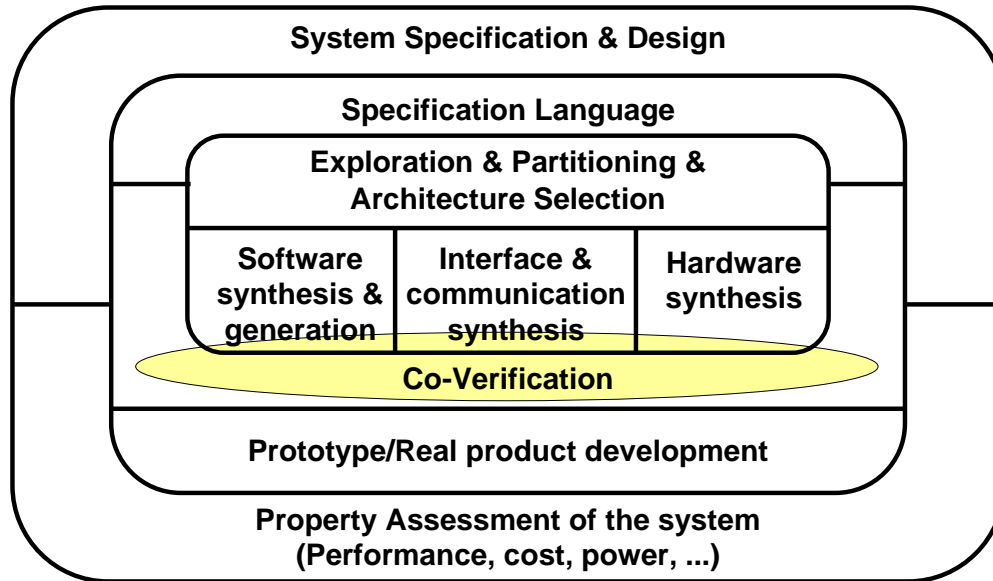
Norwegian University of Science and Technology
Department of Electronics and Telecommunications



- Ptolemy from Berkeley extremely heterogeneous
- System defined using a number of languages and models of computation
- C++, C, VHDL, Java, Matlab, etc. can be used to specify system blocks in various model of computation domains such as ...
- The Ptolemy kernel
 - does not define any model of computation
 - consists of a number of C++ classes enabling definition of model of computation domains
 - defining ways for blocks to connect and interchange information.
- The lack of common model of computation hamper design exploration
- Mainly a simulation environment
- In addition it facilitates generation of C and VHDL code

- Other heterogeneous tools are CoWare NsC from CoWare, Virtual Component Codesign from Cadence, Seamless from Mentor Graphics and CosiMate from Arexsys.
- ArchiMate, the architecture exploration tool from Arexsys is in a way homogenous

Ptolemy



Norwegian University of Science and Technology
Department of Electronics and Telecommunications

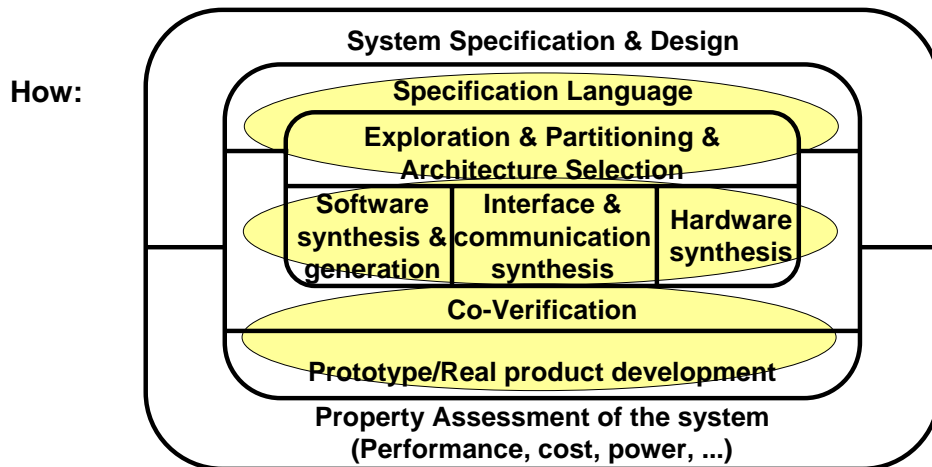


- Ptolemy is somewhat more restricted in its objective than SpecC.
- It is more a co-verification tool
- and the heterogeneous specification makes exploration difficult

Conclusion

Why: Codesign necessary to bridge the productivity gap

What: Exploit synergism of hardware and software through concurrent design



Norwegian University of Science and Technology
Department of Electronics and Telecommunications



- why
- what
- how:

- The MCC/OMI research focus model nicely pictures the different elements of a codesign methodology
- Currently there are mature techniques and tools at rapid prototyping, co-simulation, and lower levels of software, hardware and interface synthesis
- for exploration, partitioning and architecture selection, interactive tools are becoming available
- synthesis is available for specialized product types, such as digital signal processing
- More research is needed to enable automation of this step, especially within estimation techniques to guide the exploration
- Much activity in the research community is currently centered around specification languages.