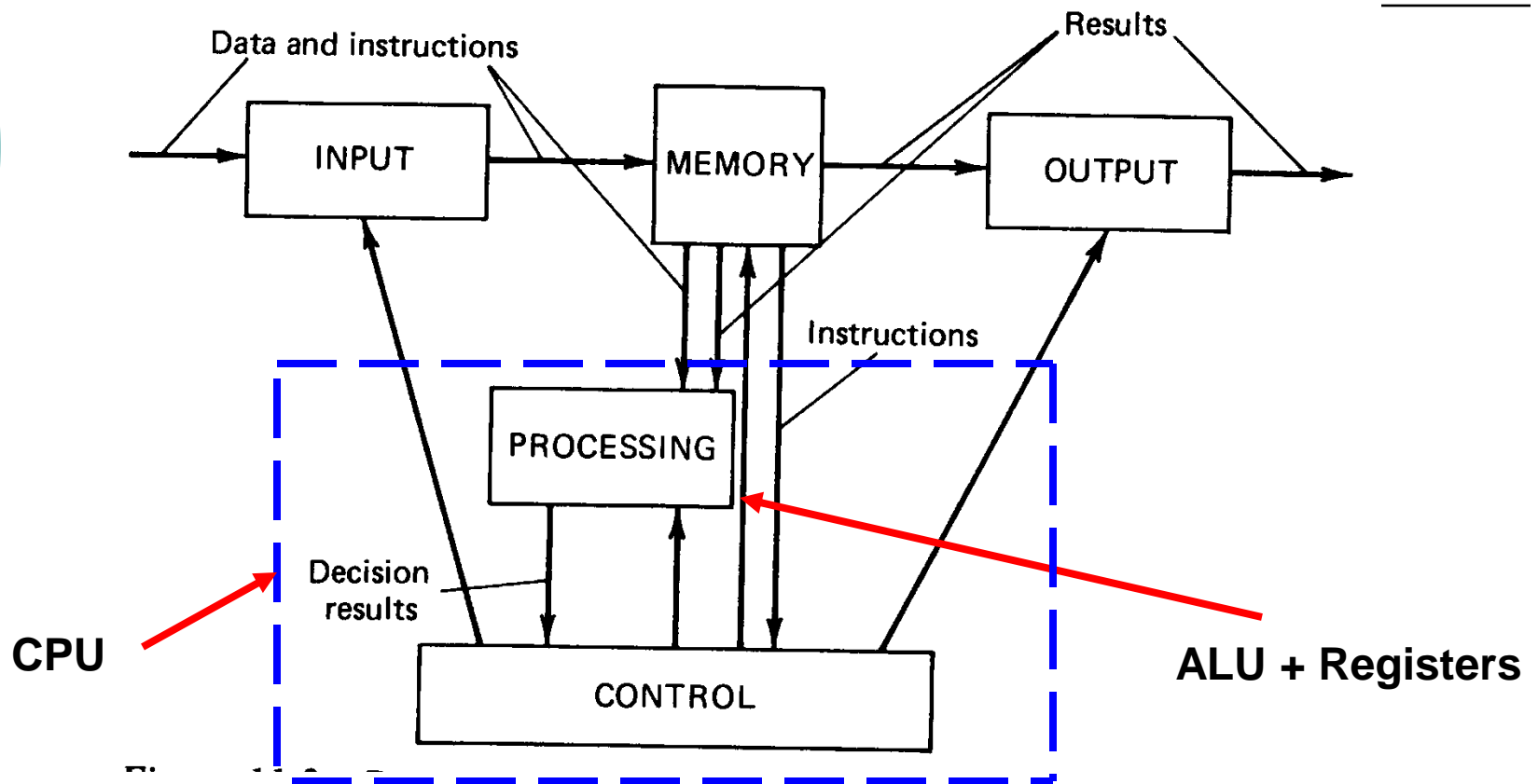

Real Time Embedded Systems

Hamid Vakilzadian
Department of Electrical Engineering
University of Nebraska-Lincoln

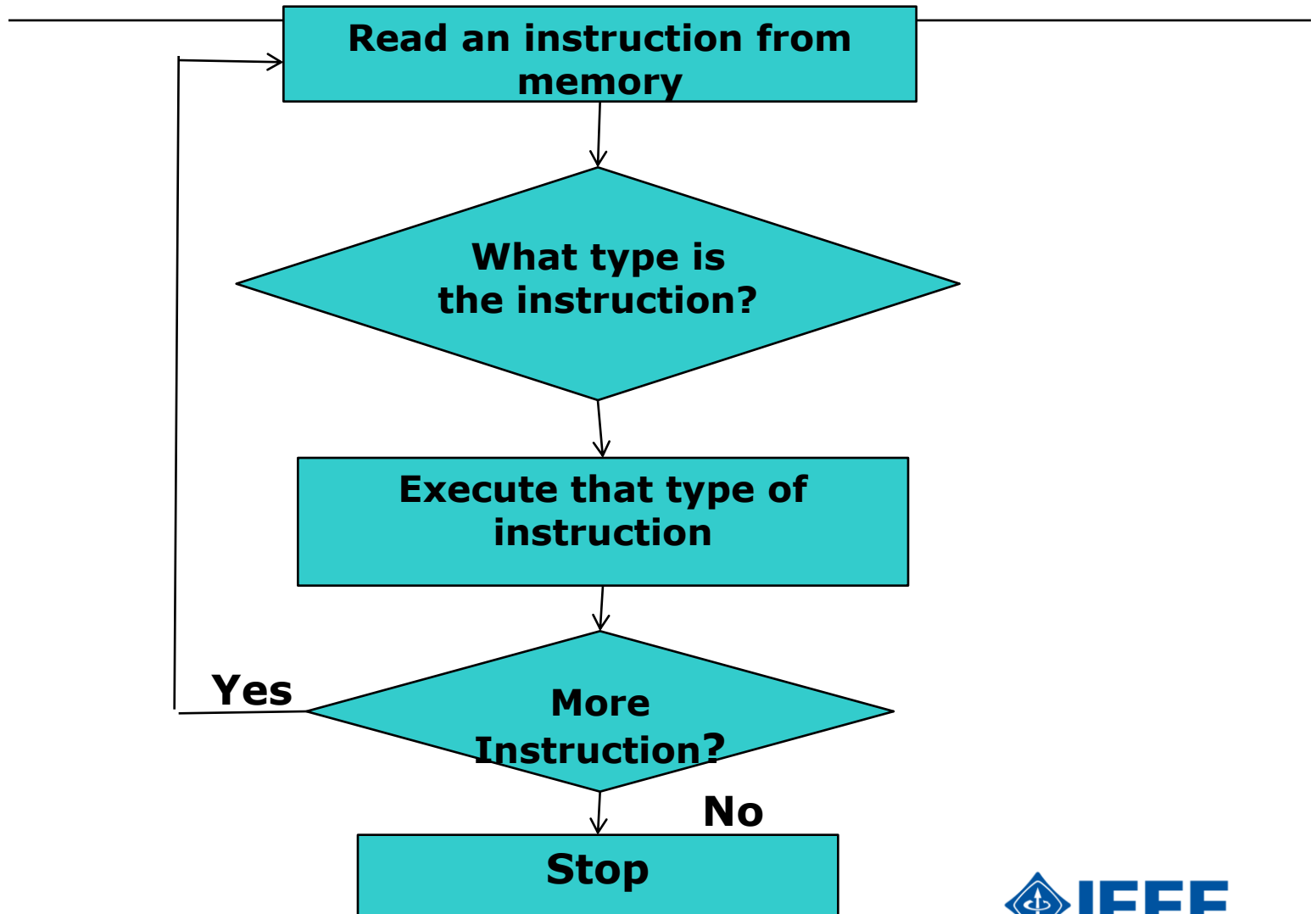
2010 IEEE Annual Meeting
Nashville, TN

Structure of a Computer



Von Neumann Architecture

How Does an Instruction Get Executed?



What is a Real-Time System?

- A system that its correctness depends not only on the logical results, but also on the **time** in which the results are produced.
- works in a time-constrained environment.
- Can be very simple such as a microcontroller or a microprocessor controlling an automobile engine or a highly sophisticated, complex, and distributed system such as air traffic control system.

What is an Embedded System?

- An integration of hardware and software engineering to create a dedicated system that performs specific and pre-defined tasks.
- A way to convert your idea into marketable product => A way to start your business!
- Huge market for them.
 - Example
 - Pacemaker & Defibrillator
 - Smart card reader
 - ATM Machines

Embedded Systems: Examples

- Weather/GPS satellite/Navigation Systems
- Banking and transaction processing applications
- Automobile engine control units
- Signal processing applications
- Home appliances (microwave ovens)
- Industrial controllers in factories
- Cellular communications

Characteristics of Embedded Systems

- Include processors dedicated to specific a function/application
- React to external input(s)
- Meet real-time constraints
 - Performance
 - Reliability
 - Form factor (acceptable weight, size, power consumption requirements)

Embedded Systems: Specific Trends

- Use 8, 16, 32-bit microprocessors ($\sim 30 - 100$ MHZ)
- Contain limited amount of memory ($\sim 10-64$ MB)
- Must satisfy strict real-time and/or performance constraints
- Must optimize additional design objectives:
 - Cost
 - Reliability
 - Design time
- Increased use of hardware/software codesign principles to meet constraints

ES Hardware Platform

- Choosing a proper hardware platform can optimize the design performance and reduce the design time and cost.
- Most common ES hardware platform:
 - Microcontroller / Microprocessor based system
 - Programmable Logic based system
 - Hybrid system

ES Hardware Platform

Microcomputer/Microprocessor System

- Main Advantages:
 - Optimized for software programming.
 - A huge variety of uC's that cater to both general and application-specific purpose.
 - Include subset of ANSI-C IDE and RTOS.
 - Standard control-datapath IC structure for easy assembly programming.
 - Numerous manufacturers, large number of alternative selections.
 - Lower Cost (?)

ES Hardware Platform

Microcomputer / Microprocessor System

Disadvantages:

- Lack of hardware design flexibility.
- Limited number of on-chip modules (ex: ADC/DAC).
- May require a lot of additional ICs to establish a complete hardware platform.
- Degraded performance with lengthy lookup table operations.
- General purpose, not ideal for highly customized ES design.

ES Hardware Platform

Programmable Logic System (FPGA, CPLD, ASIC)

○ Main Advantages:

- Great flexibility for customizing hardware design.
- On-chip modules can be hardware customized on ASIC or software customized with IP cores on FPGA and CPLD.
- Superb performance with lookup table and combinatorial logic operations with CPLD.
- Ideal for parallel pipelining operations (ex. Image processing) with FPGA.
- Also carries on-chip microcontroller with customized ANSI-C IDE and RTOS

ES Hardware Platform

Programmable Logic System

- Disadvantages:
 - Require HDL knowledge.
 - Longer development time.
 - Less manufacturers are providing programmable logic ICs in the market (compared to that of uC). Less alternative selections.
 - Hardware design debugging are much more challenging than that for software design.
 - More costly (?).

ES Hardware Platform: Hybrid system

- Multiple uC's
- uC + CPLD
- On-chip multicore system with FPGA
- Mix-and-Match

RT Embedded System Design Flow

- How to start your business (convert you idea into a marketable product)?
- Clarify your idea and identify its important features.
- Select a suitable hardware platform.
- If available, purchase the dev. kit along with the IDE and possibly the customized OS/RTOS and IP Cores.
- Partition your design into hardware/software components and do firmware Programming.

RT Embedded System Design Flow

- Identify the hardware components being used on the dev. kit, and determine their specs and available alternatives to optimize your design cost.
- Construct your PCB design. You can often find the recommended PCB designs for the ICs in their spec sheets.
- Prototyping – Optional, can be very costly!
- Revise and finalize your design.
- Time for mass production!
- Casing / Packaging / Advertisement

Real Time ES Computing

- Software & hardware for a system has real-time constraints and are interrupted often.
- Comprises of:
 - ✓ Synchronous Programming Language.
 - ✓ Real-time Operating System.
 - ✓ Real-time Design Methodologies.

Synchronous Programming Language

- Support for the management of time
- Provide abstract modules for scheduling algorithms, parallelism, pre-emption.
- Provide predictable execution time, synchronous dataflow programming, etc.
- Examples: Lustre, Quartz, SOL, Esterel.

Real-Time Operating System (RTOS)

- Intended for real-time applications
- Able to meet deadlines required by the application.
- Standards:
 - Round-robin scheduling
 - Preemptive scheduling
 - Pre-emptive fixed-priority scheduling
- Examples of usage: Appliance Controller, Spacecraft, Robots, Industrial Control.

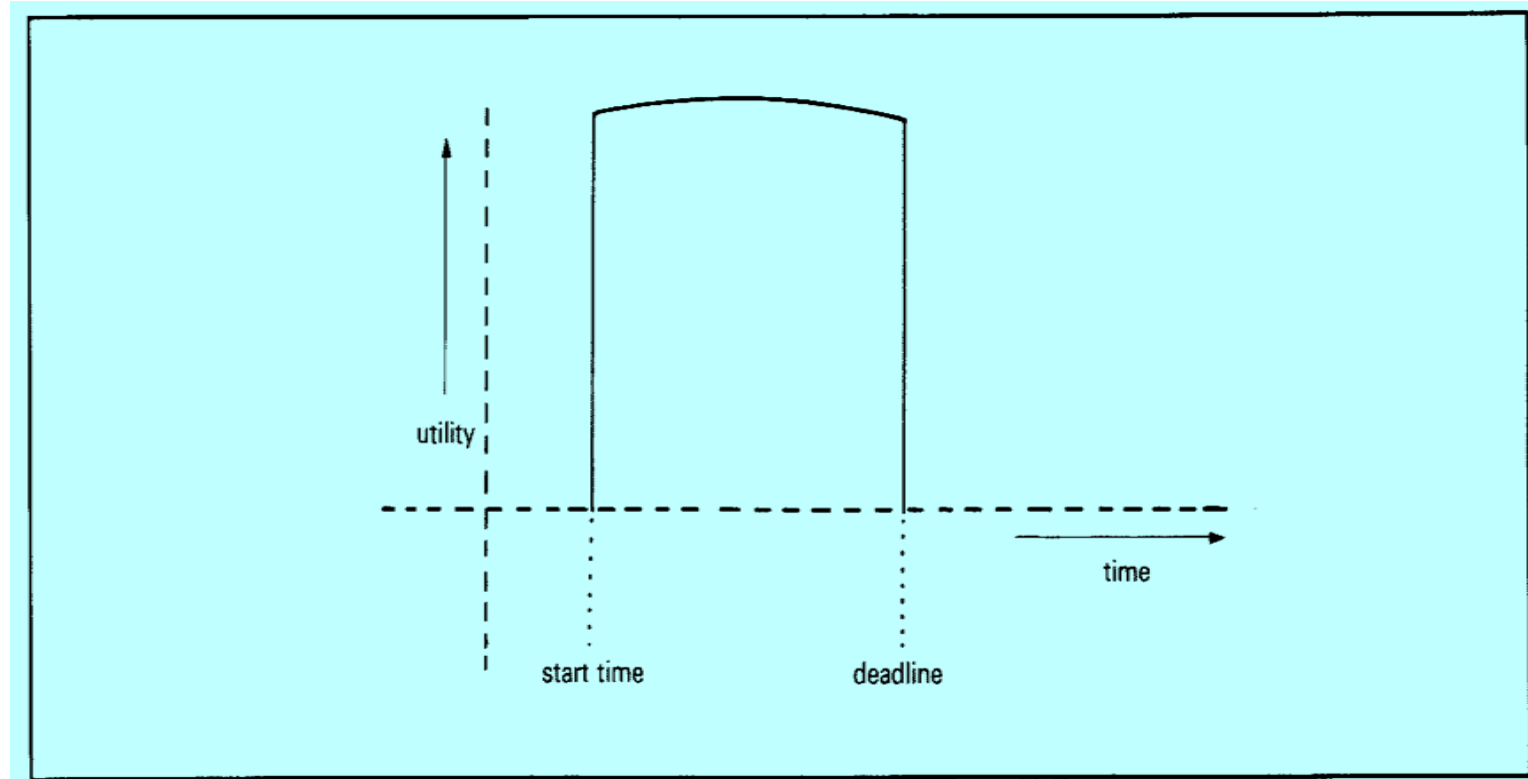
Real-Time Databases

- In Industrial controllers real-time database system is needed. Important issues considered include:
 - Scheduling transactions to meet deadlines.
 - Specifying explicit semantics for timing and other constraints.
 - checking the database system's ability to meet transaction deadlines during initialization of the application.

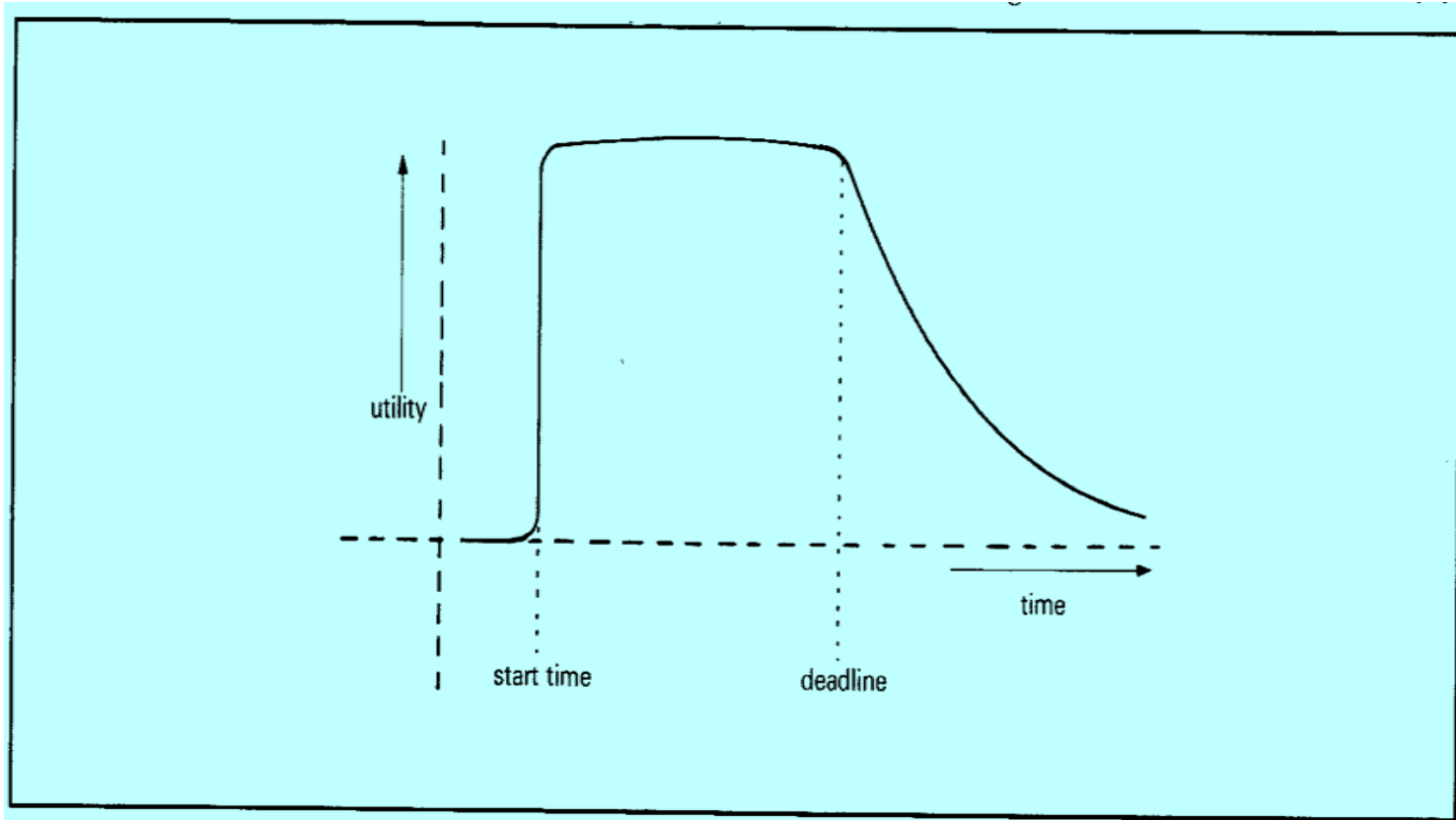
RT Systems Classification

- Hard (critical) RTS: if the deadlines is missed the consequence can be disastrous.
 - Examples: planes, trains, factory control, nuclear facilities, etc
- Soft (non-critical) RTS: If a transaction is missed it will only degrade system quality.
 - Example: multimedia, thermostat, etc
- Firm real-time systems are similar to soft real-time (late results are worthless), except late tasks are discarded.

Hard deadline



Soft deadline



Example: Cruise Control

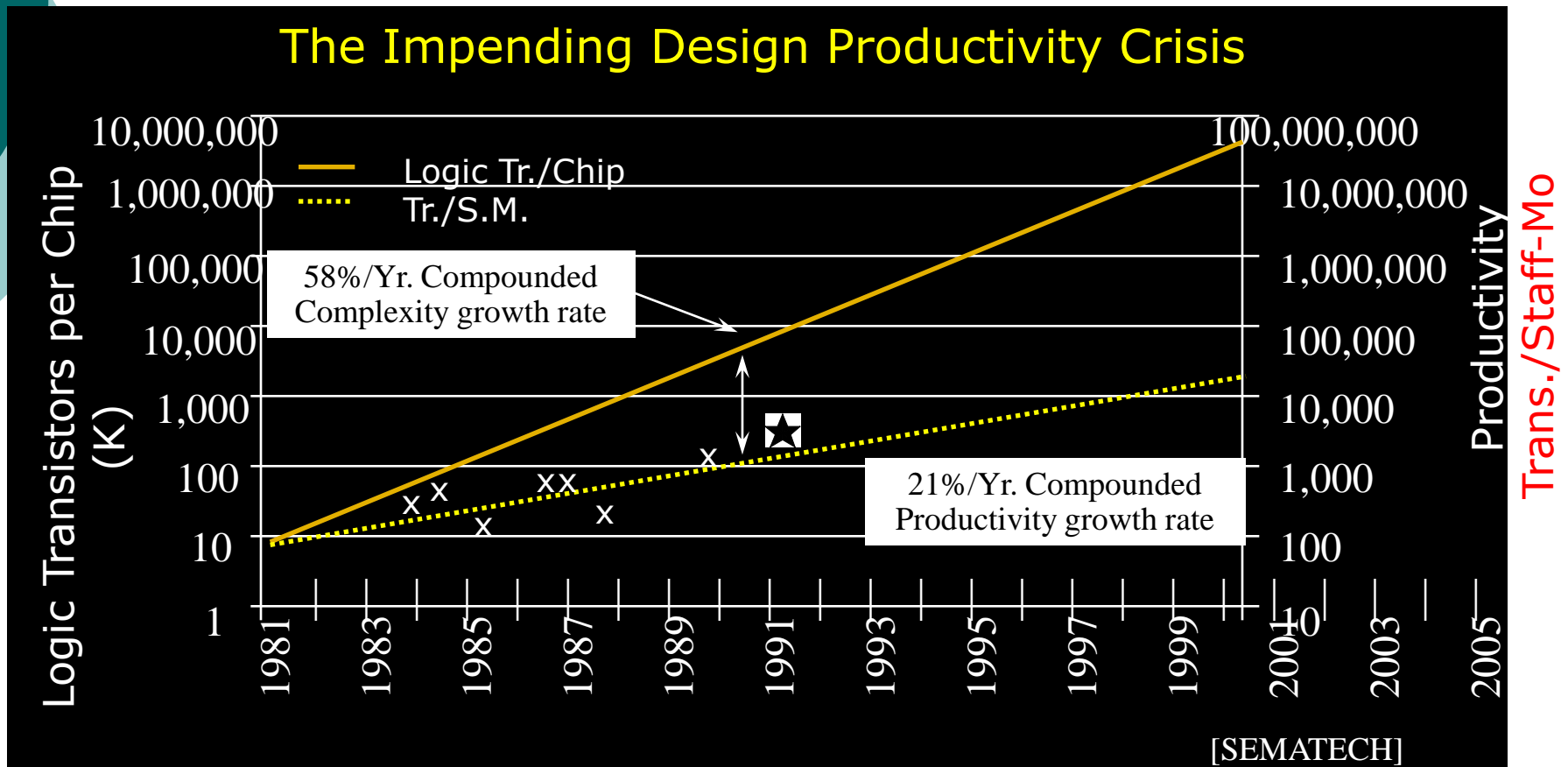
- Regulates speed of car by adjusting the throttle: driver sets a speed and car maintains it.
- Measures speed through device connected to drive shaft.
- Hard real-time: drive shaft revolution events.
- Soft real-time: driver inputs, throttle adjustments.



Embedded Systems: Complexity Issues

- Complexity of embedded systems is continually increasing
- Number of states in these systems (especially in the software) is very large
- Description of a system are becoming more complex, making system analysis extremely hard
- Complexity management techniques are necessary to model and analyze these systems
- Systems becoming too complex to achieve accurate “first pass” design using conventional techniques
- New issues rapidly emerging from new implementation technologies

Design Complexity Is Outpacing Designer Productivity



New Design Methods/Techniques Are Needed to Meet the Complexity Issues

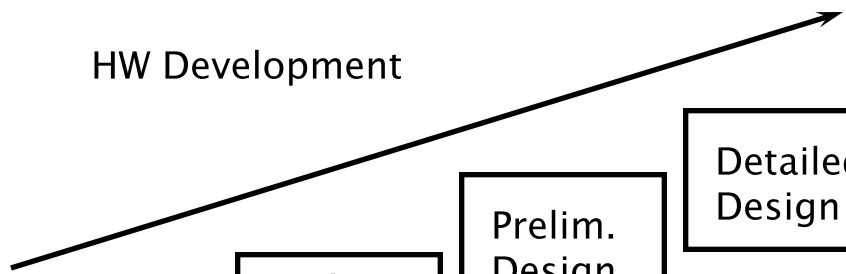
Techniques to Support Complexity Management

- **Delayed HW/SW partitioning**
 - Postpone as many decisions as possible that place constraints on the design
- **Abstractions and decomposition techniques**
- Incremental development
 - “Growing” software
 - Requiring top-down design
- **Description languages**
- **Simulation**

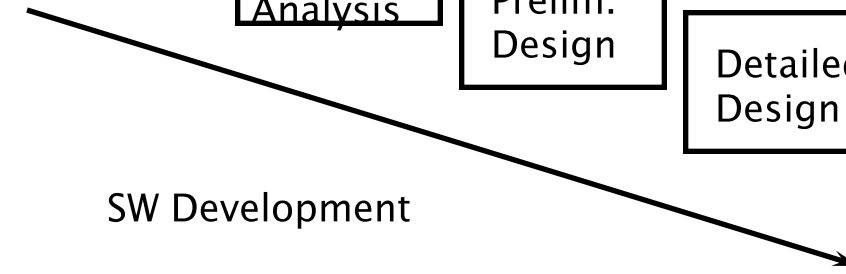
Current Hardware/Software Design Process

DOD-STD-2167A

HW Development



SW Development



System Concepts

Sys/HW Require. Analysis

Sys/SW Require. Analysis

Hardware Require. Analysis

Software Require. Analysis

Prelim. Design

Prelim. Design

Detailed Design

Detailed Design

Fabric.

Coding, Unit test., Integ. test

HWCI Testing

System Integr. and test

CSCI Testing

Operation. Testing and Eval.



Current Hardware/Software Design Process (Cont.)

- **Basic features of process:**
 - System immediately partitioned into hardware and software components
 - Hardware and software developed separately
 - “Hardware first” approach often adopted
- **Implications of these features:**
 - HW/SW trade-offs restricted
 - Impact of HW and SW on each other cannot be assessed easily
 - Late system integration
- **Consequences these features:**
 - Poor quality designs
 - Costly modifications
 - Schedule slippages

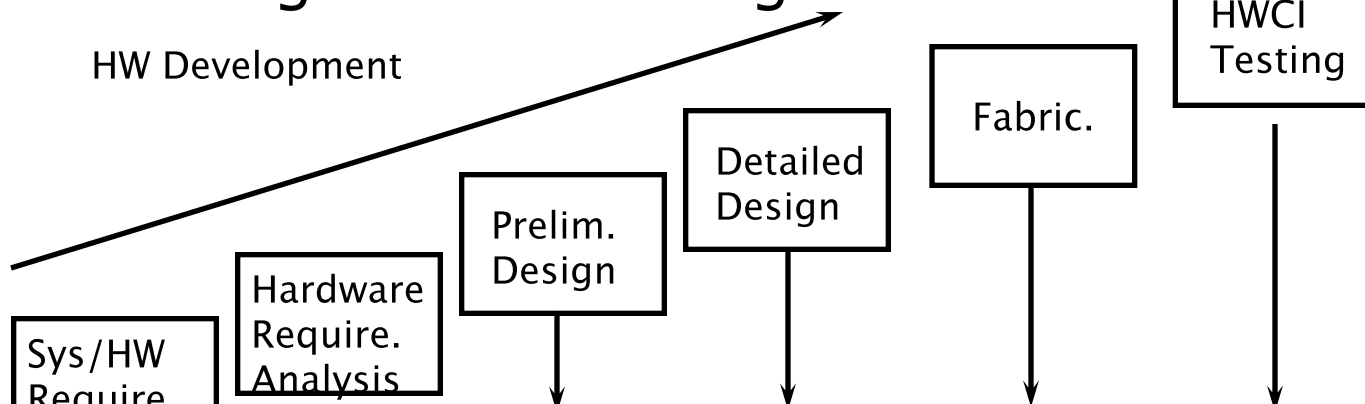
Incorrect Assumptions in Current Hardware/Software Design Process

- Hardware and software can be acquired separately and independently, with successful and easy integration of the two later
- Hardware problems can be fixed with simple software modifications
- Once operational, software rarely needs modification or maintenance
- Valid and complete software requirements are easy to state and implement in code

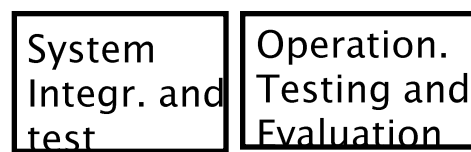
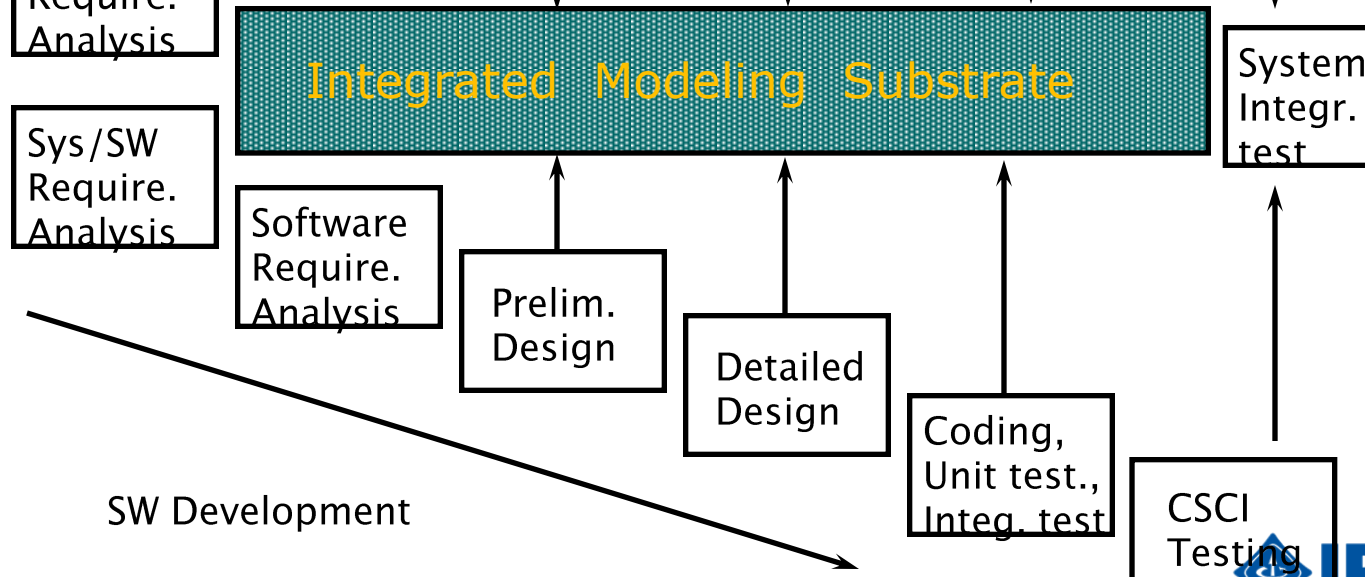
Directions of the HW/SW Design Process

Integrated Modeling Substrate

HW Development



SW Development





Thank you!