

GrK 1286

International Summer School

Course 6

Simulation of Micro- and  
Nanostructures

Lecture 2

**Prof. Dr.-Ing. Dietmar P. F. Möller**  
University of Hamburg, Germany  
Faculty of Mathematics, Informatics and Natural Sciences  
Department Computer Science  
Chair Computer Engineering  
[dietmar.moeller@informatik.uni-hamburg.de](mailto:dietmar.moeller@informatik.uni-hamburg.de)



# Content



- Introduction
- Basic Models of Semiconductor Devices
- Numerical Techniques Used in Device Simulation
- Exercises
- Instructors:
  - Prof. Dr.-Ing. Dietmar P. F. Möller
  - Dipl. Inf. Bernd Güde
  - Dipl. Inf. Massoud Najafi

Lecture 1

Lecture 2

# Content



- Introduction
- Basic Models of Semiconductor Devices
- **Numerical Techniques Used in Device Simulation**
  1. Computational Modeling and Simulation
  2. Systems Theory
  3. Modeling of Dynamic Systems
  4. Model Validation
  5. Digital Simulation
  6. Simulation Systems
  7. Simulation Examples
  8. Capability and Limitations of Devices Simulators
  9. Parallel Paradigm
- Exercises
  - Instructors:
    - Prof. Dr.-Ing. Dietmar P. F. Möller,
    - Dipl. Inf. Gunner Selke
    - Dipl. Inf. Bernd Güde
    - Dipl. Inf. Massoud Najafi

Questioning a simple answer

---



# Why modeling?

## Answer (1)



---

### That's why:

- Test is expensive – not achievable
- For safety reasons
- Non destructive testing
- Colored ambiguous of the environment
- Unstructured data
- Phenomenological description
- Very complex reality
- Nonlinearities
- Abstract system
- ....

## Answer (2)



---

### That's why:

Attempting to understand

- Unknowns
- Phenomena
- ...

in science and engineering, we are thinking in terms of models

Models are the most common possibility in science & engineering describing complex processes/systems and/or phenomena of real world problems

## Answer (3)



---

### That's why:

A model can be introduced as a reproduction of a dynamic system/process which with

- simulation experiments can be done much more easier as with the real system itself,

and/or will be

- the only possibility while it is not possible with the real object under test.

Based on simulation a deeper insight into a dynamic system/process is best possible under

- best case operating conditions
- worst case operating conditions
- real case operating conditions

# Computational Modeling and Simulation (1)

---



- Physics and Engineering are concerned with understanding and controlling the materials and forces of nature.
- Analyzing and improving the performance of systems, when the components of which originate from different domains:
  - adapting existing systems to new demands and/or conditions
  - designing new applications.
- Systems may include components derived from many different scientific domains:
  - Physics
  - Engineering
  - Chemistry
  - ....
- Problem oriented solutions have been found applying appropriate mathematical models and computer simulation tools.

## Computational Modeling and Simulation (2)

---



- Computational modeling and simulation has become one of the most powerful analysis and design methods available today, particularly for analyzing and controlling dynamic systems.
- This has been primarily due to remarkable advances in:
  - systems theory,
  - computer science,
  - engineering,
  - other human activities in engineering and science.
- A recent White House report identifies computational modeling and simulation as one of the key enabling technologies of the 21<sup>st</sup> century. Its applications are virtually universal.

# Computational Modeling and Simulation (3)

---



- M&S can be considered as an iterative process that contains
  - successive mathematical model building
  - computer-assisted simulation
  - approach to manipulate complex dynamic systems in accordance with the respective aims and scopes
  - changing model structure, and/or its parameters, and/or its inputs and/or its outputs
  - accurately match of the real dynamic system behavior
  
- A derived model achieves its purpose when an optimal match is achieved between
  - simulation results, based on the mathematical model,
  - data sets gathered through system measurements and experimentation.

# Computational Modeling and Simulation (4)

---



- Model building entails utilization of several types of information sources:
  - goals and purpose of modeling
  - determining boundaries
  - components of relevance
  - level of details
  - a priori knowledge of the real dynamic system being modeled
  - data sets gathered through experimentation and measurements on the systems inputs and outputs
  - estimations of non-measurable data, and/or state space variables of the real dynamic system
  - ...

# Systems Theory (1)



- A system contain the following objects:
  - Elements
  - Relations
  - Attributes
- **Element** can be a component, part, and so forth
- **Relation** is cooperative, coupling, and so on.
- **Attribute** introduce property, feature, signature, and so on.
  - provide connections between the system and the system environment.
  - is called a system state if it describes a system condition.
  - are called a system-related internal descriptions if they interact with each other.
- Assuming that  $A$  is a nonempty set of attributes  $\alpha$  and  $B$  are nonempty sets of relations, a system description  $\beta$  can be defined as

$$F := (\alpha \in A, \beta \in B ).$$

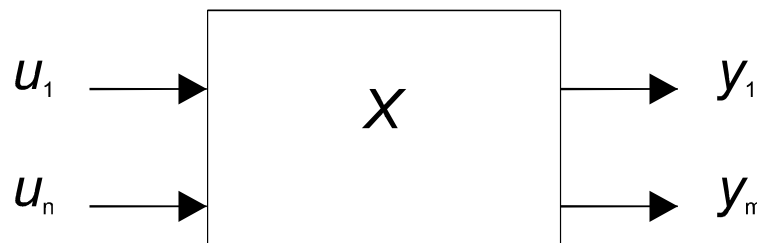
## Systems Theory (2)



- The structural description of dynamic systems can be provided by a matrix notation, which contains an:
  - $u$  Input vector
  - $y$  Output vector
  - $S$  Operator matrix.
- In the formula

$$y(t) = S(t) \cdot u(t)$$

the system operator matrix  $S$  can be expressed as the state matrix  $X$ , shown in the following figure



## Systems Theory (3)



- A system is called a dynamic system , if it is given by
  - $x(t_1)$ , the state vector
  - $x(t_0)$ , the initial state
  - $u(t)$ , the input function
  - $(t_0, t_1)$  for all  $t_1 > t_0$ , the time interval
  
- A dynamic system is of finite order if the state vector  $x(t_1)$  has a finite number of components.

# Systems Theory (4)



- A dynamic system is said to be continuous in time if the time interval  $I \subset \mathfrak{R}$  contains the definition range of the functions  $u(\cdot)$ ,  $x(\cdot)$ , and  $y(\cdot)$ , discrete in time, if the time interval  $I$  contains the definition range of the functions  $u(\cdot)$ ,  $x(\cdot)$ , and  $y(\cdot)$ .
  
- Continuous-time systems can be described by
  - ordinary differential equations (ODE's),
  - partial differential equations (PDE's).
  
- Discrete-time systems are described by
  - Petri-nets,
  - Queues,
  - Markov-chains,
  - and so forth.

## Systems Theory (5)



- The set  $Ax = b$  with  $m$  equations and  $n$  unknowns has solutions if and only if  $\text{rank}[A] = \text{rank}[Ab]$ .

Let  $r = \text{rank}[A]$ .

If condition  $\text{rank}[A] = \text{rank}[Ab]$  is satisfied and if  $r = n$ , then the existence of solutions is unique.

- The set  $Ax = b$  with  $m$  equations and  $n$  unknowns has solutions if and only if  $\text{rank}[A] = \text{rank}[Ab]$ .

Let  $r = \text{rank}[A]$ .

If condition  $\text{rank}[A] = \text{rank}[Ab]$  is satisfied and if  $r < n$ , an infinite number of solutions exists and  $r$  unknown variables can be expressed as linear combinations of the other  $n - r$  unknown variables, whose values are arbitrary.

# Systems Theory (6)



- A linear dynamic system

$$\dot{x} = A \cdot x + B \cdot u$$

$$y = C \cdot x + D \cdot u$$

is said to be

- Controllable at time  $t_0 \in T$ , if for a finite  $t_1 > t_0$ ,  $t_1 \in T$  exist.
- Completely controllable, if for each  $t_0 \in T$  a finite time  $t_1 > t_0$ ,  $t_1 \in T$  exist.
- Differential or particularly controllable, if for each  $t_0 \in T$  and each finite
- $t_1 > t_0$ ,  $t_1 \in T$ , the matrix

$$w(t_0, t_1) = \int_{t_0}^{t_1} \Phi(t_0, \tau) B(\tau) B^T(\tau) \Phi^T(t_0, \tau) d\tau$$

is regular.

- A linear dynamic system

$$\dot{x} = A \cdot x + B \cdot u$$

$$y = C \cdot x + D \cdot u$$

is said to be

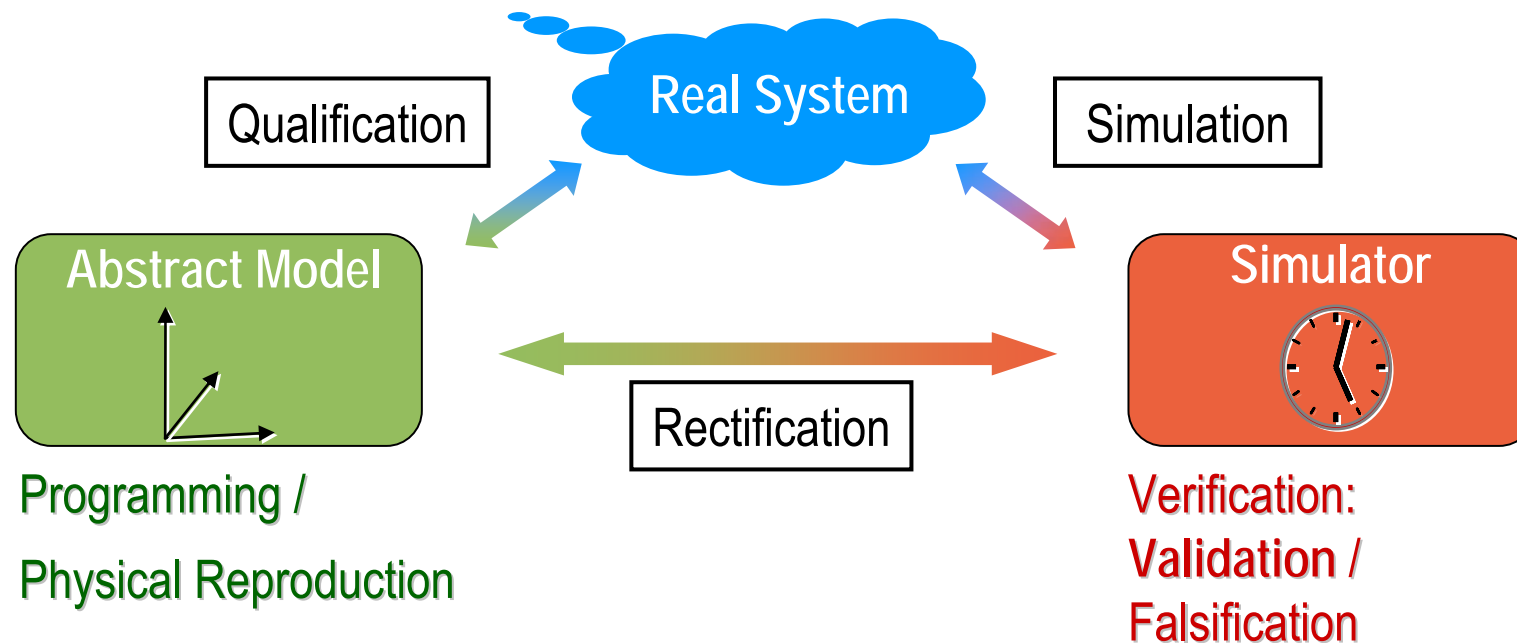
- Observable at time  $t_0 \in T$ , if for a finite  $t_1 > t_0$ ,  $t_1 \in T$  exist
- Completely observable, if for each  $t_0 \in T$  and each finite  $t_1 > t_0$ ,  $t_1 \in T$  exist
- exist
- Differential or particularly observable, if for each  $t_0 \in T$  and each finite  $t_1 > t_0$ ,  $t_1 \in T$ , the matrix

$$m(t_0, t_1) = \int_{t_0}^{t_1} \Phi^T(t_1, t_0) C^T(t) C(t) \Phi(t_1, t_0) dt$$

is a regular one.

# Modeling Dynamic Systems (1)

- Modeling is a complex procedure that contains several steps:
  - qualification,
  - rectification,
  - and finally verification



# Modeling Dynamic Systems (2)



## ■ Qualification

- process that generates the so-called abstract model.
- tries to describe the real dynamic system in an abstract manner by using elements, relations, and attributes for the description.

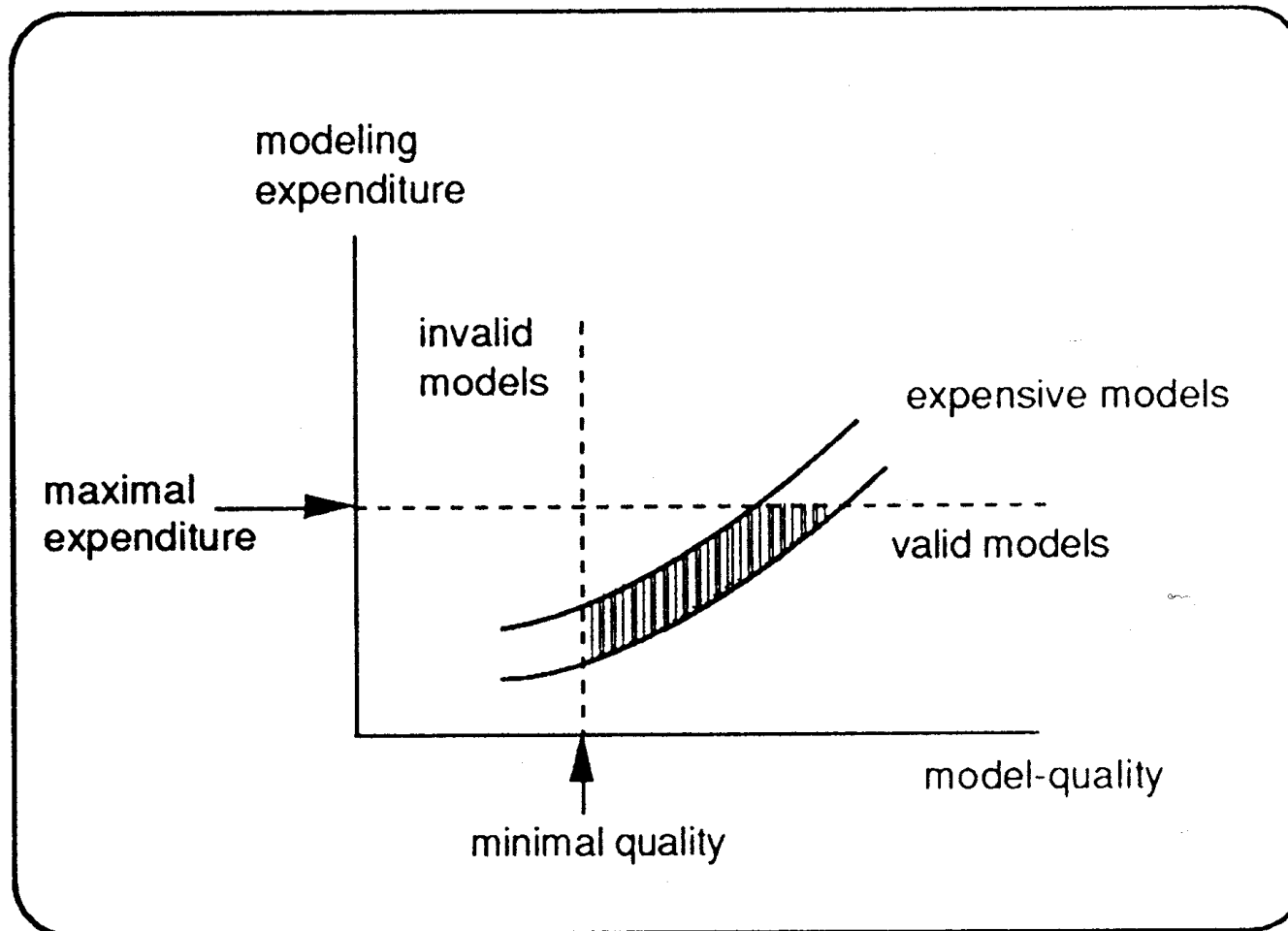
## ■ Rectification

- process which transforms the abstract model into a mathematical model, the so-called real model.
- decides the proper form for the realization including implementation, iteration algorithms, programming, imitation/simulation based on mechanical, electrical, isomorphism, etc.

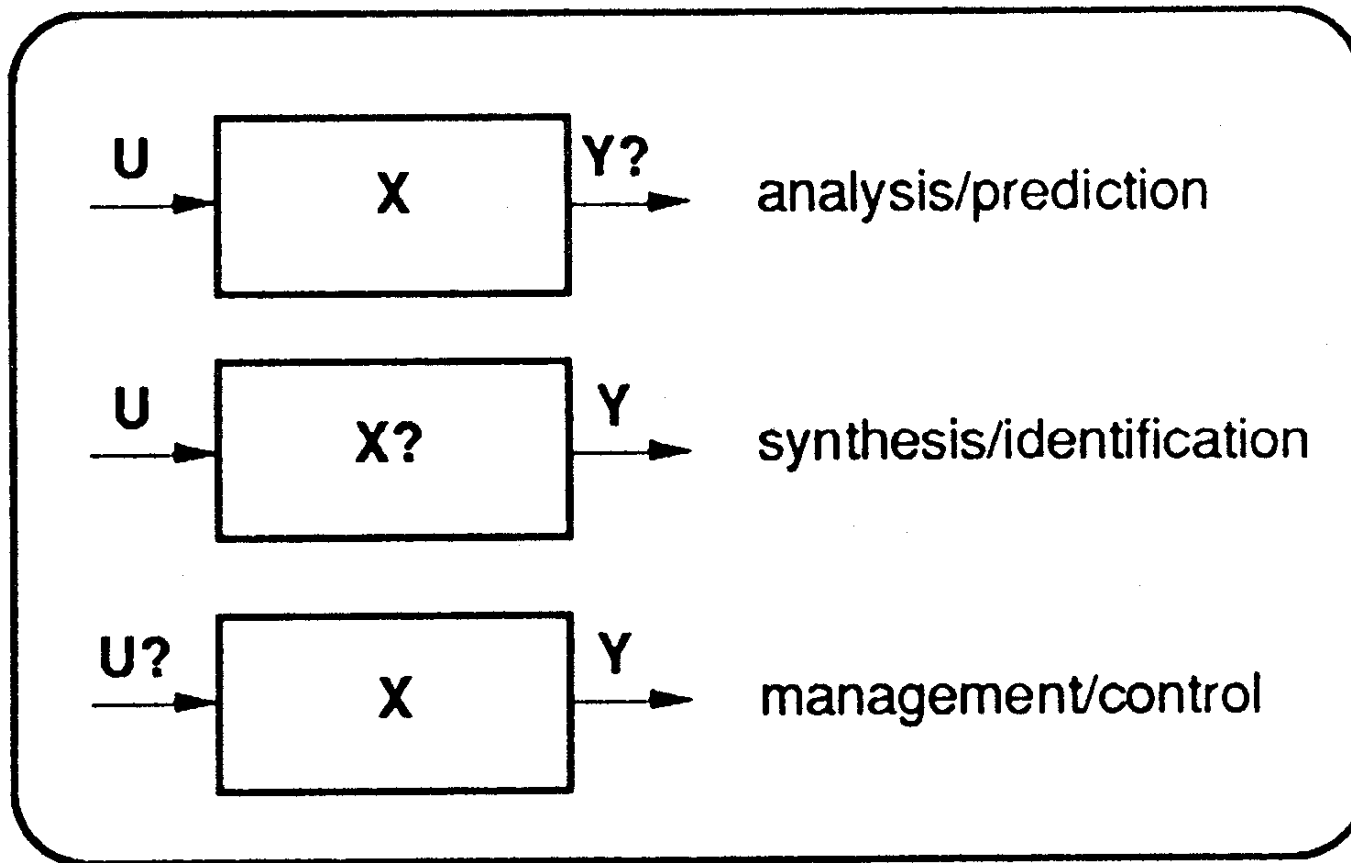
## ■ Verification

- process that is focused on fit or non fit of the model due to the respective dynamic behavior of the system.
- includes the **validation** of the model, i.e. quality of the model, and the falsification of the model, meaning less fitting.

# Modeling Dynamic Systems (3)



# Modeling Dynamic Systems (4)



# Modeling Dynamic Systems (5)

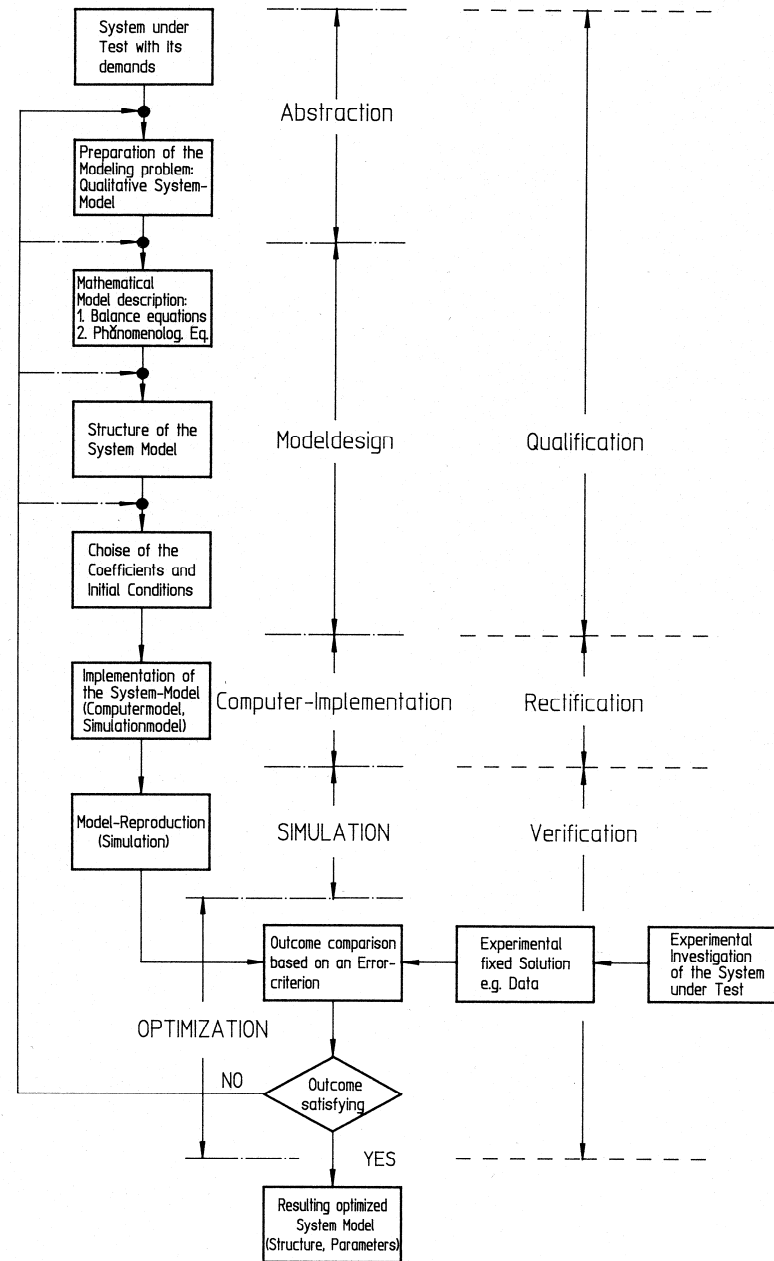
---



## Deductive method of theoretical axiomatic modeling:

- Bottom up approach, starting with high degree well established a priori knowledge of system elements to build up a mathematical model which describes the system under test in a proper and well defined way
- Problems may occur in assessing the range of applicability of these models, hence the deductive method has to be expanded by an experimental model validation technique
- Model verification by checking whether simulation results and data known from the real system match the error margin or not
- Model fit the assumed performance when the results, obtained from the model by simulation, compared with the results which may be data or measures on the real system, are within the error margin
- If model is decided being unsatisfying, a modification is necessary at the different levels of the deductive modeling scheme

# Modeling Dynamic Systems (6)

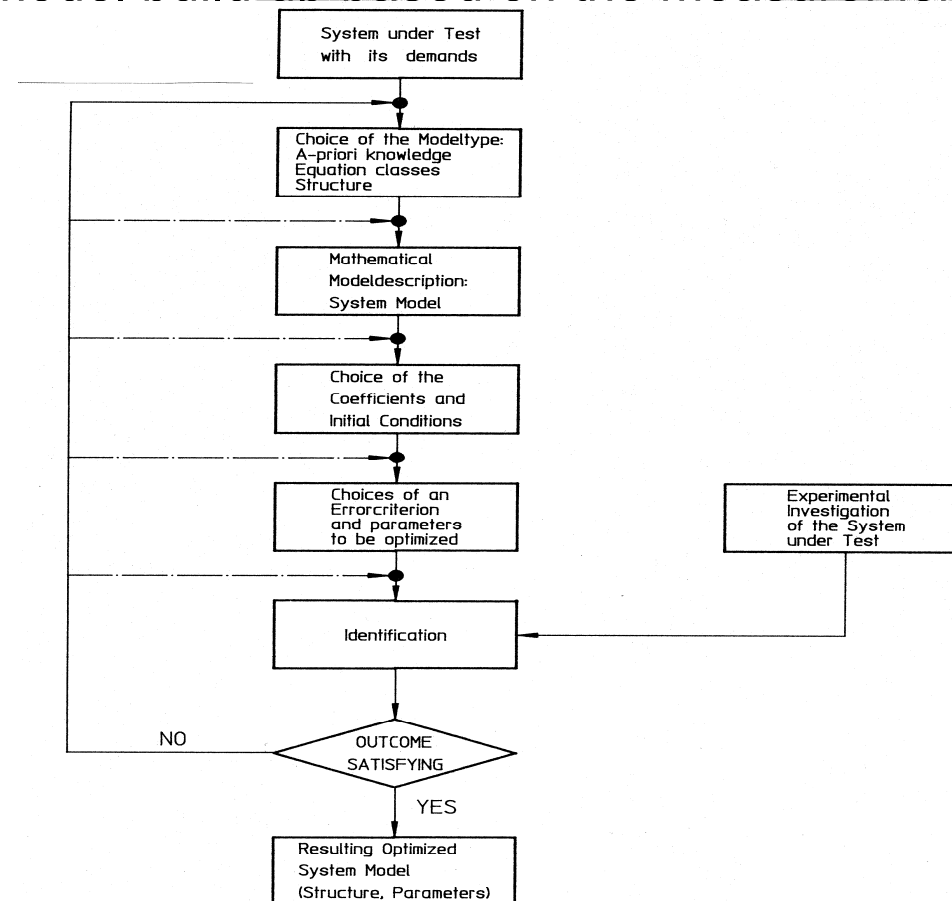


# Modeling Dynamic Systems (7)



## Empirical Method of Experimental Modeling:

- Based on measurements available on the inputs and outputs of a real system
- Experimental model build up based on the measurements



# Model Validation (1)



- Model validation is a process that involves checking
  - the focus,
  - the range of flexibility,
  - the fulfilling of the purposeaccording to the specification of the model.
  
- More in general model validation is a multi dimensional procedure reflecting the
  - model purpose,
  - current theories,
  - experimental test datarelating to the particular system of interest together with other important knowledge.

## Model Validation (2)



1. **Internal criteria:** enabling conditions within the model itself to be judged without external reference to the model purpose, theory and/or data.
  - **Examples:**
    - **Consistency:**  
requiring that the model formulated contains no logical, mathematical or conceptual contradictions.
    - **Algorithmic validity:**  
requiring that the algorithm for analytical solution or numerical simulation is appropriate and leads to accurate solutions.

# Model Validation (3)



2. **External criteria:** referring to the model itself, like the model purpose, theory an/or data.

■ **Examples:**

- **Empirical validity:**  
requiring that the model formulated should correspond to the available data.
- **Theoretical validity:**  
requiring that the model should be consistent with accepted theories and/or models.
- **Pragmatic validity:**  
requiring that testing the extent to which the model satisfies the objectives for which it has been developed.
- **Heuristic validity:**  
requiring in connection with tests that are associated with the assessment of the heuristic potential of the model, e.g. for scientific explanation, discovery, and/or hypothesis testing.

## Model Validation (4)

---



- Considerations of validity are required from the very beginning of model building.
- Empirical and theoretical validity can be used by examining whether the respective validation criteria are met or not, which then can be used as a performance index.
- A performance index (PI) is a quantitative measure of the performance (validity) of a model of a system
- PI is chosen so that emphasis is given to the important constraints.

## Model Validation (5)



- A suitable performance index  $PI$  is the integral of the square of the error

$$PI = \int_0^T e^2(t) dt$$

*where  $T$  is a chosen finite time, so that the integral approaches a steady-state value of the transient behavior and  $e$  is a measure of the error between the real-world system and the system model.*

- Another possible performance criterion is the integral of the absolute magnitude of the error, which can be written as:

$$PI = \int_0^T |e(t)| dt$$

This performance index is particularly useful for computer-simulation studies.

## Model Validation (6)



- In order to
  - reduce the contribution of the large initial error to the value of the performance integral
  - place an emphasis on errors occurring later in the response,another performance index has been proposed:

$$PI = \int_0^T t \cdot |e(t)| dt$$

This performance index is designated the integral of the time multiplied by the absolute error.

- A similar performance index is the integral of time multiplied by the squared error, which is:

$$PI = \int_0^T t \cdot e^2(t) dt$$

- The general form of the performance index is:

$$PI = \int_0^T f[e(t), u(t), y(t), t] dt$$

# Analog Simulation (1)

---



- Analog simulation
  - is a so-called continuous time simulation method
  - is based on so-called analog computers
  - are electronically devices that allow to handle DC voltage signals with standardized components
- Standardized components
  - are based on operational amplifiers as functional kernel
  - realize mathematical operations between time variant input and time variant output variables
- Mathematical operations that can be solved are based on Ordinary Differential Equations (ODEs)
- Linear operating range of op amps is  $\pm 10$  V

# Analog Simulation (2)



- Linear analog computing elements are:

- Coefficient potentiometer:  $u_a = c \cdot u_i$

- Open amplifier:  $u_a = A_0 \cdot u_i$

- Summing unit:  $u_a = \sum_i V_i \cdot u_i$

- Integrator:  $u_a = -\frac{1}{T_i} \int \left( \sum_i V_i \cdot u_i \cdot dt - u_{a0} \right)$

- Nonlinear analog computing elements are:

- Multiplication:  $u_a = u_1 \cdot u_2$

- Divider:  $u_a = \frac{u_1}{u_2}$

- Function generator:  $u_a = f(u_i)$

## Analog Simulation (3)



- Analog computer programming

$$\dot{x}_i = f_i(x_1, x_2, \dots, x_n; u_1, u_2, \dots, u_r), i = 1, 2, \dots, n$$

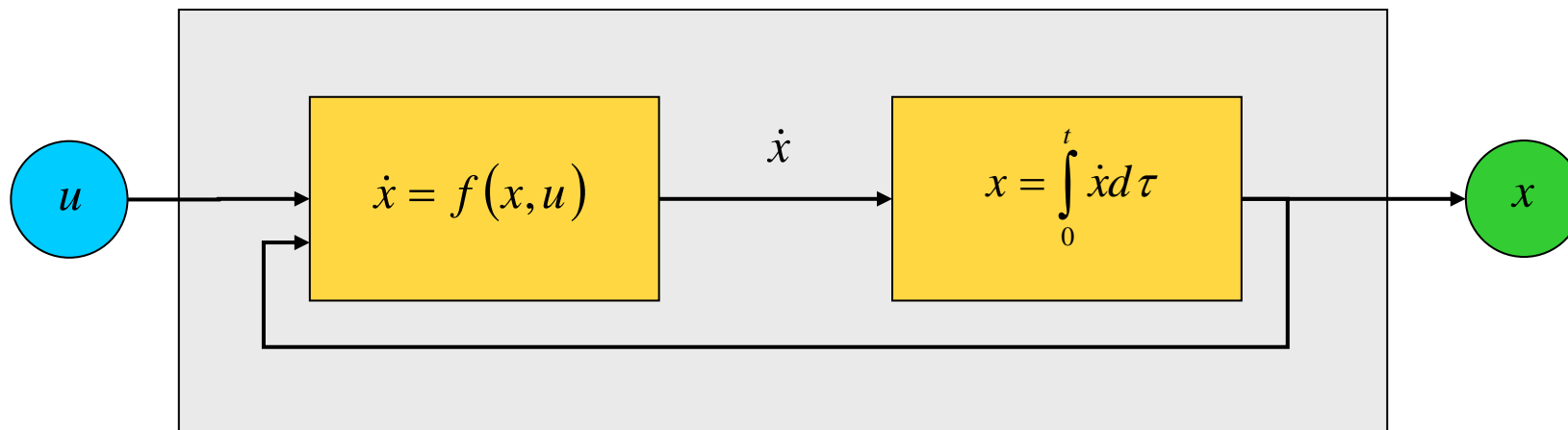
To determine the behavior of the state variables  $x_i(t)$  over the simulation time interval  $T$  we need:

- initial conditions  $x_i(t_0) = x_i^0$
  - the stimulation function  $u_j(t)$
- Assuming that the equation above can be expressed as signal flow chart, we find two simultaneous executed subtasks:
    - calculating the derivatives of the state variables by exploiting the algebraic function  $f(\cdot)$  at the right side of the equation, taking as a basis the actual values of  $x(t)$  and  $u(t)$
    - integrating the derivatives of the state variables for the sought-after solution of the state variables  $x_i(t)$

# Analog Simulation (4)



- Analog computer programming



Signal flow chart for integrating a system of state variables with an analog computer

# Analog Simulation (5)



# Analog Simulation (6)



# Analog Simulation (7)

---



## Constraints of analog computing

- System variables can not be transformed into the programming form (machine equation) of the analog computer
- Dynamic transient behavior in one and the same system model can show different velocity, meaning that the scaling speed runs in different time intervals'
- Stiff ODEs result in large scale ranged coefficients which can not be realized with normal components
- Operational amplifiers used for implementing the ODE show drift effects, which means that the output signal will change over time without the adequate change of the input signal

# Analog Simulation (8)



## Example

- A large refinery may have as many as 1000 feedback loops, a paper mill in comparison, may have up to 5000 loops, which means we will have in between 1000 and 5000 controller equations and additional plant dependent equations.
- Such type of dynamic system could not be solved with the analogous devices of the past, as well as with the so called hybrid computers which combine analogous and digital computer facilities.
- The innovation sequence in (digital) computer supported simulation from the early 50th till today can be introduced as a sequence of innovations.

# Digital Simulation (1)



- Digital Computers:
  - became available in the late 1950s. They replaced analog hardware and relays, the typical components of (analog) computers in the past.
- **1955-1960** User programming, no user support, model building based on higher programming languages like FORTRAN, ALGOL, etc.
- **1960-1965** 1<sup>st</sup> generation of simulation languages, basic user support through automatically generated computation ability, GUI
- **1965-1970** 2<sup>nd</sup> generation of simulation languages, better software tools, interactivity
- **1970-1980** 3<sup>rd</sup> generation of simulation languages with extended and new possibilities of simulation tools like combined simulation, etc.
- **1980-1990** 4<sup>th</sup> generation of simulation languages with domain specific and specialized simulators, and better animation, easier model implementation
- **1990-2000** 5<sup>th</sup> generation of simulation languages/software, embedding artificial intelligence, model specification and experimental environment, expanding the possibilities of the tools of the 4th generation like graphic
- **2000-2010** 6<sup>th</sup> generation of simulation languages/software, embedding object oriented modeling, soft computing methodology (fuzzy sets, neuronal nets, genetic algorithms, evolution theory, probabilistic methodology), virtual and augmented reality environments in simulation

## Digital Simulation (2)

---



- Simulation software permits the implementation of the mathematical models and its parameters.
- Simulation software contains
  - a simulation language
  - a set of commands for the control of the simulation process
- Simulation software systems allow the user to implement and simulate models quickly and efficiently without being an expert in programming languages or numerical integration.
- Innovation in modern computer technology increases the possibilities of simulating complex systems.

## Digital Simulation (3)

---



- Computer simulation based on mathematical models is a third column apart from theory and experiments
- In many cases experiments can not be realized:
  - safety or security
  - time consuming
  - costs
  - etc.
- Computational modeling and simulation gives the possibility to obtain solutions for complex problems
  - ➔ results in a better understanding e.g. semiconductor devices and the theory behind them.

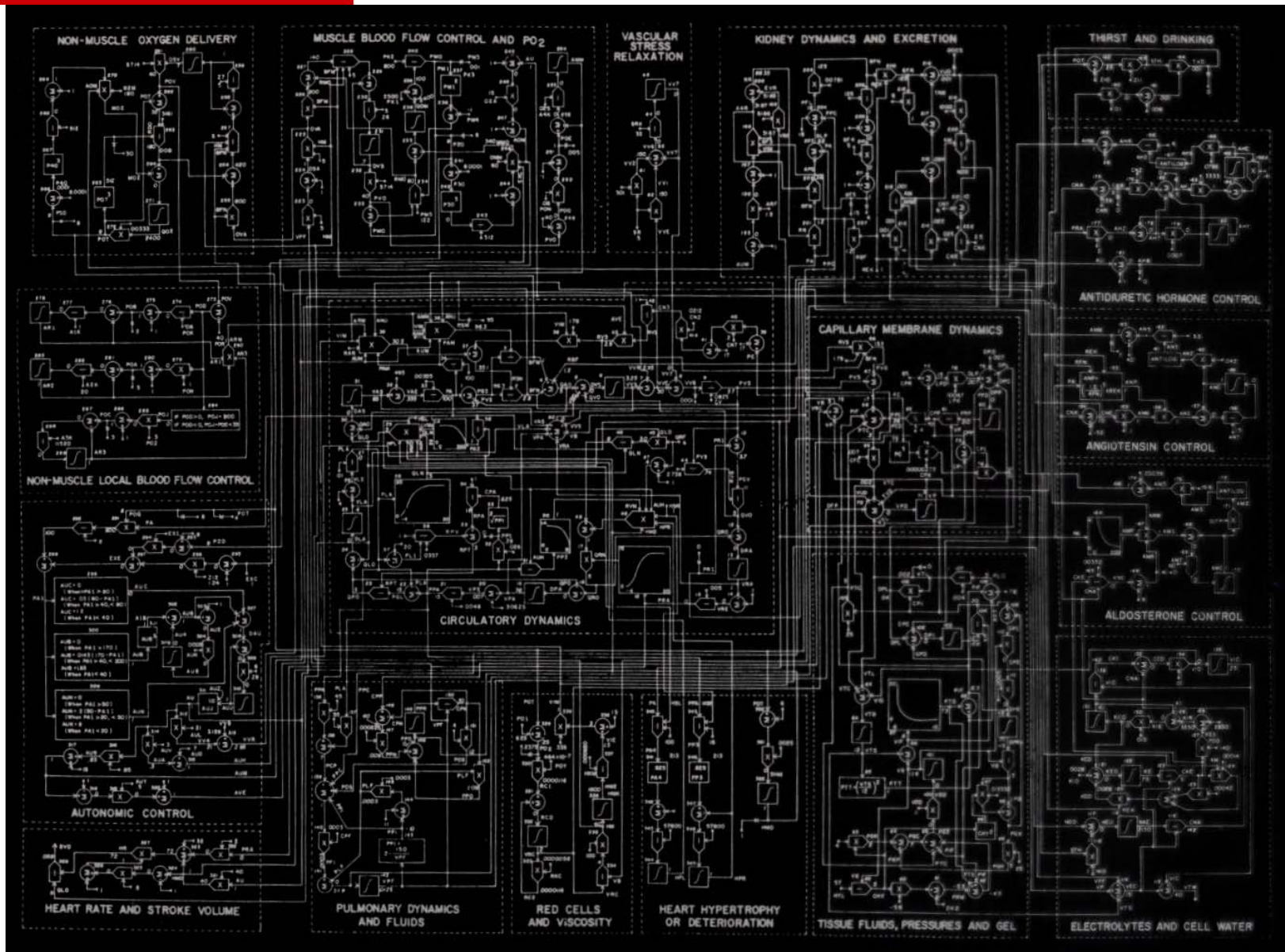
## Digital Simulation (4)

---



- Modeling and simulation provide information about the usability of hypotheses e.g.:
  - for astrophysical problems such as the time course of matter distribution of a star formation,
  - in molecular modeling due to the chemical structure for an optimal drug design,
  - in the nonlinear discrete modeling of tumor growth in humans and the respective test series for intra-individual sufficient tumor therapy, etc.
  - in physiology such as modeling and simulation of the nonlinear overall control mechanisms to explain hypertension,

# Digital Simulation (5)



# Digital Simulation (6)



- Digital simulation software systems consist of
  - the simulation language
  - the translator.
- The simulation software translator generates the compiler code for the simulation program, which will be translated into machine code, and linked with standard elements from the system library, allowing the simulation to run.

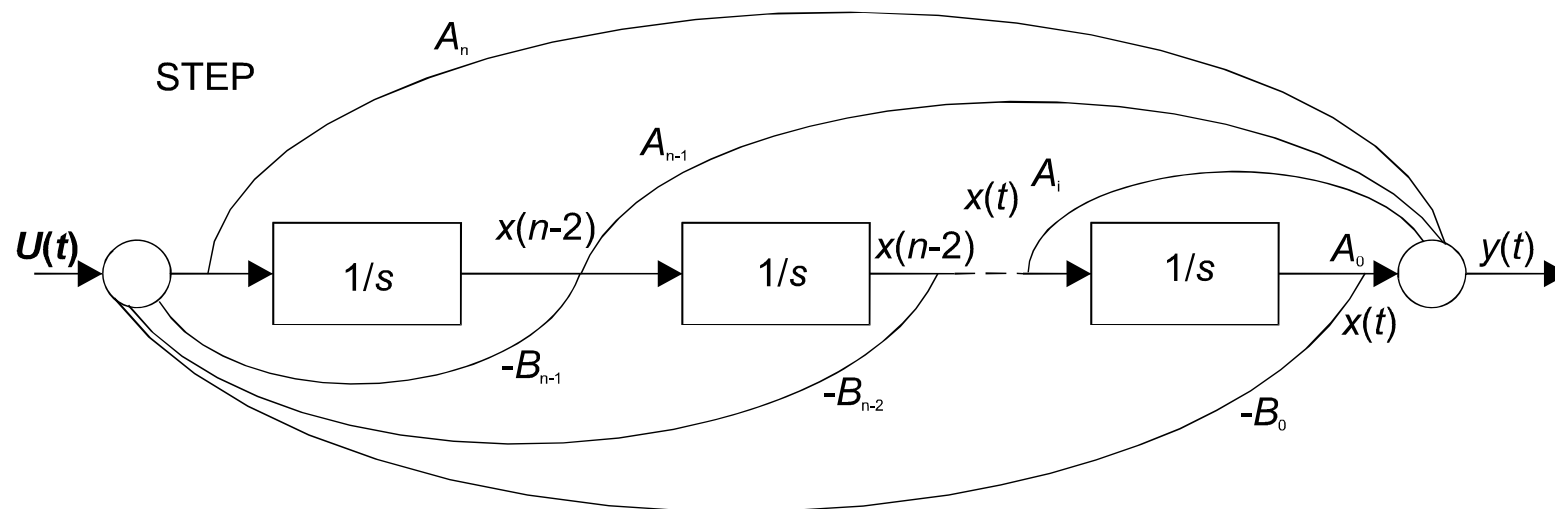
## Definition

A simulation run can be defined as a calculation of the state-variables transient behavior in discrete steps, based on a calculation sequence of the mathematical model, which starts with the initial condition  $x(t_0)$ , and finishes with the final state  $x(t_e)$ . The calculation of which can be based on fixed or variable step width.

# Digital Simulation (7)



Simulation systems can solve linear and nonlinear differential equations of  $n$ -th order, describing the engineering system. In most cases the higher order differential equations are reduced as sets of first-order differential equations, which means for the block oriented simulation software the decomposition into a block oriented scheme of first-order blocks, which can easily be solved using numeric integration.



## Digital Simulation (8)



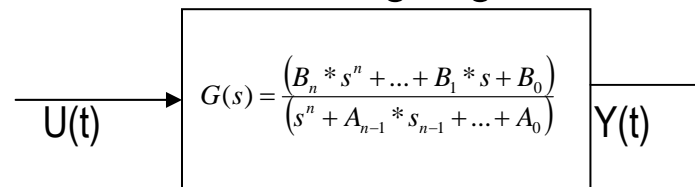
A general approach for decomposing high order transfer functions, or differential- or state equations to a block scheme (in PSI) will be given, supposing, having a transfer function  $G(s)$

$$G(s) = \frac{(B_n * s^n + \dots + B_1 * s + B_0)}{(s^n + A_{n-1} * s_{n-1} + \dots + A_0)}$$

or, equivalently:

$$y^n(t) + \dots + B_1 * y'(t) + B_0 * y(t) = A_n * u^n(t) + \dots + A_1 * u'(t) + A_0 * u(t)$$

This high order system can be decomposed by introducing the state variable  $x(t)$  as shown in the following figure



## Digital Simulation (9)



Interactive block oriented simulation software systems can be used for studying the behavior of continuous-time, and discrete-time systems. Block oriented simulation software systems use differential equations that represent the simulation model.

### Example #13

Supposing the second-order differential equation

$$y''(t) = -y'(t) - y(t) + u(t)$$

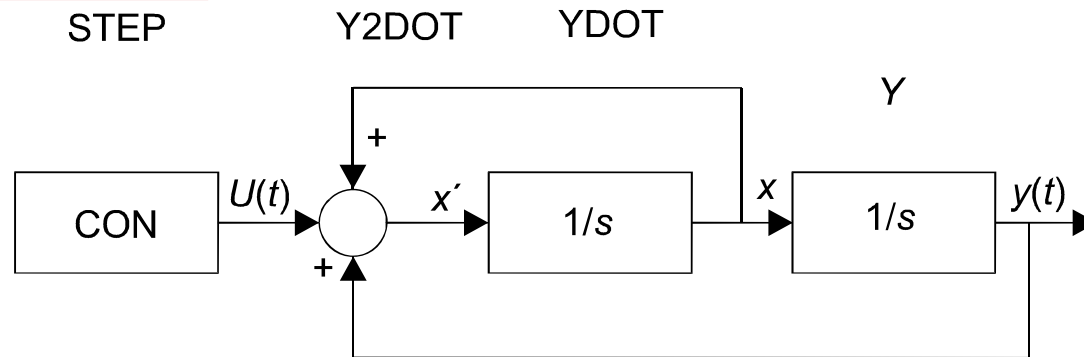
which can be rewritten as set of first order integral equations, yields

$$y'(t) = y'(0) + \int_0^t y''(\tau) d\tau$$

and modeled as a block oriented representation

$$y(t) = y(0) + \int_0^t y'(\tau) d\tau$$

# Digital Simulation (10)



Y2DOT represents the second derivative, YDOT represents the first derivative,  $Y$  is the original function, and STEP represents a constant  $U(t)$ .

The second-order system needs specific components such as:

- Structure
- Parameters
- Numeric integration
- Output

The structure is given by defining the inputs of each block. If the inputs of all blocks are defined the structure of the simulation model is known, and the block-oriented simulation software is able to calculate the behavior of the system

# Digital Simulation (11)



The idea behind the block oriented simulation approach is that any simulation model can be built up from basic block elements such as integrators, gain-transfer functions, look-up tables, nonlinear function blocks, constants, etc. Each block is identified by its name, which determines the block and its output, as well as the block type and its inputs. We have:

$$\begin{aligned} \text{STEP} &= u(t) \\ Y &= y(t) \\ \text{YDOT} &= y'(t) \\ \text{Y2DOT} &= y''(t) \end{aligned}$$

Together with the simulation software specific control commands the block structure can be defined. For PSI we use B as follows:

PSI·B

*Configuration Specification*

*Block, Type, Input1, Input2, Input 3*

B·STEP, CON : STEP is a CON block

B·YDOT,INT,Y2DOT : YDOT is an integrator with Y2DOT as input

B·Y,INT,YDOT : Y is an integrator with YDOT as input

B·Y2DOT,SUM,STEP,Y,YDOT : Y2DOT is a summer to add all inputs

B·

## Digital Simulation (12)



Parameters of each block can be defined using specific control commands, which depends on the simulation software used. For PSI P is used, hence PSI·P

*Parameters*

*Blocks, Par1, Par2, Par3*

P·STEP,1 : STEP has value 1

P·Y,0,1 : initial condition=0; input gain=1

P·YDOT,0,1 : initial condition=0; input gain=1

P·Y2DOT,1,-1,-1 : gains of the corresponding inputs

P.

The variables that determine the **numeric integration**, are the integration method, the integration interval, and the simulation time, which can be defined by specific control commands. For PSI we use T and obtain:

PSI·T

Integration interval=0.1

Integration time=10.0

Using 0.1 for the integration interval and 10.0 for the integration time, the simulation run will be calculated for 10 time units with an integration interval of 0.1 time unit.

# Digital Simulation (13)



Blocks are calculated during simulation run, some of which can be shown on the screen. Supposing  $y(t)$  is the output variable shown on the screen, some specific control commands needed, depending on the simulation system

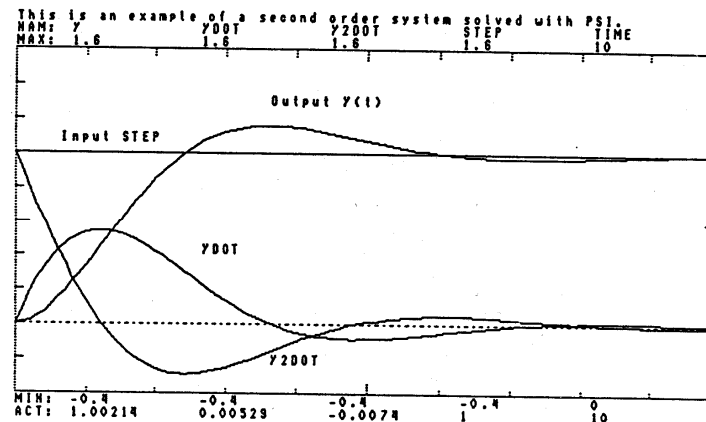
## PSI·O

Name of blocks to be shown=?

Now the required transient behavior can be calculated and the output variable indicated shown on the screen. The simulation run will start using a specific control command.

## PSI·R

PSI allows the presentation of more than the one variable, more complex figures can be shown representing the system variables, including inputs and outputs.



## Digital Simulation (14)

---



- The **structural statements** of equation oriented simulation systems transform the mathematical expressions of the real-world system through algebraic statements into functional blocks.
- The **data statements** connect the symbolic defined parameters, constants, and initial conditions, with the respective numerical values.
- The **control statements** schedule the time dependence of the output variables.
- **INITIAL** contains the parameter values and initial values.
- **DYNAMIC** contains the model description.
- **END** terminates the simulation run, and contains the parameters with its new values, which may be calculated in a second run.

# Digital Simulation (15)



**Execution control statement** used to specify items in relation to the actual simulation run, like run time, integration time., relative error, etc. The most important ones are:

- **TIMER**: this feature allows to specify the values of the following systems variables
- **PRDEL**: print increment for output printing
- **OUTDEL**: print increment for print-plot output
- **FINTIM**: maximum simulation value for independent variables
- **DELT**: integration interval
- **DELMIN**: minimum allowable integration interval
- **FINISH**: this label allow to specify run termination conditions
- **METHOD**: this label specifies integration routine to be used for simulation. If none is specified, RKS=default
- ❖ **ADAMS**: 2<sup>nd</sup> order integration with fixed interval
- ❖ **MILNE**: variable step, 5<sup>th</sup> order predictor corrector integration
- ❖ **RKS**: Runge-Kutta 4<sup>th</sup> order with variable integration interval
- ❖ **RKSFX**: Runge-Kutta 4<sup>th</sup> order with fixed integration interval
- ❖ **SIM**: Simpsons integration method with fixed integration interval

# Simulation Systems (1)

---



- Simulation models are mostly either language or platform dependent, which can be:
- Block (Schematic) Oriented Simulation Systems
- Equation Oriented Simulation Systems
- General-Purpose Simulation Systems
- Component-Based Simulation Systems
- High-Performance Simulation Systems
- Combined Simulation
- Integrated Circuits Simulation Systems
- Parallel Simulation

## Simulation Systems (2)

---



- Besides the previously introduced block-, equation and general purpose simulation systems another trend in simulation software are component-based systems (CBS)
- CBS provide a wide range of model parts – the components – which can be used for model building.
- Model building first involves constructing a diagram on the screen that represents the various model parts.
- This diagram is composed of a series of components, each of which are intended for a different type of mathematical operation.
- Each component has a definition that can be edited to insert its equation and any other information.

## Simulation Systems (3)

---



- Using the component-based approach, first one can visualize the system and then built the model.
- The user can rationalize the results using analysis methods like optimization, minimization, Monte Carlo and sensitivity analysis.
- Monte Carlo analysis enables model parameters to be specified as random distributions, while a model runs for a specified number of times, and during each run the value of the parameter is taken from a specified probability distribution.
- Hence, the effect of varying parameters of a component value can be observed.

## Simulation Systems (4)

---



- Another trend in simulation software toward are high-performance languages applicable for technical computing.
- These simulation software languages are mathematically and graphical-based combining the model parts using special scripts.
- Scripts are ASCII text files describing a sequence of statements and functions that can be saved and used as desired without having to recreate them each time they are needed.
- They are useful for automating series of simulation-language commands, such as computations that have to be performed repeatedly from the command line.
- Scripts operate on existing data in the workspace, or they can create new data on which to operate.

## Simulation Systems (5)

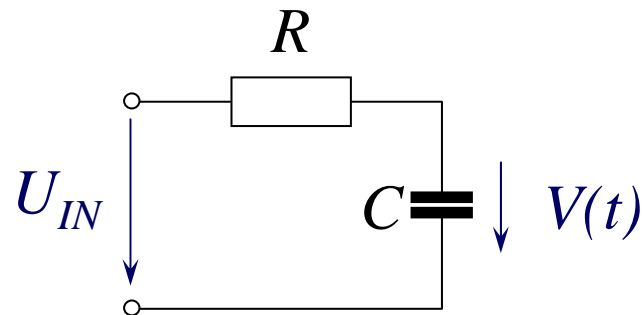
---



- Any variables that scripts create remain in the workspace after the script finishes so they can be used for further computations.
- A typical representative of a script-file-based simulation software is MATLAB, which is a high-performance software for technical computing that integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in the mathematical notation.
- In MATLAB scripts are identified as M files, which contain the sequence of commands that are ordinarily processed following the command prompt.
- Several types of M files are used with MATLAB.
- The special toolboxes available for MATLAB are in fact made up of M files, developed for the particular application.

## Simulation Examples: First order RC network (1)

- Consider a first order RC network. The voltage while charging the capacitor is described by  $V(t)$



$$V(t) = \left( 1 - e^{-\left(\frac{1}{R \cdot C}\right) \cdot t} \right)$$

where the  $t$  denotes the time and  $TC = R \cdot C$  is the time constant. ( $R$  is the resistance and  $C$  the capacitance)

- We plot the output of the RC circuit
  - time constant of 0.5 seconds
  - time range 0 to 3 seconds
- The corresponding command sequence in MATLAB is shown on the next slide

## Simulation Examples: First order RC network (2)



- MATLAB-Script for a First order RC network:

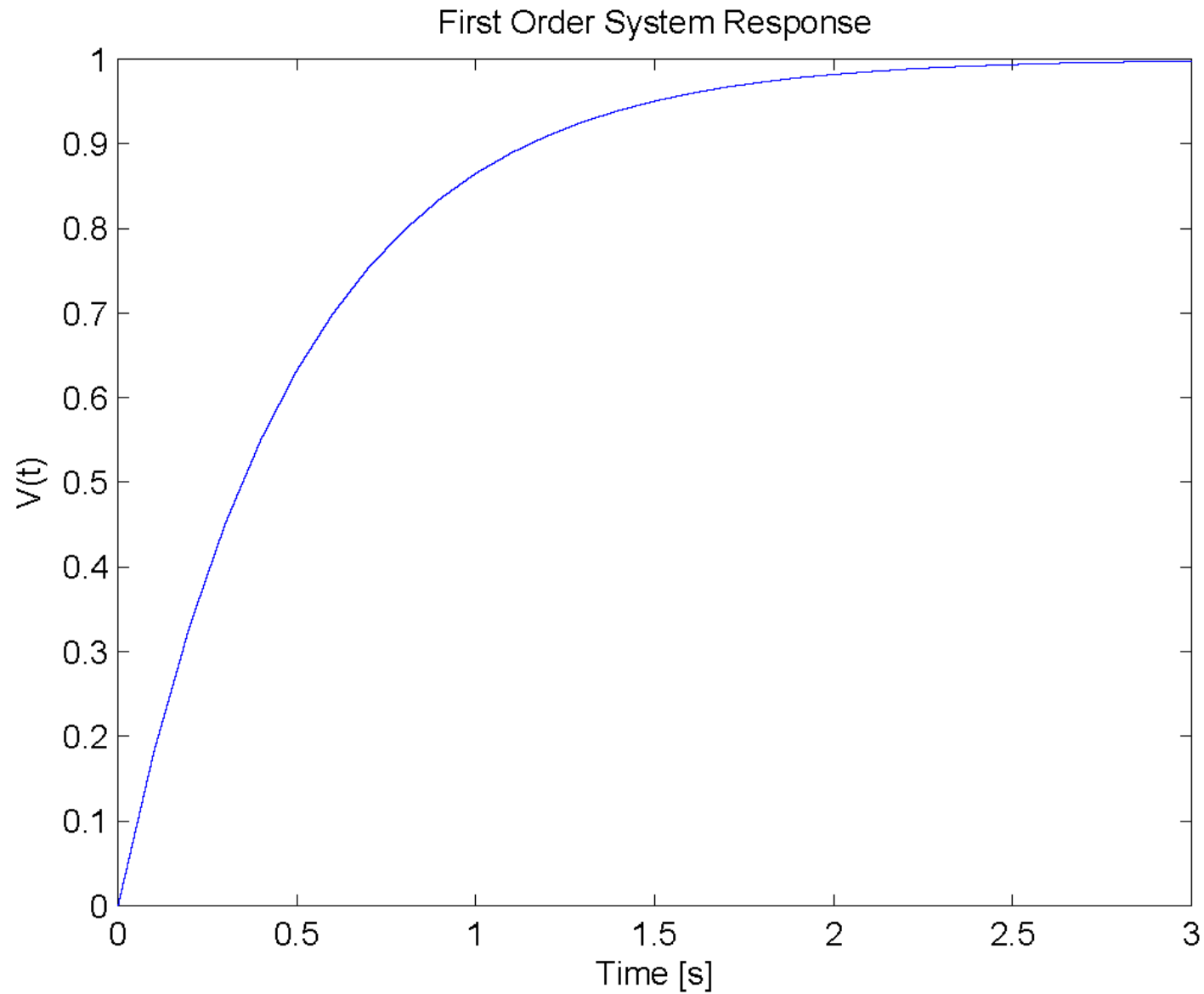
```
% time constant
TC = 0.5;

% time range (from 0s in steps of 0.1s to 3s)
t = [0:0.1:3];

% system-output
V = (1-exp(-t/TC));

% graphics
plot(t, V);
title('First Order System Response');
xlabel('Time');
xlabel('Time [s]');
ylabel('V(t)');
```

# Simulation Examples: First order RC network (3)



## Simulation Examples: B2SpiceA/D V4 (1)



- Traditionally, electronic circuit design was verified by building prototypes, subjecting the circuit to various stimuli, such as input signals, temperature changes, power-supply variations, etc., and then measuring its response using appropriate laboratory equipment.
- Prototype building is somewhat time consuming, but it produces practical experience from which to judge the manufacturability of the design.
- Computer programs that simulate the performance of an electronic circuit provide a simple, cost-effective means of confirming the intended operation prior to circuit construction, and of verifying new ideas that could lead to improved circuit performance.

## Simulation Examples: B2SpiceA/D V4 (2)



- Berkeley's Spice and the Georgia Techs Xspice simulators are the classical ones in this domain.
- The B2SpiceA/DV4 circuit simulator helps to design analog, digital, and mixed-mode circuits.
- Rather than working on circuit design with physical components, which require expensive equipment and a lab, the B2SpiceA/DV4 allows one to perform realistic simulations on circuits without clipping wires or splashing solder.
- Editing and simulating circuits is a quick and easy procedure. The current B2SpiceA/DV4 information for the latest patches and more can be found on the web site at *www.beigebag.com*.

## Simulation Examples: B2SpiceA/D V4 (3)

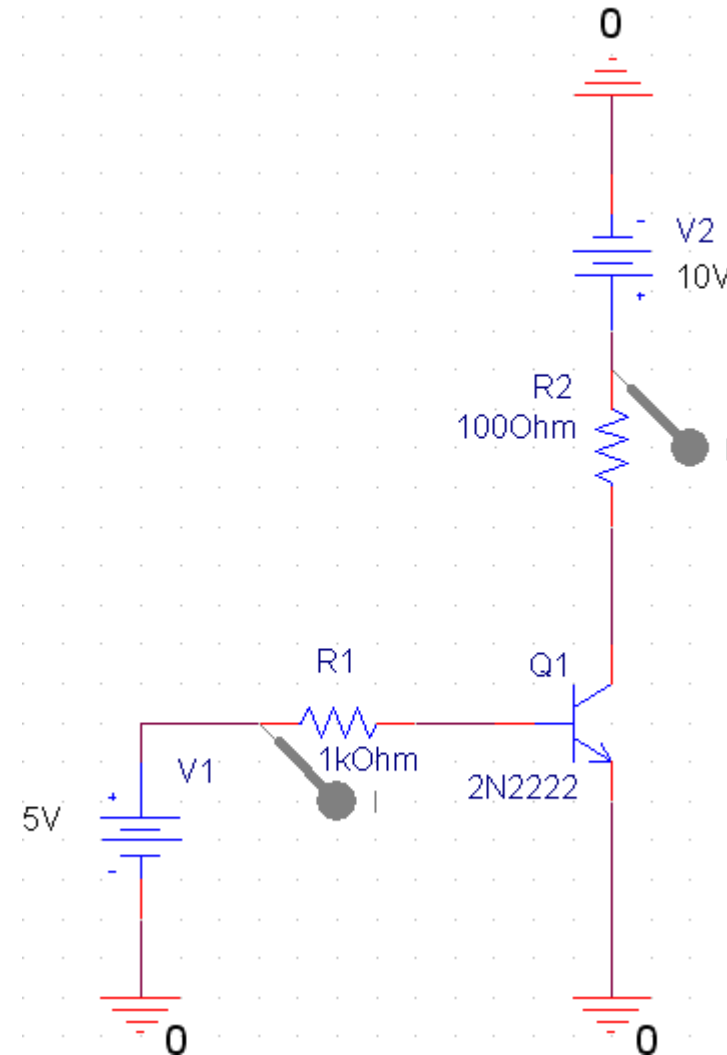


- The schematic B2SpiceA/D V4 editor allows one to enter a circuit design.
- When building a new circuit all parts will be added into the circuit by choosing them from menus and drawing wires to connect the devices, and setting the properties for the devices to customize their behavior.
- Building a simple resistor-transistor-logic (RTL) inverter circuit is as follows:
  - Place the devices
  - Customize the transistor
  - Set the device properties
  - Simulate the circuits AC and DC transfer curve
  - Simulate the circuits transient behavior

## Simulation Examples: B2SpiceA/D V4 (4)



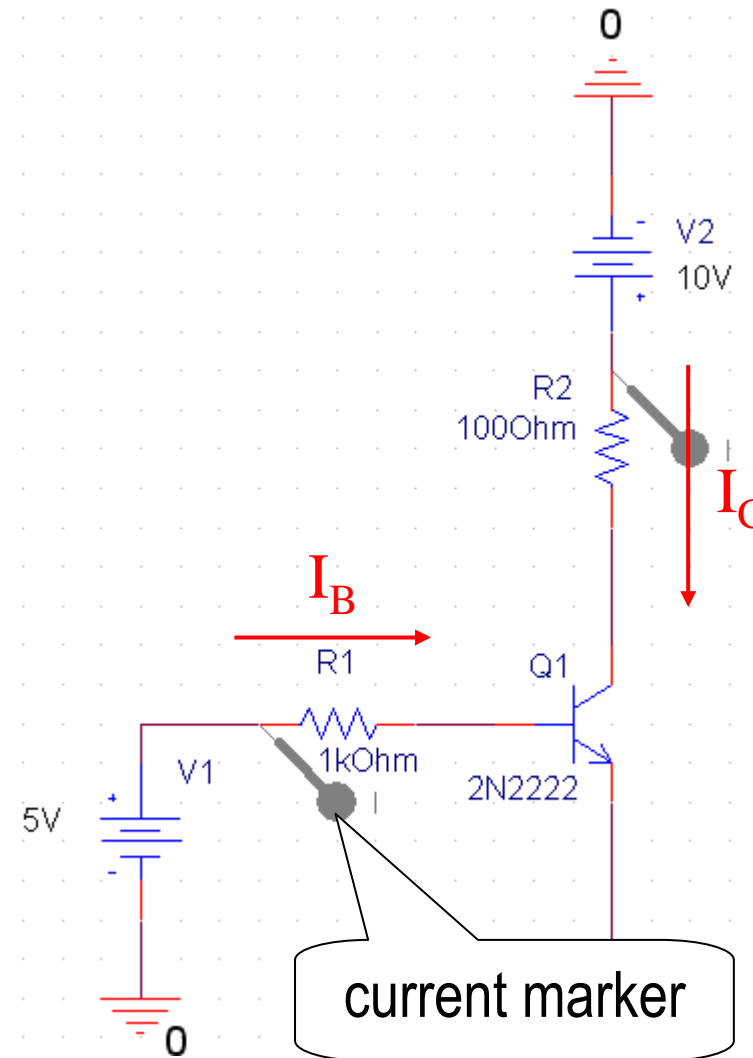
- Bipolar transistor was introduced as an amplifier and an inverter
- Transistor: controllable resistor.
- If we open the base-emitter junction (that means, we apply a voltage to the base above the threshold voltage), a current can flow from the collector to the emitter.



## Simulation Examples: B2SpiceA/D V4 (5)



- This means: We can control the collector current  $I_C$  (from the emitter to the collector) with the base current  $I_B$
- The currents can be adjusted by the choice of the base resistor (in the picture R1) and the collector resistor (in the picture R2) for given values of the two voltage sources V1 and V2
- The higher the resistance is, the smaller the current gets.



## Simulation Examples: B2SpiceA/D V4 (6)

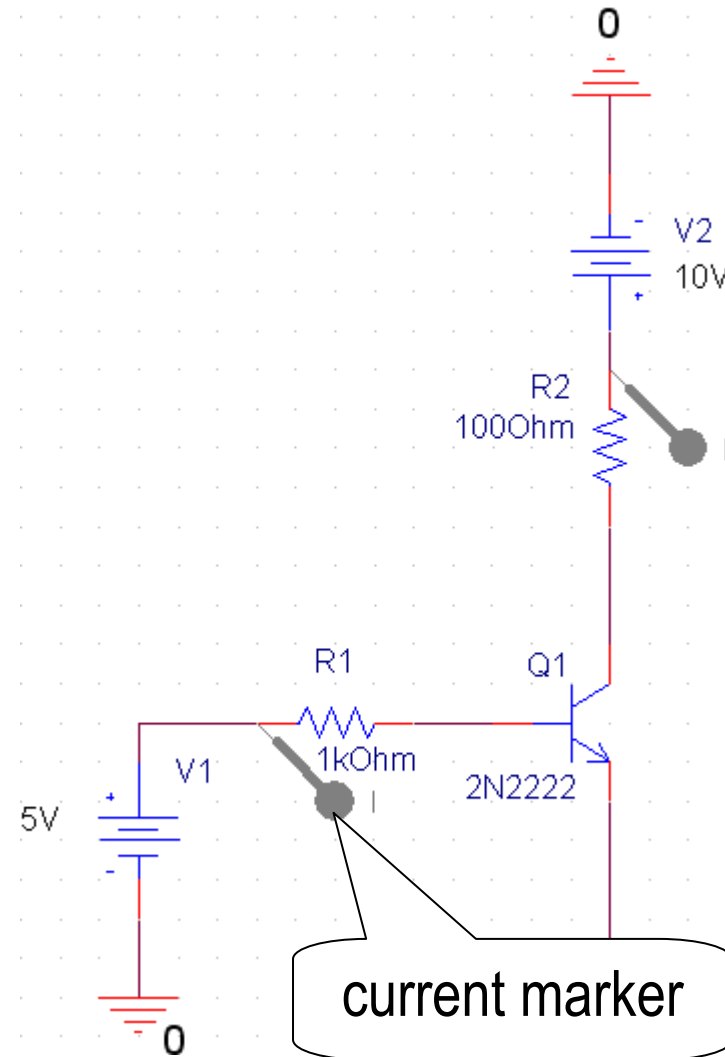


1. Perform a transient analysis (i.e. simulate the circuit's behavior over time.)

- What are the currents you measure?
- Of what base- and collector-current depend?

2. Change the values of R1 and R2 for fixed values of V1 and V2.

- What do you see?



# Simulation Examples: B2SpiceA/D V4 (7)



The screenshot shows the OrCAD Capture CIS interface. The main window displays a circuit schematic with a 5V DC source (V1), a 1k resistor (R1), a 2N2222 transistor (Q1), and a 100 Ohm resistor (R2). A callout box points to the File menu icon in the toolbar, containing the text: "open transi stor. opj (to be found under Ei gene Datei en\exampl es\transi stor\)". To the right of the schematic, a list of transistor parameters is displayed:

- VAR = 500
- ISC = 0.0
- NC = 1.0
- RB = 0.676
- RBM = 0.676
- RE = 0.1
- RC = 0.654
- CJE = 22.25E-12
- VJE = 1.333
- MJE = 0.522
- TF = 454.4E-12
- XTF = 13.24
- VTF = 4.83
- ITF = 216.3E-3
- PTF = 0.0
- CJC = 8.37E-12
- VJC = 1.333
- MJC = 0.518
- XCJC = 0.5
- TR = 117.5E-9

The status bar at the bottom indicates "0 items selected" and "Scale=200% X=4.70 Y=0.60". The Windows taskbar shows the Start button and several open applications: OrCAD Capture CIS, exercise, 266. The Beatles - Go..., Unbenannt - Paint, and Microsoft PowerPoint.

# Simulation Examples: B2SpiceA/D V4 (8)



The screenshot displays the OrCAD Capture CIS interface. The main workspace shows a circuit schematic with a 5V DC source (V1), a 1kOhm resistor (R1), a 2N2222 transistor (Q1), and a 100Ohm resistor (R2). A red circle highlights the 'Run PSpice' button in the top toolbar. A callout box points to this button with the text: "Press the button 'Run PSpice'".

Simulation parameters listed on the right side of the workspace:

- IS = 166.78E-15
- BF = 150
- NF = 1.074
- VAF = 78
- IKF = 0.5
- ISE = 3.92E-12
- NE = 1.776
- BR = 2.394
- NR = 1.074
- RC = 500
- PTF = 0.0
- CJC = 8.37E-12
- VJC = 1.333
- MJC = 0.518
- XCJC = 0.5
- TR = 117.5E-9
- CJS = 0
- VJS = 0.7
- MJS = 0.5
- XTB = 2.34
- EG = 1.11
- XTI = 3.0

The status bar at the bottom indicates "Ready", "0 items selected", and "Scale=196% X=3.30 Y=2.30". The Windows taskbar shows the Start button and several open applications: "Posteingang - Thunder...", "exercise", "spice", "OrCAD Capture CIS - ...", "Microsoft PowerPoint - ...", "HWV - Mehr als ein Ziel - ...", "Unbenannt - Paint", and "SCHEMATIC1-transisto...". The system clock shows "10:06 PM".

# Capability and Limitations of Devices Simulators (1)

---



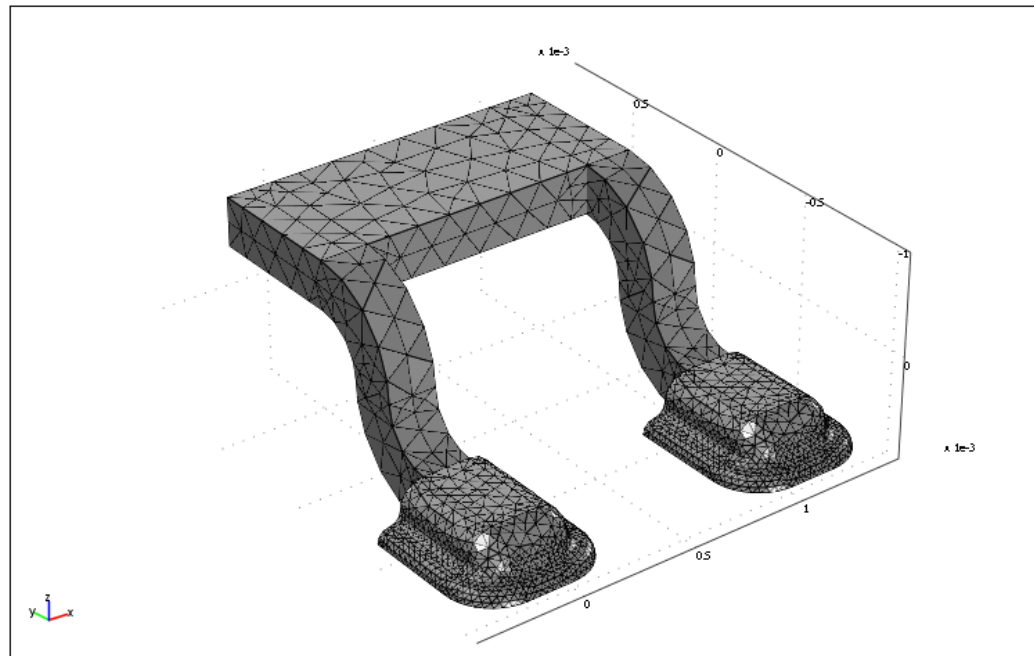
- Let us start by defining and solving a simple model example that treats current conduction and heat generation in an aluminum film deposited on a silicon substrate with the COMSOL multiphysics program
- Purpose of the model is to investigate the temperature in the device after the current is applied
- Important features:
  - use of predefined physics
  - definition of a multi physics problem
  - expression feature to define physical properties that depend on the solution itself
  - extraction of design numbers using post-processing
  - parameter solving provides fast and efficient parameter screening



# Capability and Limitations of Devices Simulators (3)



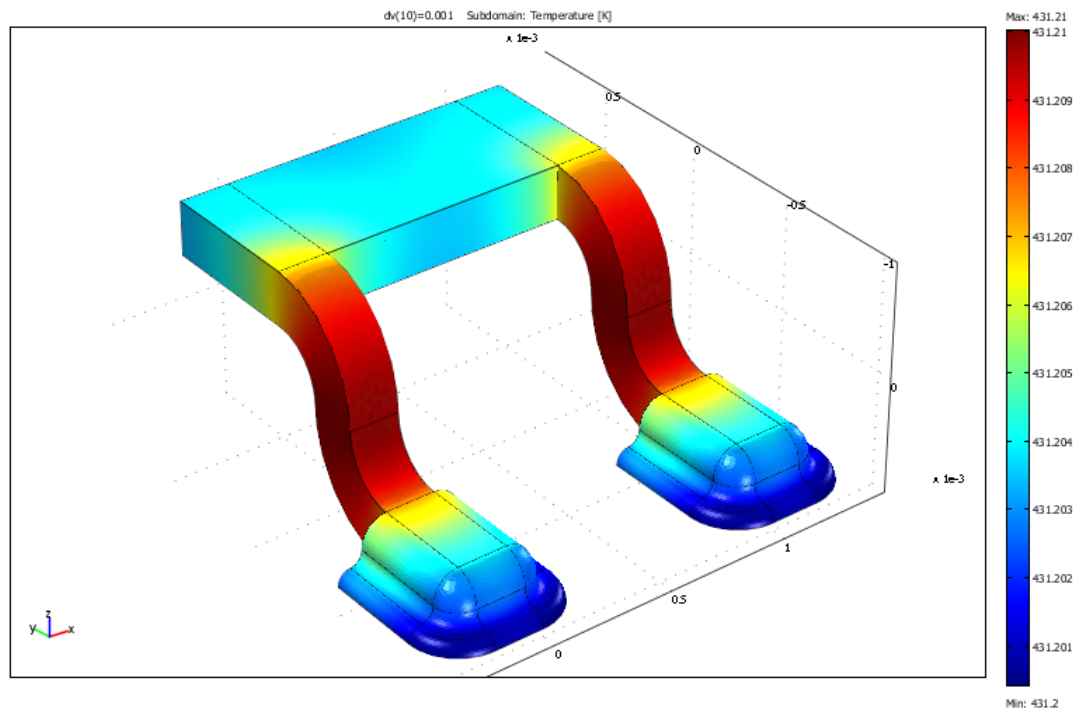
- Modeling steps:
  - Meshing



# Capability and Limitations of Devices Simulators (4)



- Modeling steps:
  - Solving



- Post-processing
- Parametric studies

# Capability and Limitations of Devices Simulators (5)

---



- Most device simulation software use
  - finite difference method (FDM)
  - finite element methods (FEM)for solving the basic semiconductor equations (see lecture 6.1)
- [..\FE\fsi\\_trachea.avi](#)
- In the finite difference method, derivatives of the semiconductor equations are replaced by finite difference formulas at each node in a finite difference mesh
- It is important to employ small mesh spacing in the vicinity of junctions where the potential is changing rapidly
- The boundary conditions are normally fixed values at the contacts and gradient fixed at the other surfaces.
- The system of equations may be solved by direct methods or iterative schemes

# Capability and Limitations of Devices Simulators (6)

---



- Device simulators are powerful software tools
- Device simulators can model
  - 2D or 3D distribution of potential and carrier concentration
  - current vectorsin order to predict its electrical characteristics for any bias conditions
- Device simulators are useful not only for majority carrier devices involving a single carrier type, such as MOSFETs, JFETs, and MISFETs, but also for minority carrier devices involving both carriers, such as  $p$ - $n$  junction diodes,  $n$ - $p$ - $n$  BJT and  $p$ - $n$ - $p$  BJT
- Device simulators can also be used to study devices under steady-state, small signal, and transient operating conditions

# Capability and Limitations of Devices Simulators (7)

---

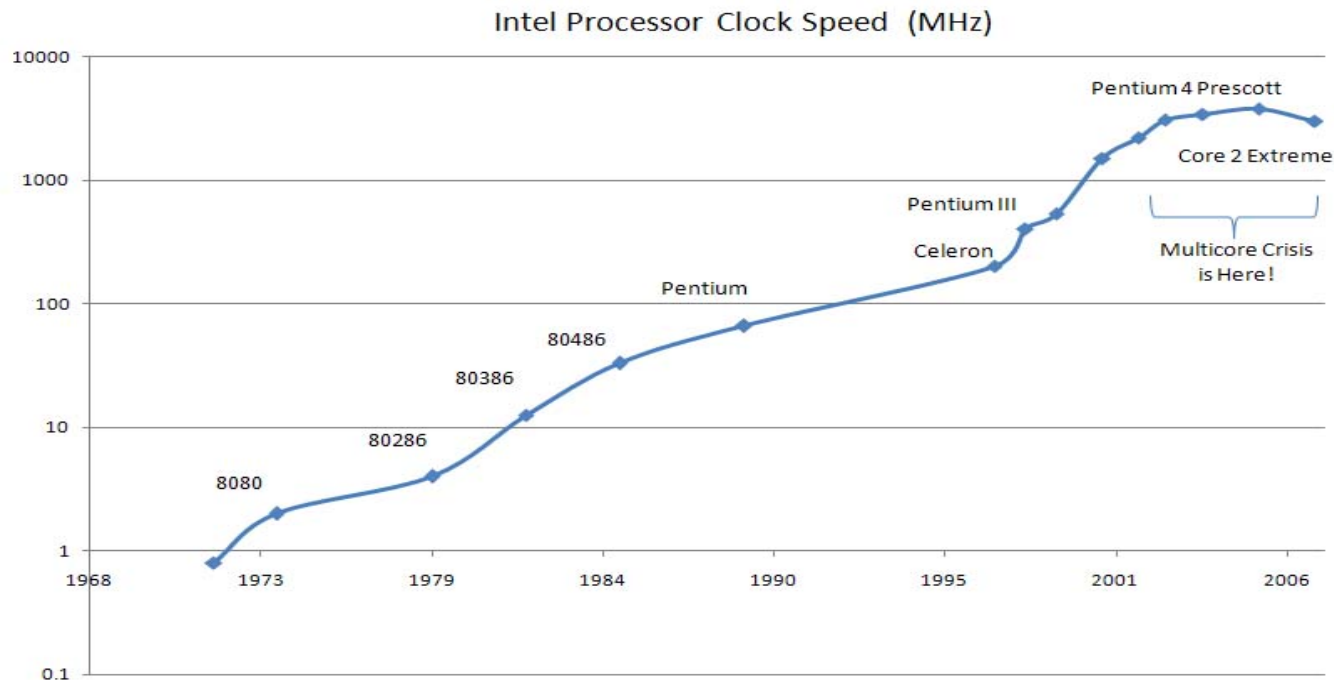


- Capabilities of device simulators are different. In general the potential and carrier concentration distributions can be evaluated, also a variety of electrical parameters' could be derived; they are:
  - Layer conductance's
  - Layer resistances
  - Low Frequency capacitance
  - Flat –band voltage
  - Threshold voltage
  - Large signal current gain
  - Small signal current gain
  - Cutoff frequency
  - Etc.

# Parallel Paradigm (1)



- Why a parallel paradigm ?
  - “The free lunch is over.” - Herb Sutter
    - A few years ago: The same program executed faster on new processors with higher clock rate.
    - Today: Clock speeds no longer increasing exponentially:

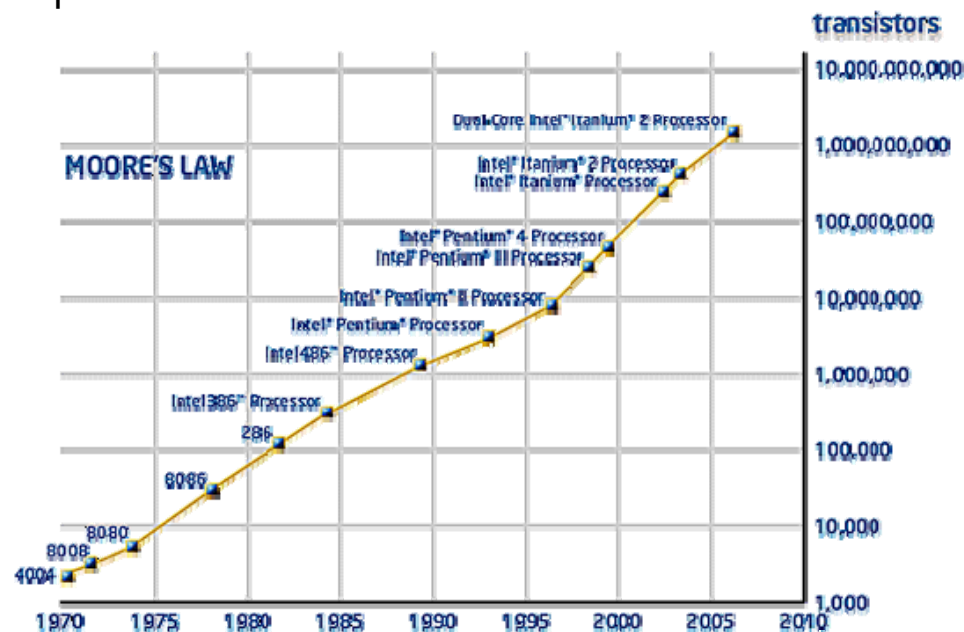


- Problems caused by higher speeds
  - Excessive power consumption
  - Heat dissipation
  - Current leakage

## Parallel Paradigm (2)



- Why a parallel paradigm?
  - Moore's law is still in effect
    - Although „more transistors“ doesn't mean „higher clock rate“ any more
    - The transistors are now spent on multiple processor cores, capable of running in parallel – essentially many processors one chip
    - Multi-core processors are now mainstream



- Performance increase has to come from parallelism!
  - Distribute the computational load across the processor cores
  - Try to keep all processor cores busy doing useful work

# Parallel Paradigm (3)

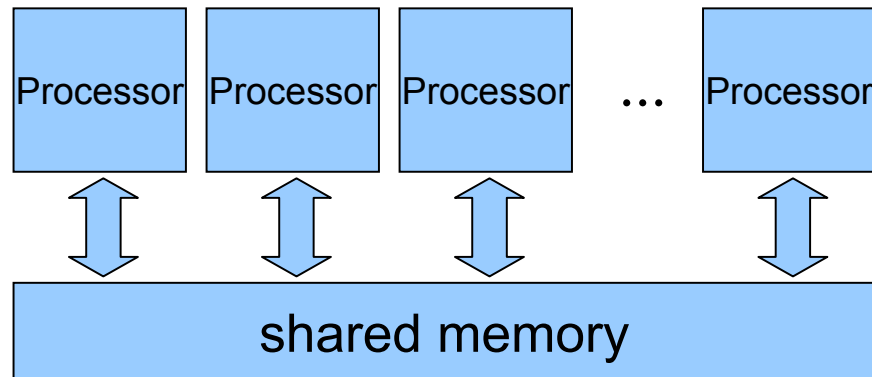


- Parallel computer taxonomy by Flynn, 1966:
  - SISD: „Single instruction, single data stream“
    - Classical model with a single uni-processor
    - Resembled by the von Neumann-architecture
  - SIMD: „Single instruction, multiple data stream“
    - One CPU instruction is applied to data from multiple data streams („vector processors“)
      - e.g.: Graphic Processing Units (GPU) on graphics cards
      - e.g.: MMX and SSE instruction set extensions developed by Intel, 3DNow! by AMD
  - MISD: „Multiple instruction, single data stream“
    - (Not commonly used...)
  - MIMD: „Multiple instruction, multiple data stream“
    - Multiple autonomous processors execute different instructions on different data
    - Two important subtypes (next slide):
      - a) *shared-memory systems*
      - b) *distributed-memory systems*

## Parallel Paradigm (4)



### ■ a) Shared-Memory Systems



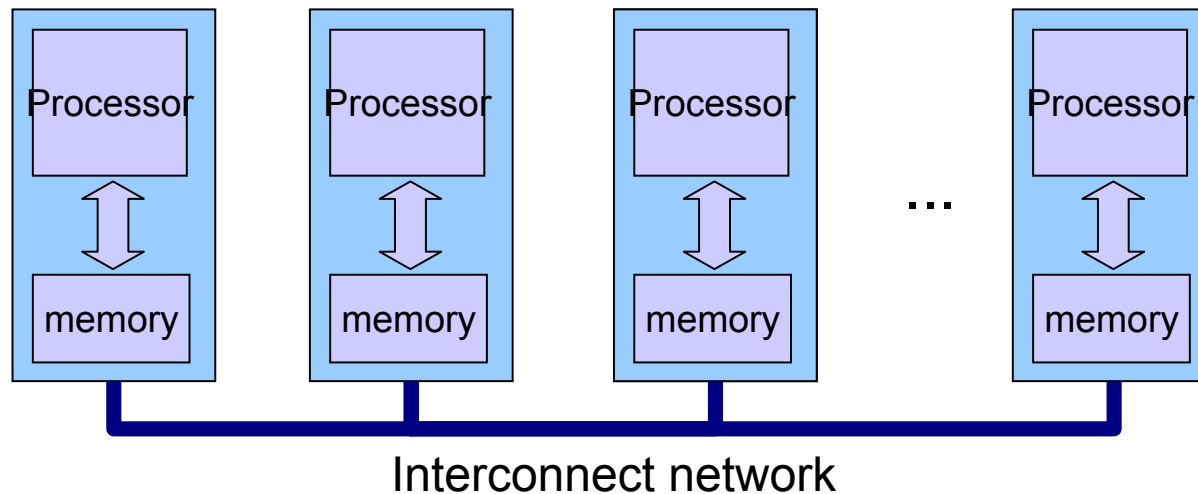
- Many processing units operate on a shared memory
- Most common programming model:
  - Thread programming model
- Example: Intel QuadCore CPU
  - 4 processing units (cores) on one CPU
  - Shared memory: System RAM
  - „Symmetric multiprocessing“ (SMP): no processing unit is privileged



## Parallel Paradigm (5)



### ■ b) Distributed-Memory Systems



- Cluster of many separate computers connected by a communication network
- 410 of the supercomputers in the TOP500 list of supercomputers are of this type (as of 06/2009) [1]
- Programming model:
  - One or more process per node
  - Communication via message passing over the network

[1] TOP500 computer sites - [www.top500.org](http://www.top500.org)

# Parallel Paradigm (6)



- IBM „Roadrunner“, #1 in the TOP500
  - Cluster of clusters of nodes
  - Combination of shared- and distributed memory systems

Roadrunner is a supercomputer built by IBM at the Los Alamos National Laboratory in New Mexico, USA. Currently the world's fastest computer, the US\$133-million Roadrunner is designed for a peak performance of 1.7 petaflops, achieving 1.026 on May 25, 2008, and to be the world's first TOP500 Linpack sustained 1.0 petaflops system. It is a one-of-a-kind supercomputer, built from off the shelf parts, with many novel design features.

In November 2008, it reached a top performance of 1.456 petaflops, retaining its top spot in the TOP500 list.

([http://en.wikipedia.org/wiki/Ibm\\_roadrunner](http://en.wikipedia.org/wiki/Ibm_roadrunner))



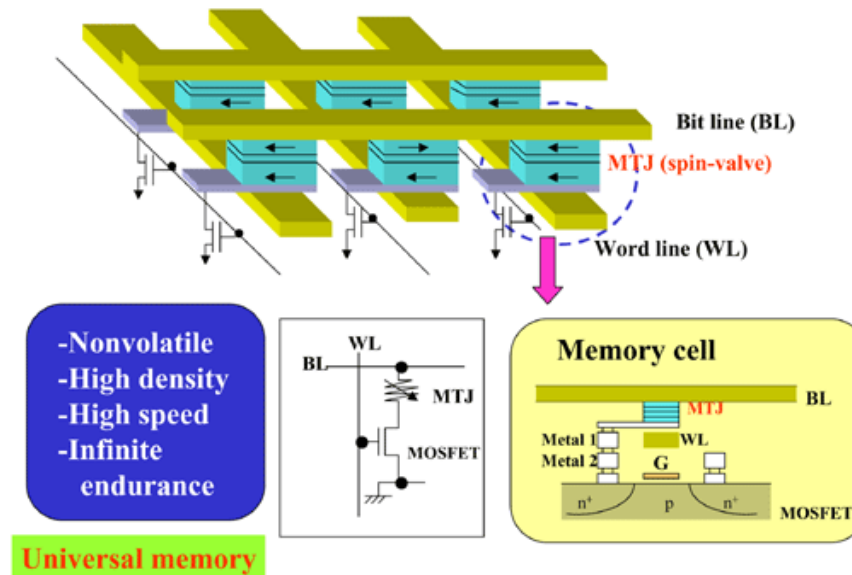
# Parallel Paradigm (7)



- Standards for parallel program development
  - Shared-Memory Systems:
    - *OpenMP* (Open Multi-Processing)
      - Programming language extensions for C/C++ and FORTRAN, which introduce parallel programming constructs, e.g. parallel for-loops
      - open standard jointly developed by a group of major hardware and software vendors
    - *POSIX threads* (Pthreads)
      - Application programming Interface (API) for the C programming language
      - Introduces functions and data types to create, manage and synchronize *threads*
      - IEEE POSIX 1003.1c standard from 1995
    - many others...
  - Distributed-Memory Systems:
    - *MPI* (Message Passing Interface)
      - Provides an communication protocol and an application programming interface (API) to send messages between compute nodes over a network
      - today's dominant model used in high-performance computing (HPC)
      - open standard backed by major hardware and software vendors
    - many others...

# Parallel Paradigm (8)

- Example: Non-volatile storage devices: Magnetic random access memory (MRAM)
- Storage cells:
  - Spin valve / Magnetic Tunnel Junction (MTJ)

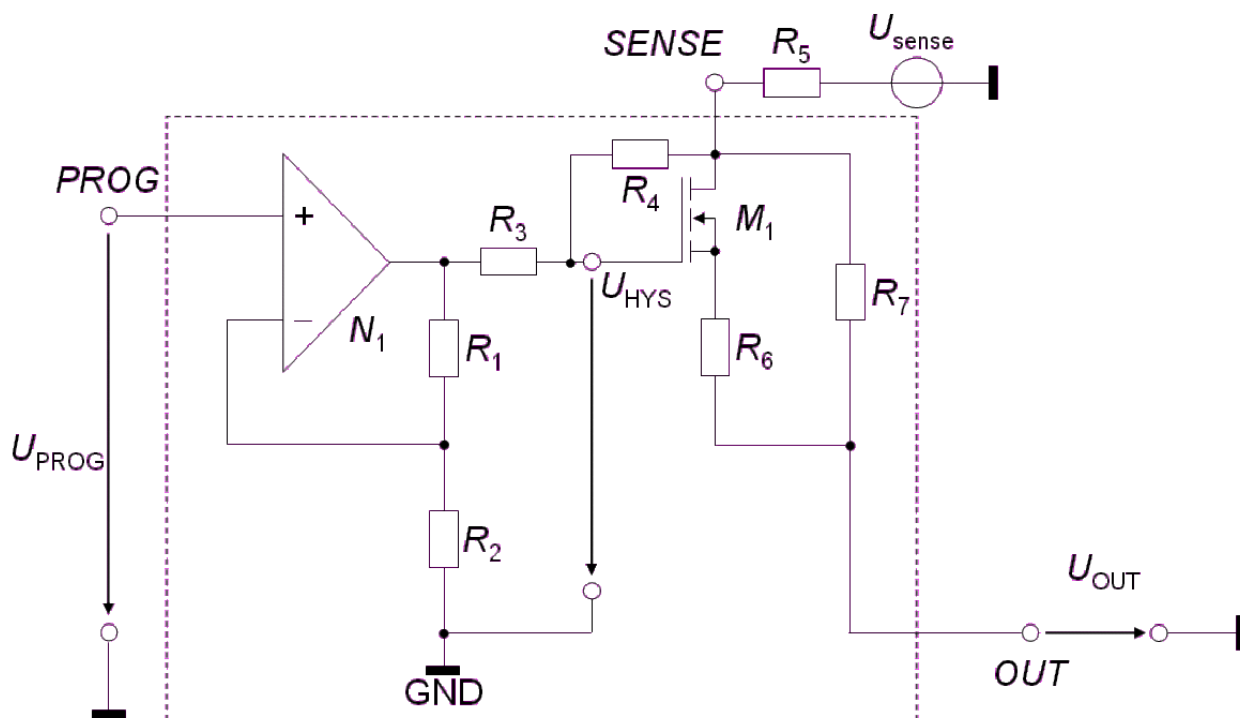


- Bit representation:
  - Orientation of the magnetization in ferromagnetic layers
  - Read process: Exploit GMR effect
  - Write process: Use magnetic fields, electrical currents

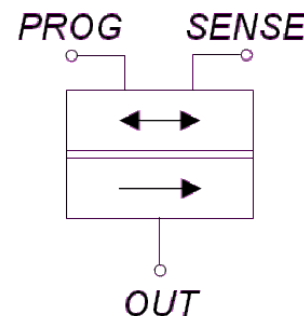
# Parallel Paradigm (9)



## Equivalent



## Spin valve



Simulation using an equivalent circuit

- many electronic components

Micromagnetic simulation

- single ferromagnetic element

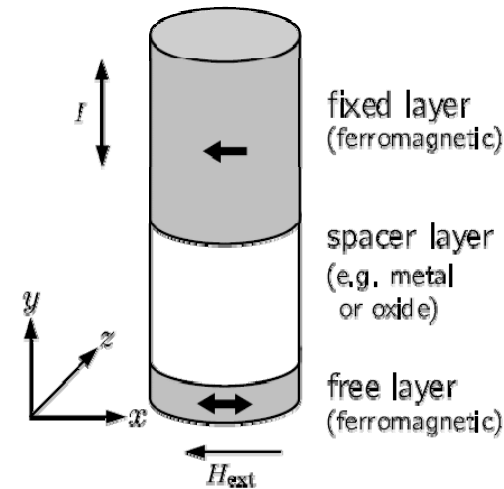
# Parallel Paradigm (10)



- Extended Landau Lifshitz Gilbert equation (LLGE)

$$\frac{d\vec{M}}{dt} = -\gamma\vec{M} \times \vec{H}_{\text{eff}} + \frac{\alpha}{M_s} \times \frac{d\vec{M}}{dt} - \frac{\gamma a_j j}{M_s} \vec{M} \times (\vec{M} \times \vec{P})$$

- $\alpha$ : damping constant
- $\gamma$ : gyromagnetic ratio
- $M_s$ : saturation magnetization
- $P$ : spin polarization
- $j$ : current density
- $a$ : coupling parameter between  $j$  and  $M$
- $H_{\text{eff}} = H_{\text{exchange}} + H_{\text{stray}} + H_{\text{aniso}}$



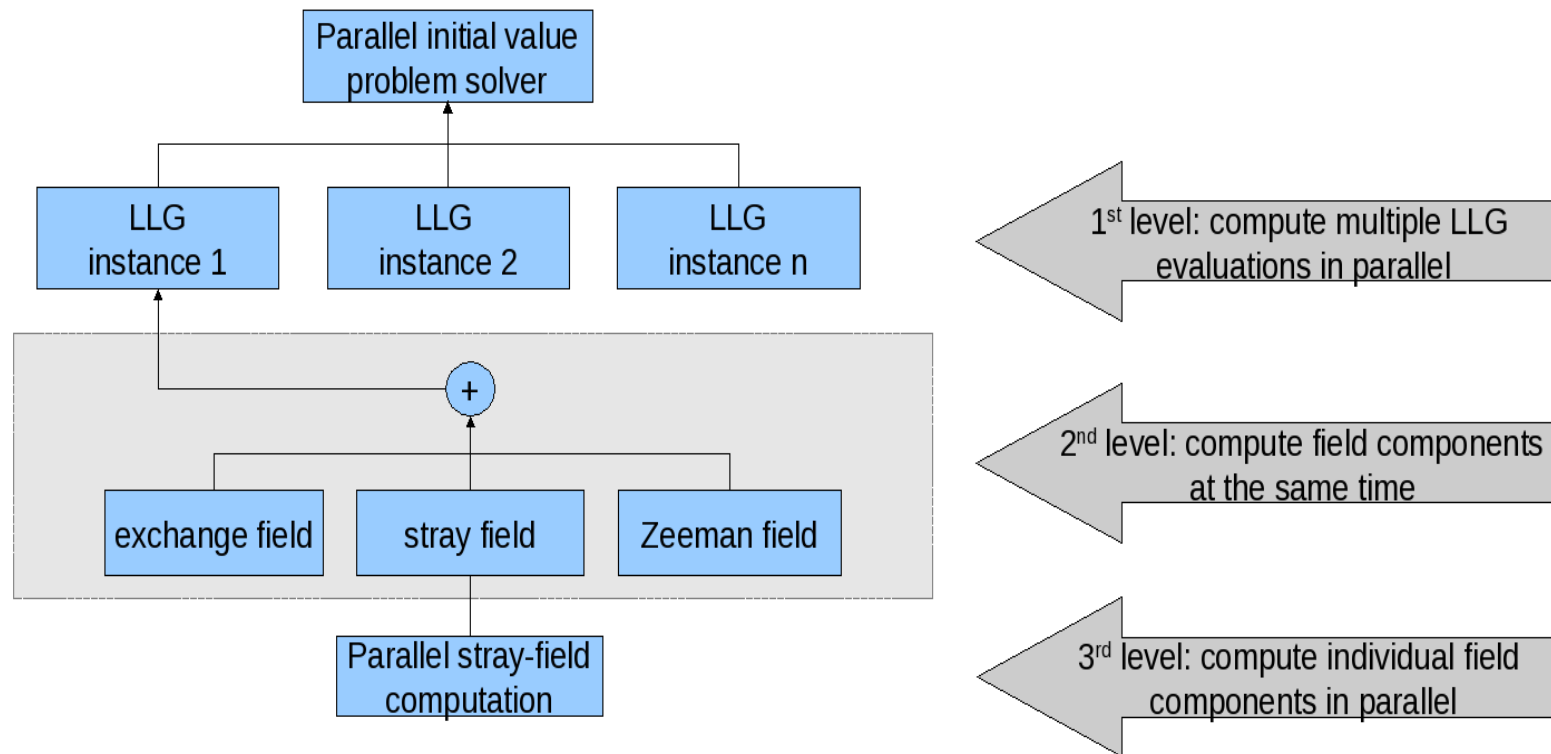
- Numerical methods:

- LLGE ← Runge-Kutta method (matrix operations)
- $H_{\text{exchange}}$  ← finite difference method (matrix operations)
- $H_{\text{stray}}$  ← convolution (matrix operations)
- $H_{\text{aniso}}$  ← scalar product (matrix operations)

# Parallel Paradigm (11)



- Parallelism simulation strategies: Possibilities at various levels:
  - First level: Use a parallel initial value problem solver
  - Second level: Compute LLGE terms, Slonchewski extension, and effective field terms parallel
  - Third level: Parallelize each individual field computation



# Download



- 
- <http://www.informatik.uni-hamburg.de/TIS>
  - Downloads
  - ISS 2009
  - Lecture 6.2

# References



- Arora, N. D., J. R. Hauser, D. J. Roulston (1982): IEEE Trans. Electron. Devices, ED-29, pp. 292ff
- Bardeen, J., W. H. Brittain (1948): Phys. Rev. 74, pp. 230ff
- Dziwior J., W. Schmid (1977): Appl. Phys. Lett. 31, pp. 346ff
- Möller, D. P. F. (2000): Rechnerstrukturen (in German), Springer Publ.
- Möller, D. P. F. (2002): Mathematical and Computational Modeling and Simulation, Springer Publ.
- Rouston, D. J., N. D. Arora, S. G. Chamberlain (1982): IEEE Trans. Electron. Devices, ED-29, pp. 284ff
- Swirhuhn, J. S. E., J. del Alano, R. M. Swanson (1986): IEEE Trans. Electron. Devices, ED-7, pp. 168ff
- Thompson S. E., F. A. Lindholm (1990): IEEE Trans. Electron. Devices, ED-37, pp. 2422ff
- Lagemann, K. (1987): *Rechnerstrukturen*. (in German) Springer-Verlag; Berlin. ISBN 3-540-17618-7
- Weste, N. H. E., K. Eshraghian (1994): *Principles of CMOS VLSI design: a systems perspective*. Addison-Wesley; Reading, MA. ISBN 0-201-53376-6
- Smotherman, M.: Computer Systems Organization. The Pentium Chronicles <http://www.cs.clemson.edu/~mark/330/p6.html>