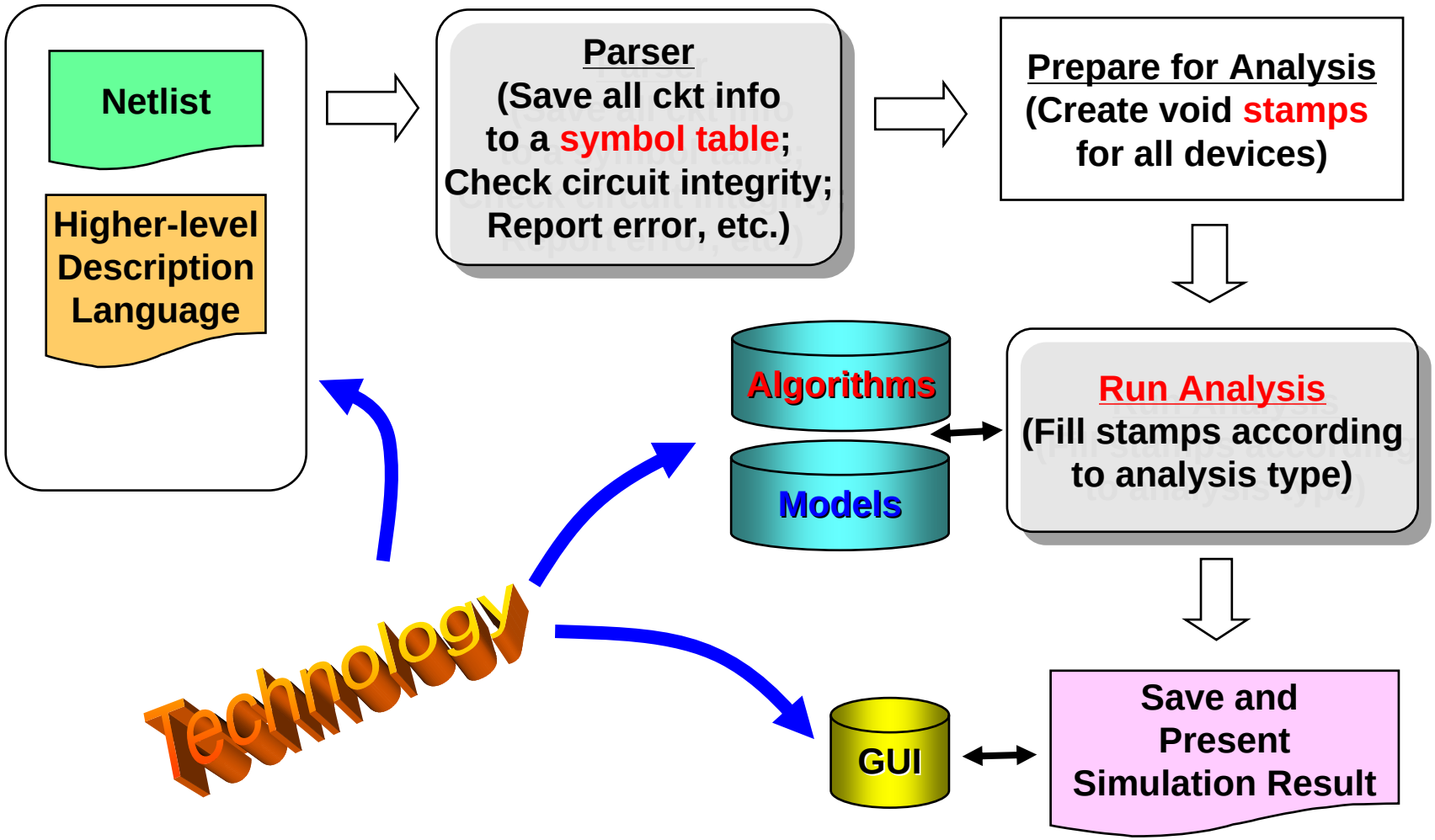
Lecture 6.
Simulator Architectural Design

Guoyong Shi, PhD
shiguoyong@ic.sjtu.edu.cn
School of Microelectronics
Shanghai Jiao Tong University
Spring 2010

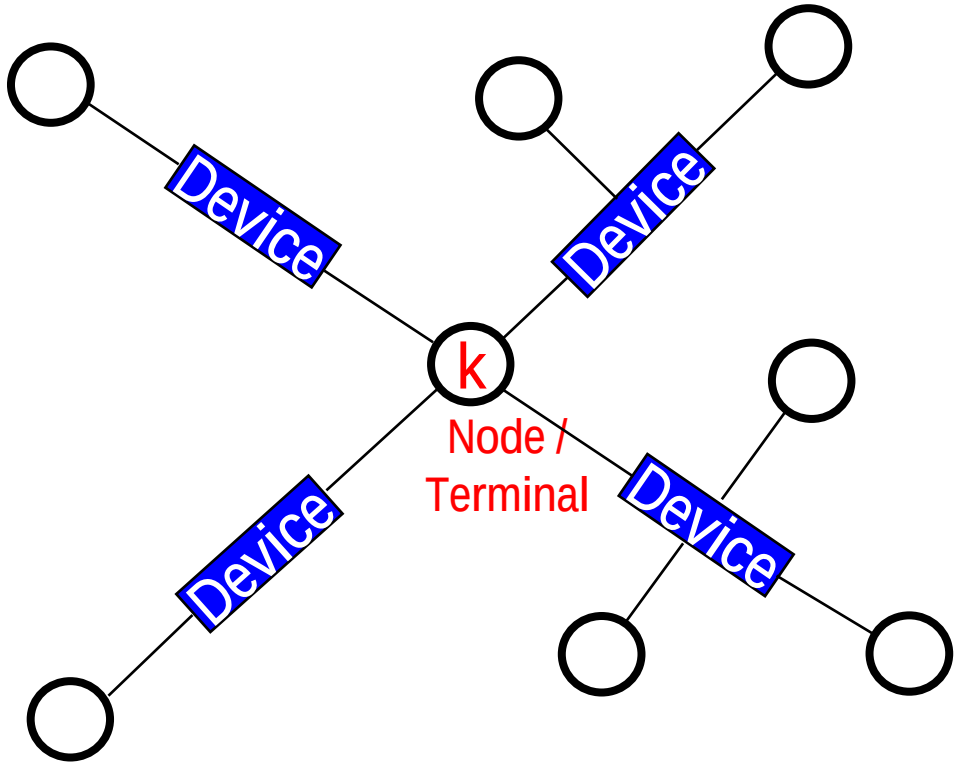
Outline

- **Basic constructs**
- **Model & Device Classes**
- **Device loading**
- **Object-oriented programming**
- **Model Compiler**

A Detailed Simulator Flow

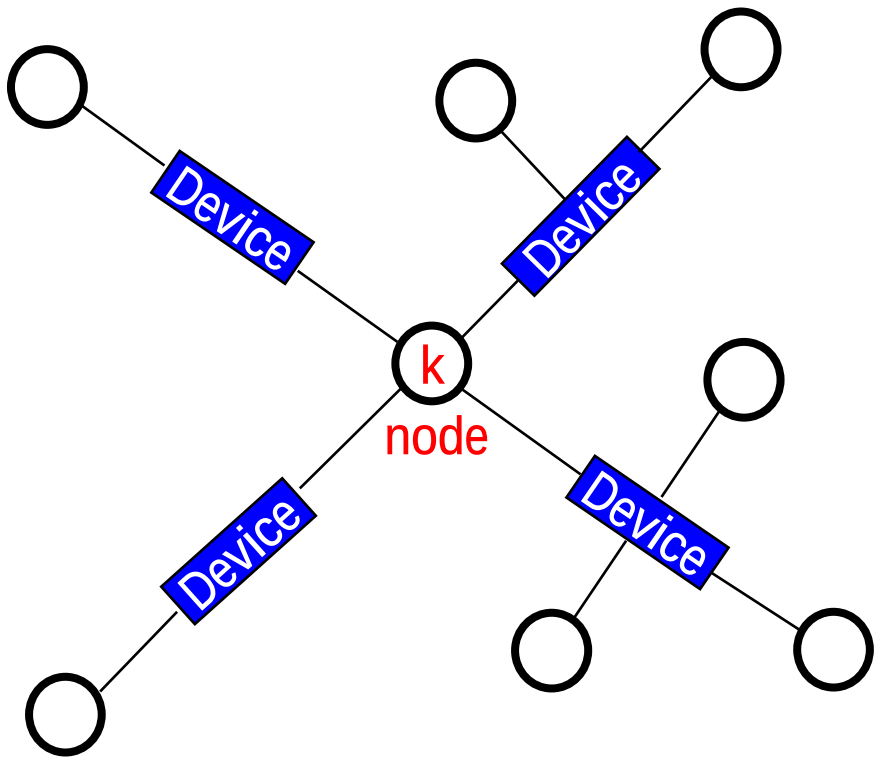


Node versus Terminal



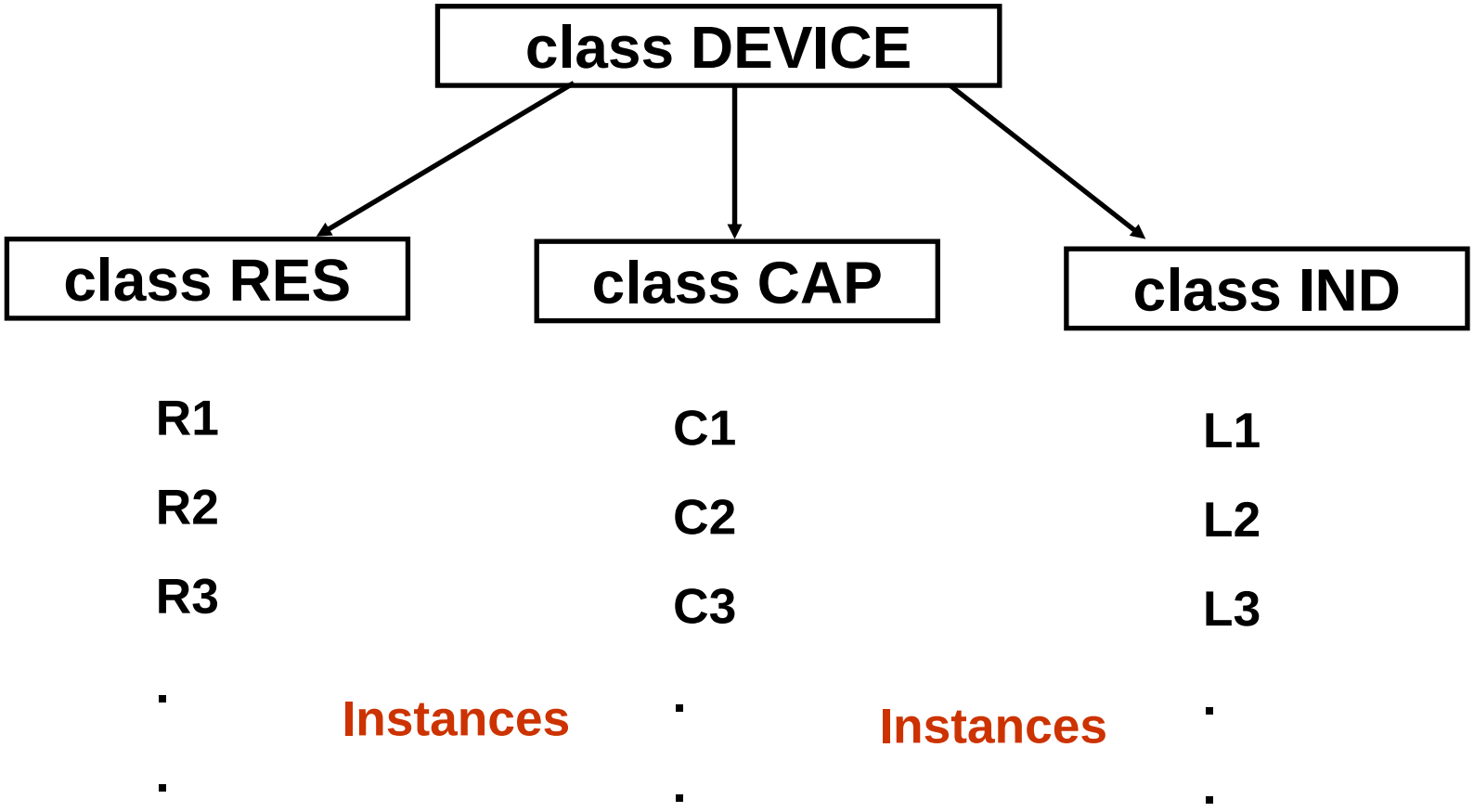
- "**Node**" is a circuit attribute:
  - Each **node** is connected to a list of **devices**.
- "**Terminal**" is a device attribute:
  - Each **device** has a list of **terminals**.
- A device **terminal** becomes a circuit **node** when it is connected in a circuit.
- Multiple device **terminals** may share the same **node**.
- Each device **terminal** identifies a **branch current**.

Node versus Device



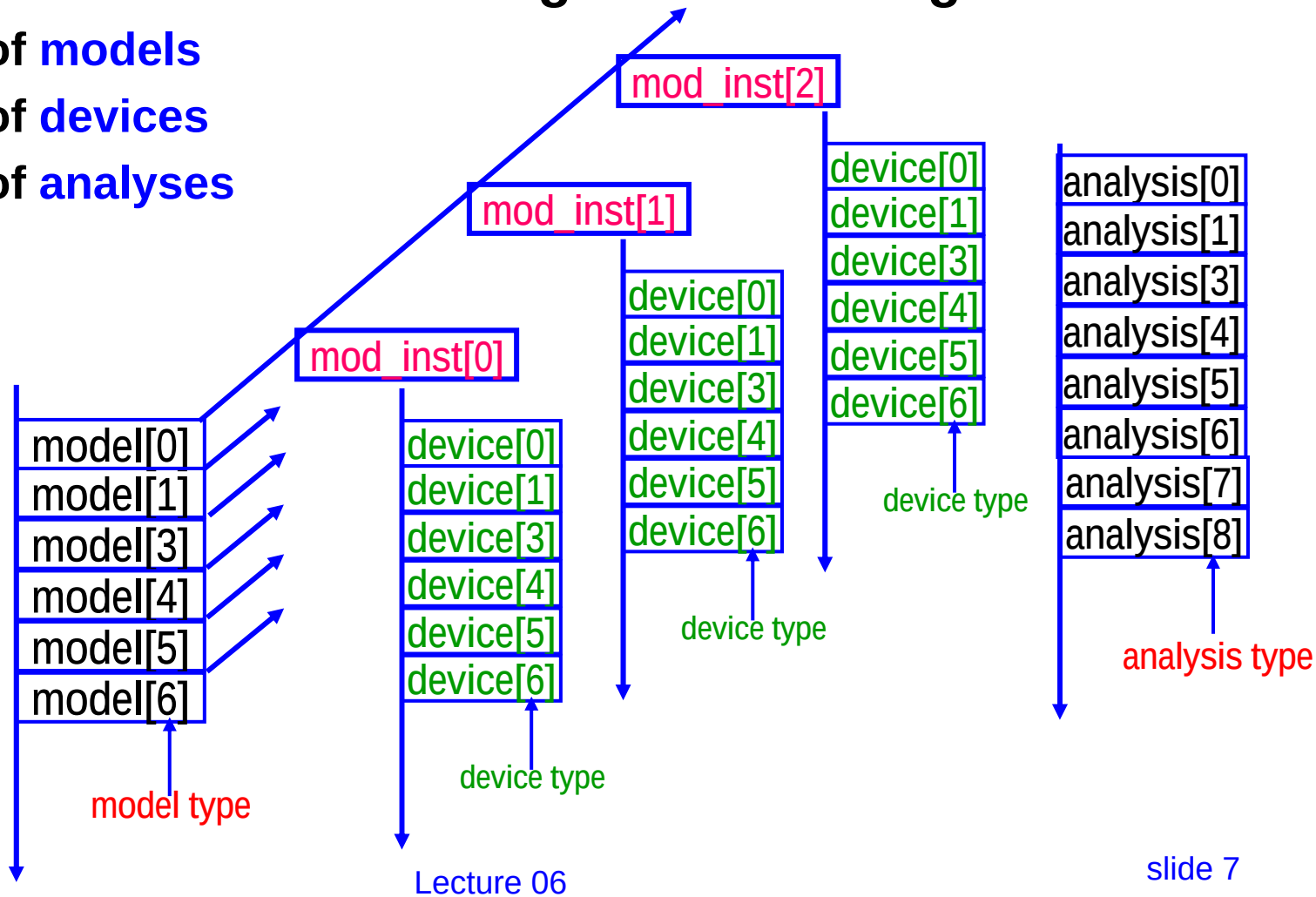
- Each **Node Object** manages **a list of devices** connected to it.
- Each **Device Object** manages **a list of nodes** as its terminals.
- Such information is directly used in the **MNA matrix stamping**.

Example

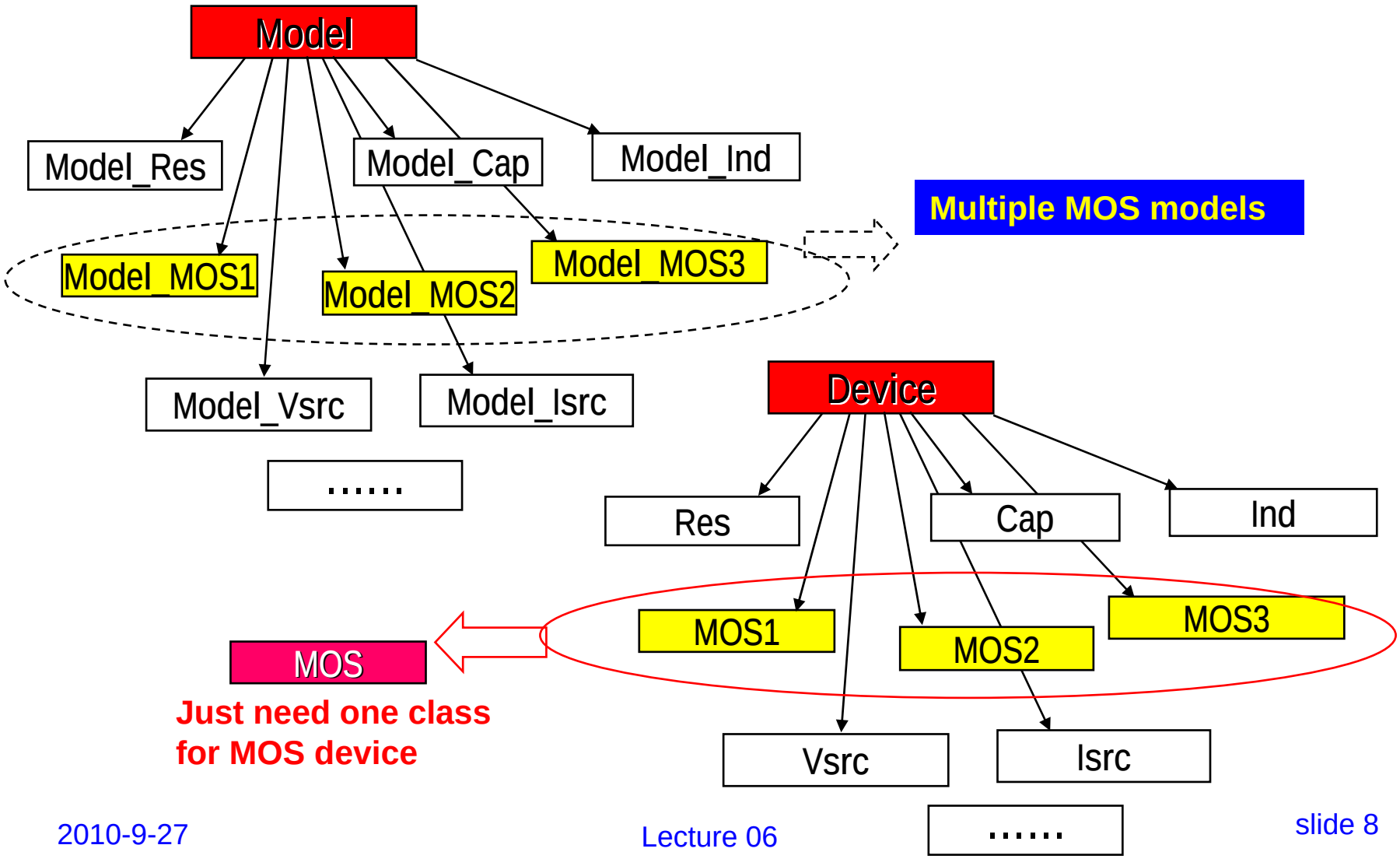

class DEVICE
class RES
class CAP
class IND
R1
R2
R3
.
.
Instances
C1
C2
C3
.
.
Instances
L1
L2
L3
.
.

Simulator Construction

- **The simulator should manage the following lists:**
  - **List of models**
  - **List of devices**
  - **List of analyses**



Model & Device Classes

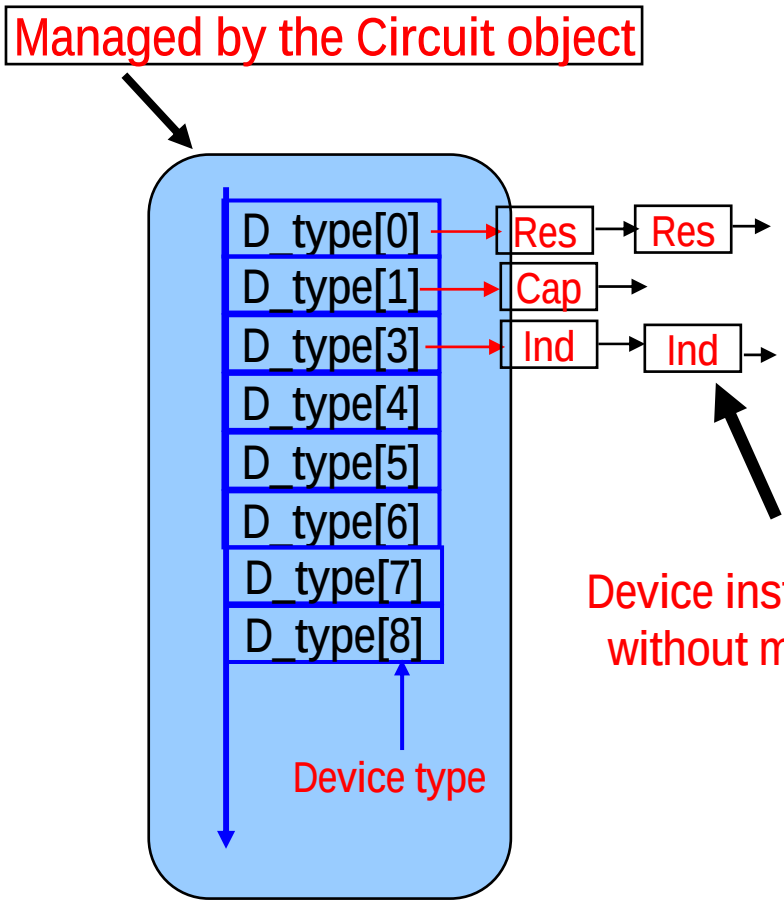


Device Instances

- **Device instances are divided into two categories:**
  1. **Instances without models**
  2. **Instances with models**

- **These two types of instances are managed by the Circuit Object.**

Device Instances without Model

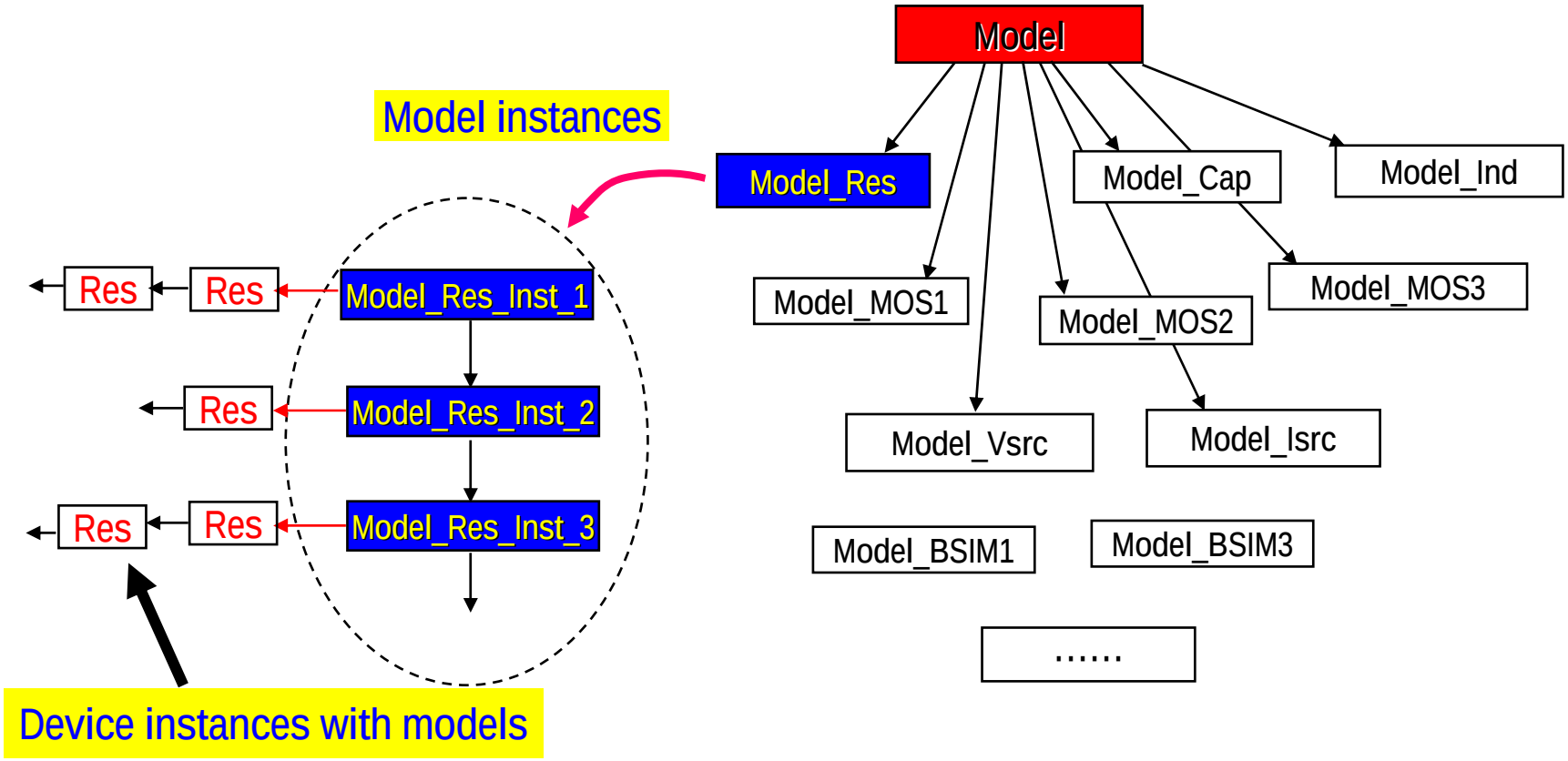


Spice takes a model-driven device instantiating strategy.

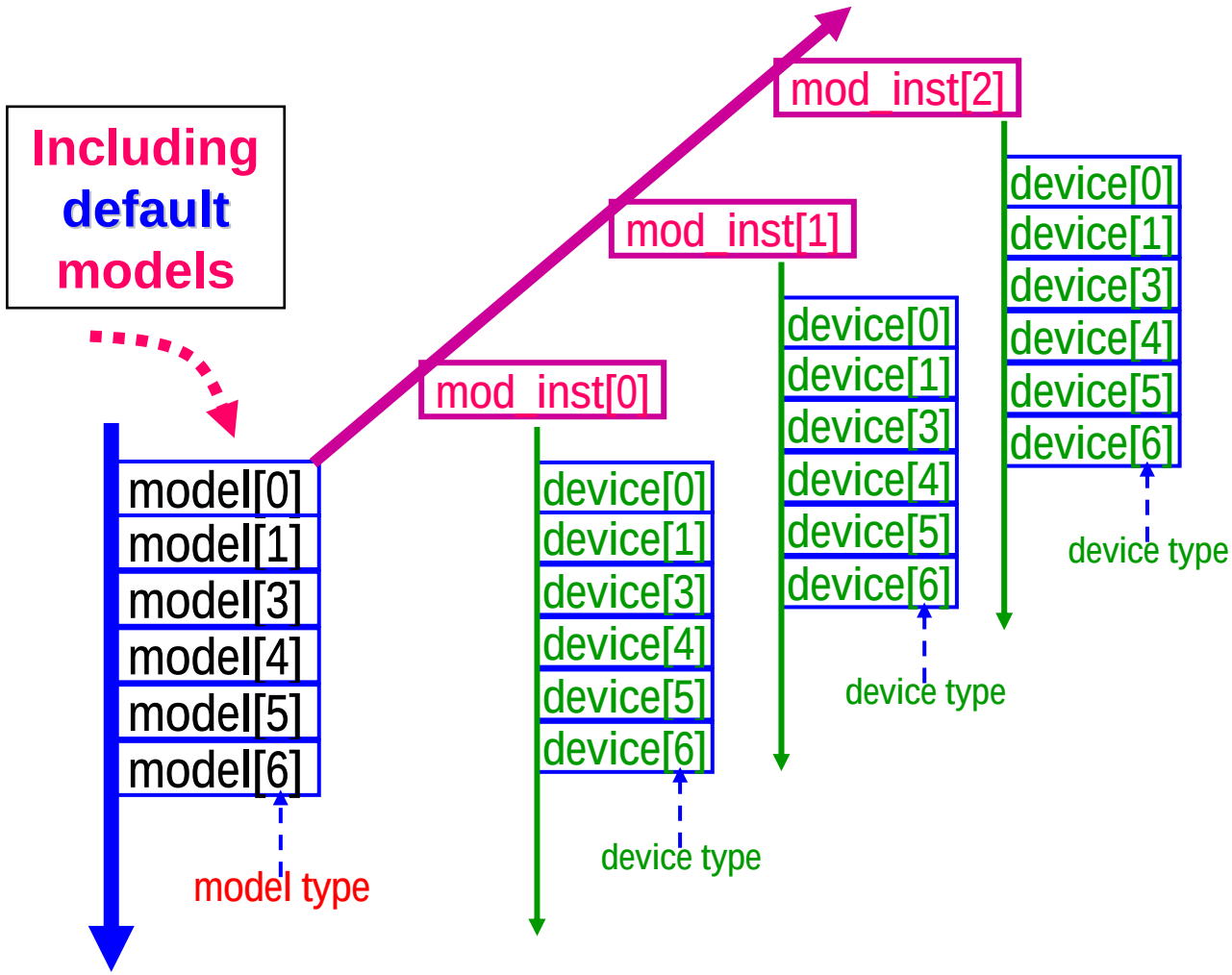
Spice creates a default model for the same type devices without a model.

This means all device instances have models.

Device Instances with Model



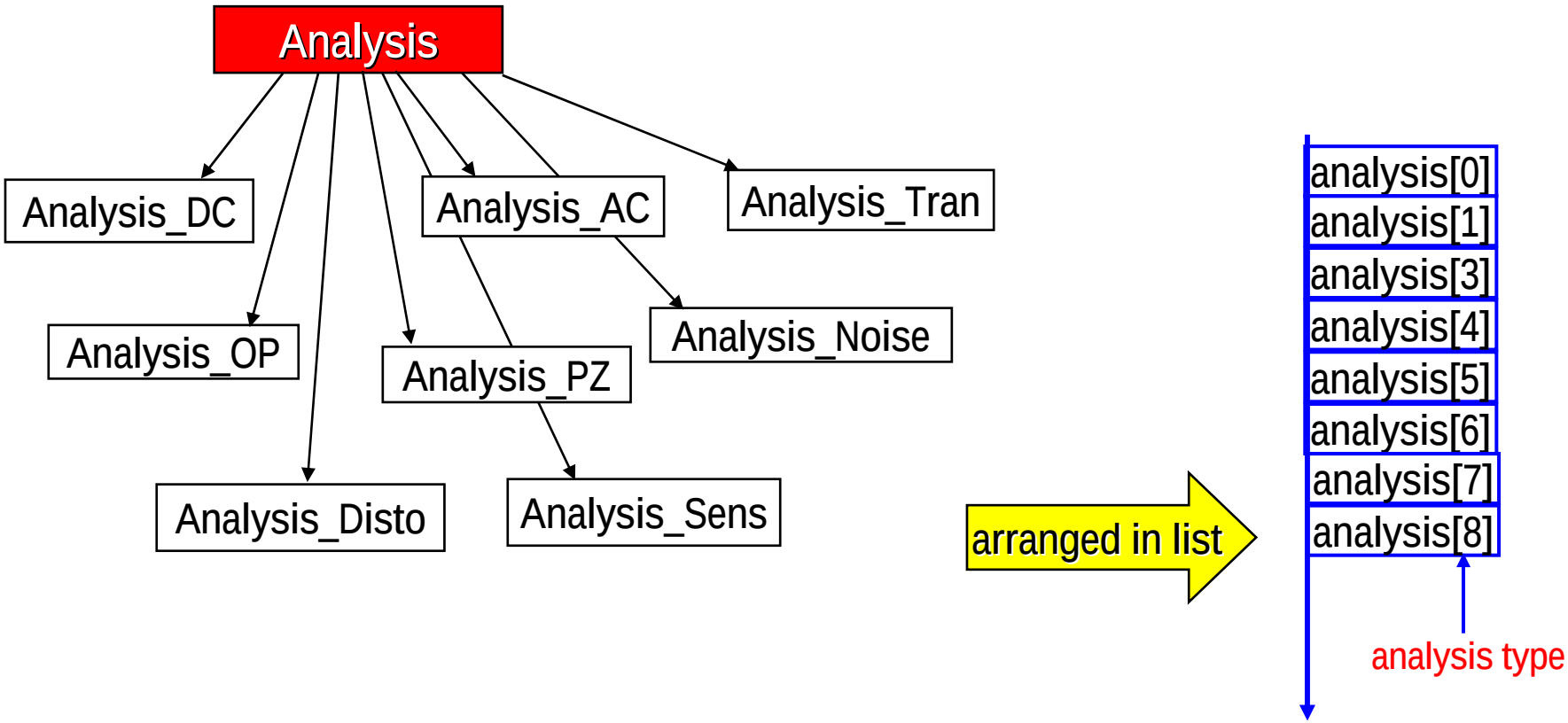
Links among Models & Devices



Model-based Device Instances

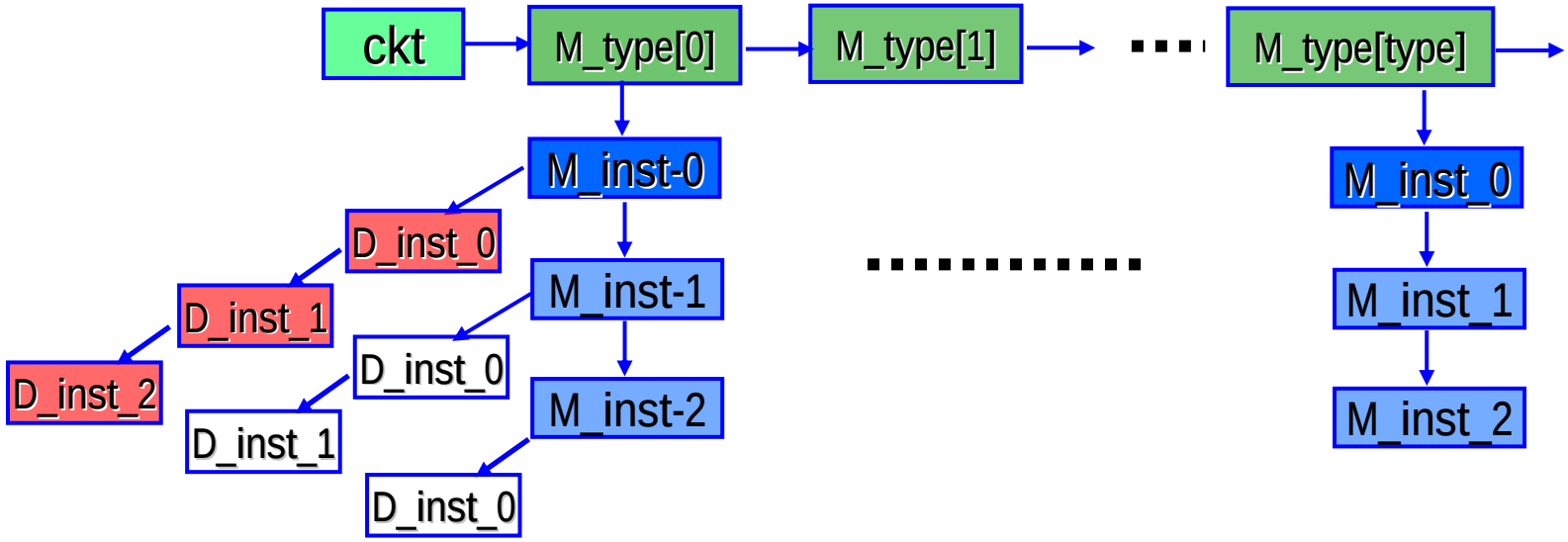
- **If a device does not have a model, a default model is created for it.**
- **The same type devices without model share the same default model.**
- **The parameters for a default model are not assigned.**
- **Spice uses XXXtemp() function to evaluate the stamp, which uses the temperature information.**
  - **If a default model, the model parameters are not used at all.**
  - **Only the lumped parameter given in the netlist is used for evaluation.**

Analysis Classes



Models and Devices

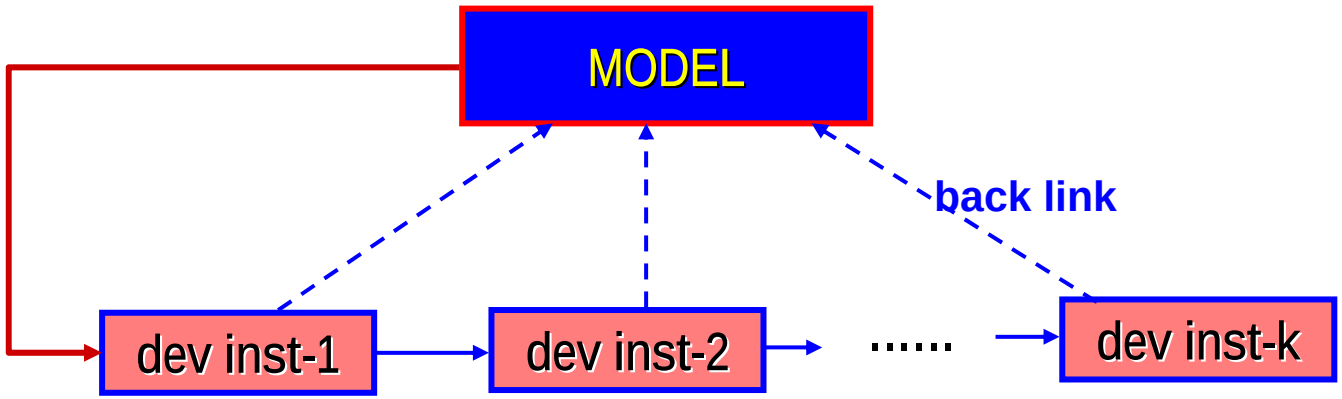
- A netlist may have a list of devices (identified by device types)
- Each type of device may have a list of models
- Each device model may have a list of instances



* This data structure will be used during circuit loading!

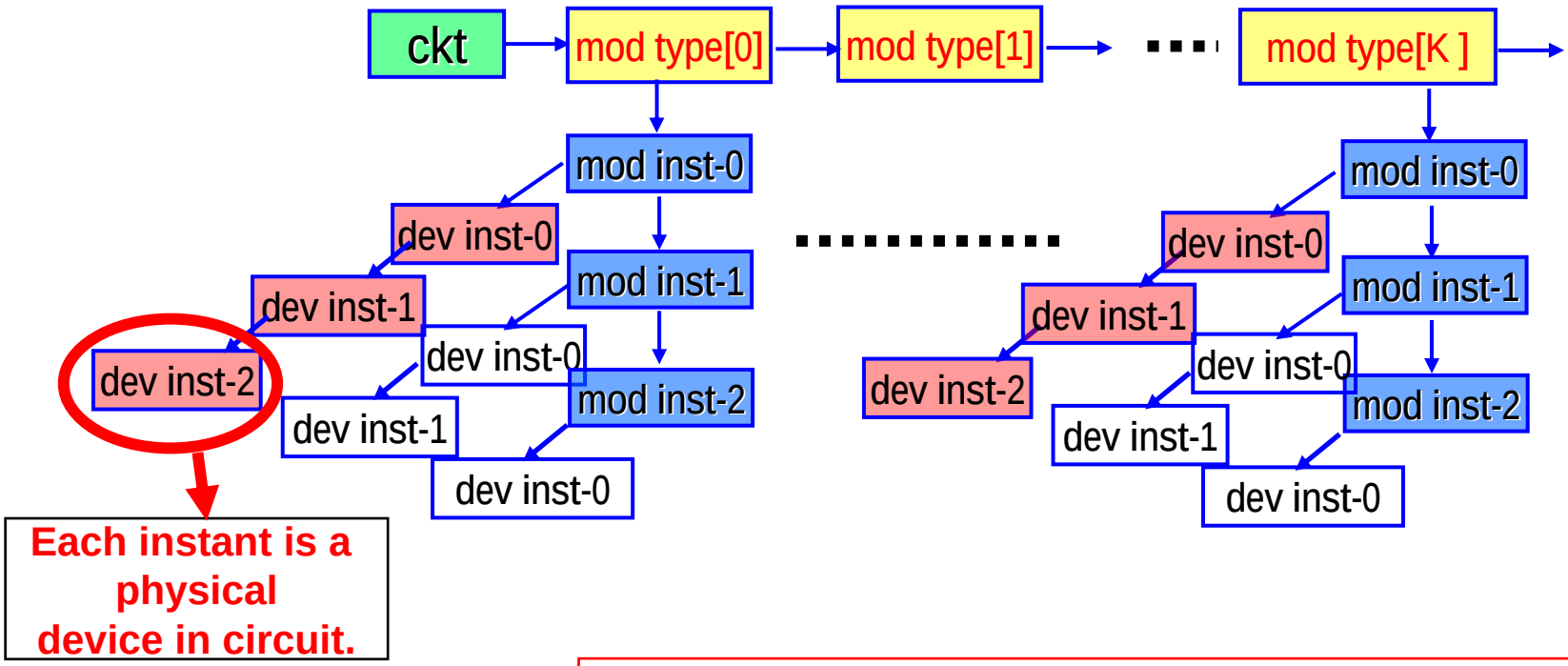
Referencing Model and Instances

- **Each model instance may be linked with a list of device instances.**
- **Each device instance has a back pointer pointing at the belonging model instance.**



*** One model supports multiple devices.**

Device Loading

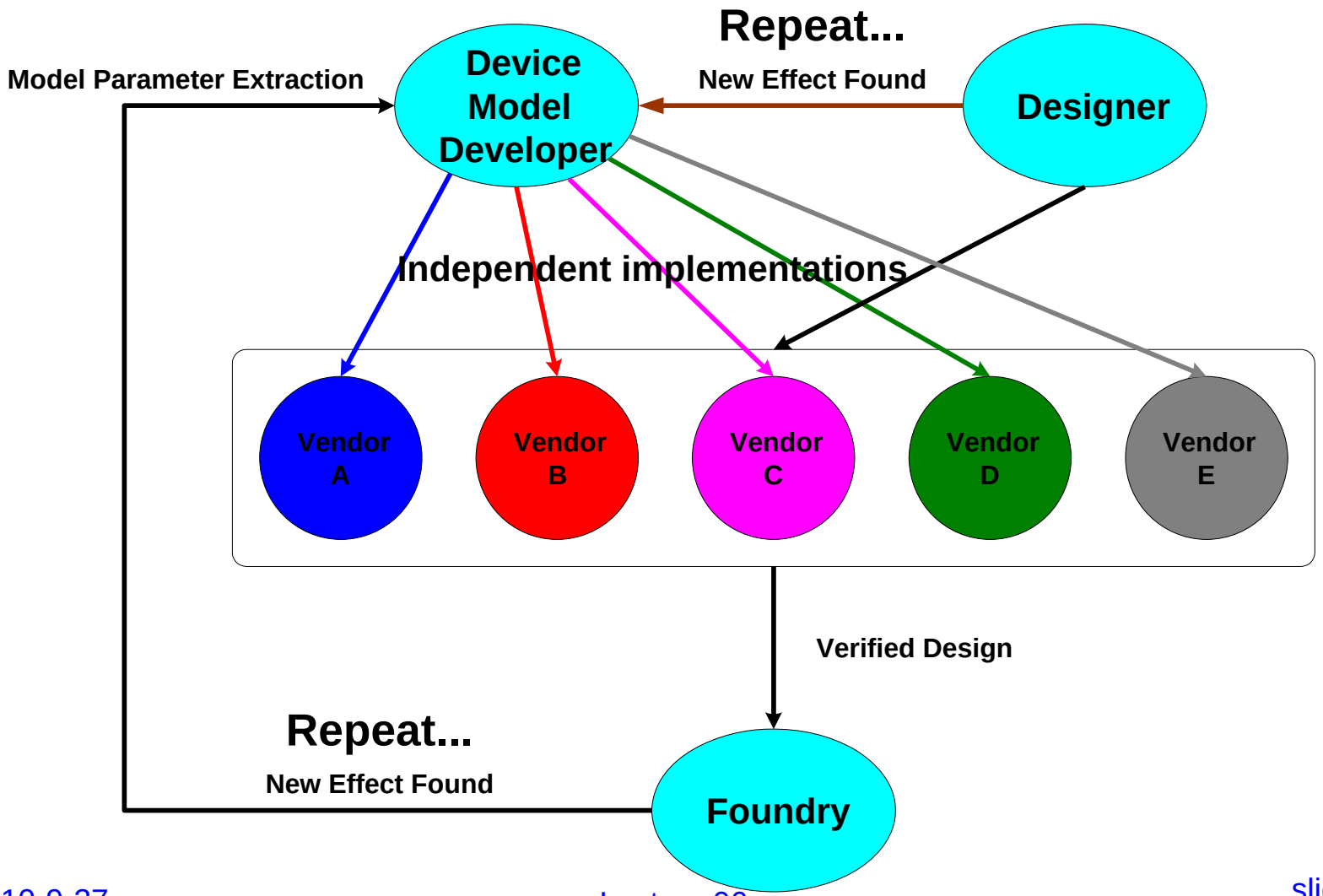


*** This whole structure is scanned during device loading.**

Device Models

- **Semiconductor devices are described by a set of analytical equations.**
- **One model may take several hundreds of pages (BSIM3/4).**
- **Can be written in description language – Verilog-AMS.**
- **Have to be translated into C/C++ code.**

Typical Modeling Flow



Model Implementation

- **Refer to C implementation of model equations;**
  - **including derivative computations (Jacobian matrix, ...)**
- **Model implementation is more *difficult, time consuming and error-prone* than model creation (writing math eqns).**
- **Simulator vendors have to develop their own implementations (IP issues)**
- **Model implementation is also costly.**

- **Designers have less control over models.**
- **Designers may identify flaws in models while using simulators.**

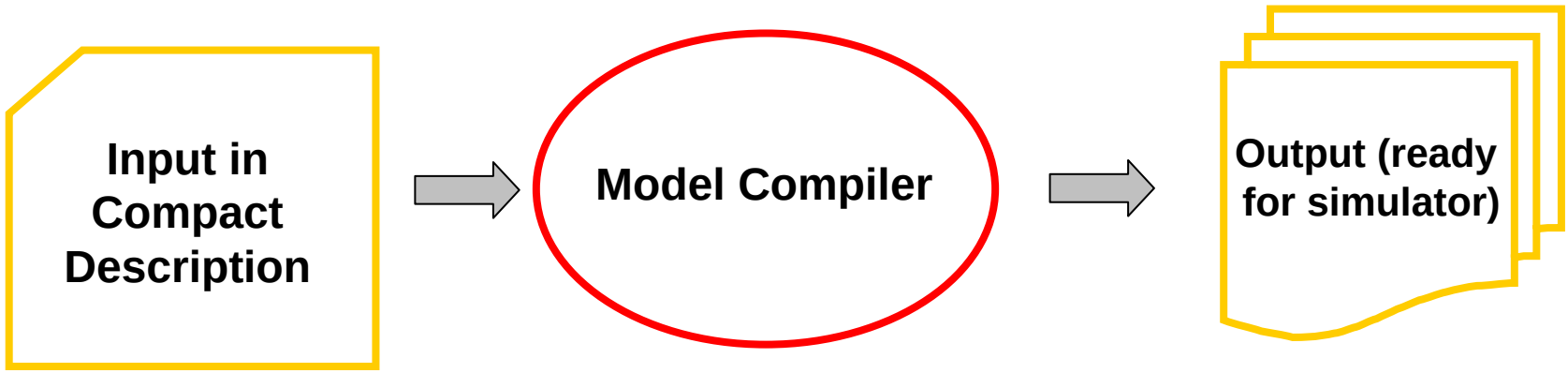
Designers Want ...

- **Accurate and robust models.**
- **Models covering all cases likely to appear in real design.**
- **Models that can be simulated efficiently.**

- **Sometimes want to modify the model equations in their own favor ...**

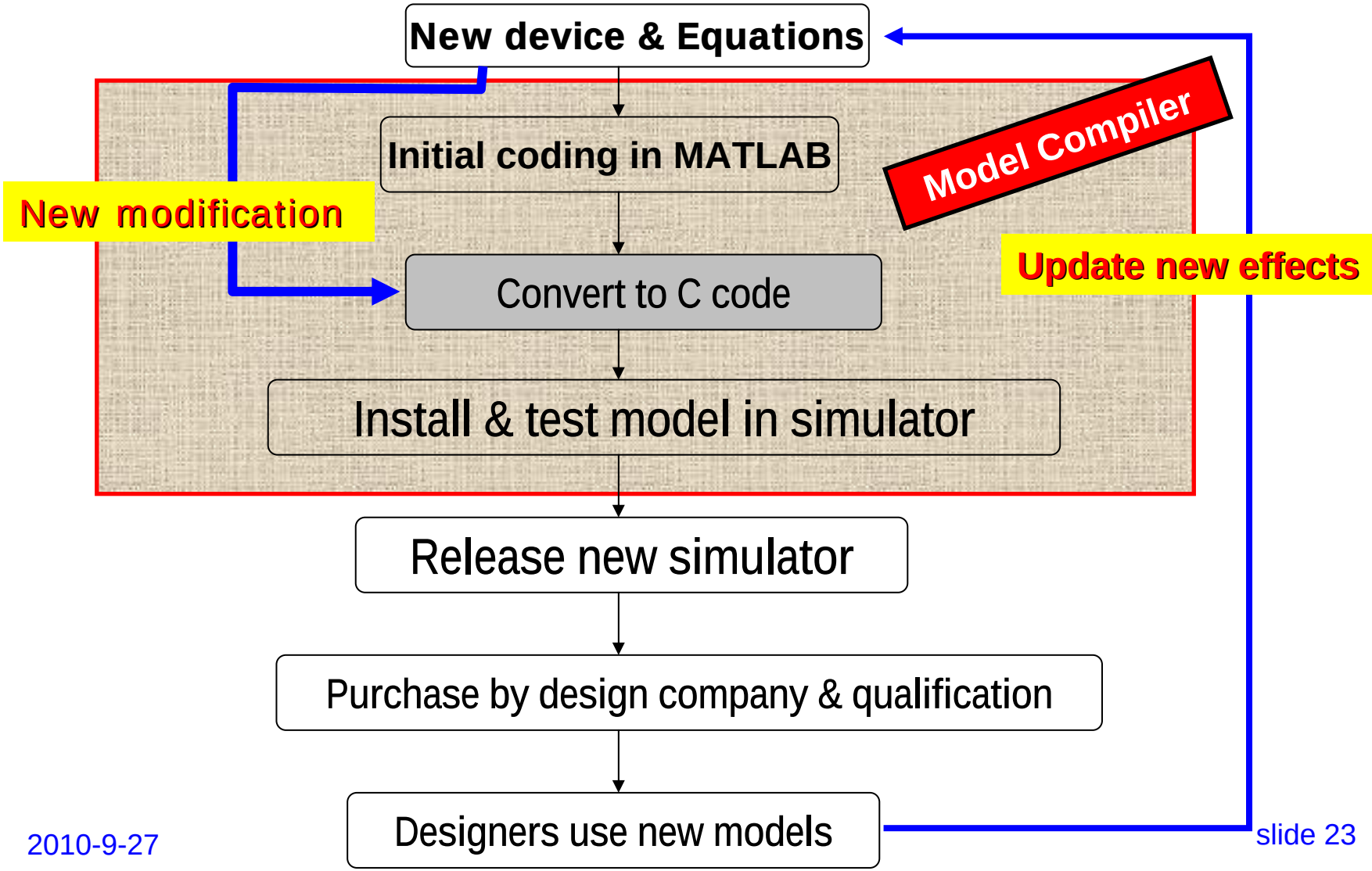
Model Compiler Can Help !

A Model Compiler is a CAD tool that supports automatic implementation of compact device model.



1. **Input: compact device models in a description language - Verilog-AMS**
2. **Output: device code in C/C++ that can be directly compiled in Spice-like simulators.**

Device Model Implementation Flow



APPENDIX -- Object-Oriented Programming

Object-Oriented Programming

- **Data and functions associated with the same object are collected in one class**
  - **So-called "encapsulation"**
  - **Provides modularity of code.**

Use Polymorphism

- **Define object interface in the base classes using virtual functions in C++ for polymorphism.**
- **Implementation of objects defined in the derived classes.**
- **Better code readability / flexibility, and**
- **Easier code management.**

- **Polymorphism is very suitable for model interfacing and device methods.**

Modular Development

- **Make the numerical methods, modeling, and analysis independent from each other as much as possible.**
- **Make the Solver Module independent of the analysis algorithms; so that it is easier to update the solver.**
- **Make device models Independent of the simulator analysis engine;**