The Abdus Salam International Centre for Theoretical Physics

2499-24

International Training Workshop on FPGA Design for Scientific Instrumentation and Computing

11 - 22 November 2013

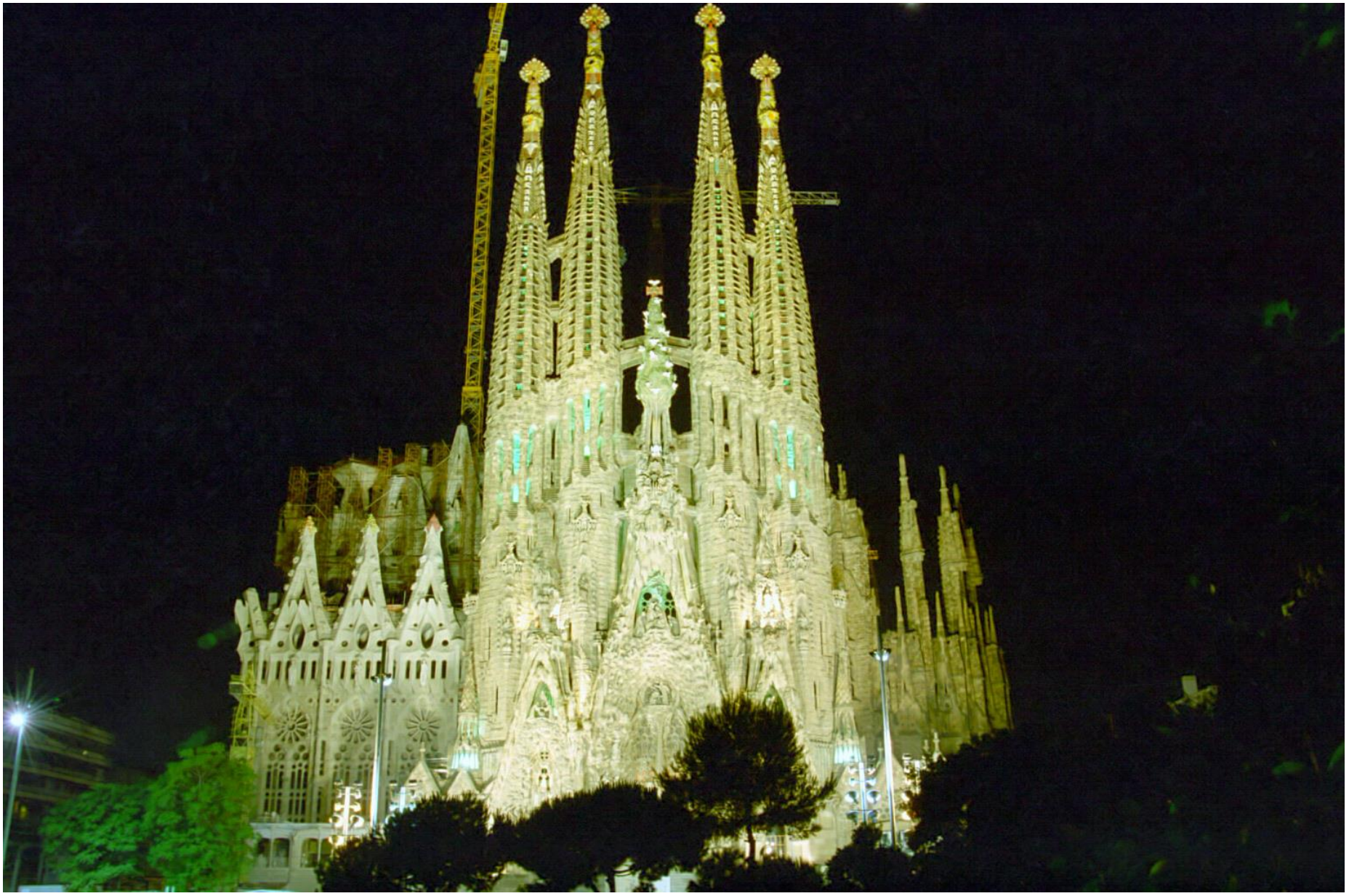
Introduction to SoC Modeling

Julio D. DONDO GAZZANO

Departamento de Tecnologias y Sistemas de la Informacion, Escuela Superior de Informatica, Universidad de Castilla la Mancha
Ciudad Real
Spain

Introduction to System on Chip modelling

Source:http://memetician.livejournal.com/201202.htm



Source:http://historiadelarte-amparosantos.blogspot.com/2010/04/la-sagrada-familia-antonio-gaudi.html

Model

What a model is?

An abstract view of a design.
Providing relevant aspects
Hiding non relevant ones

Design gap

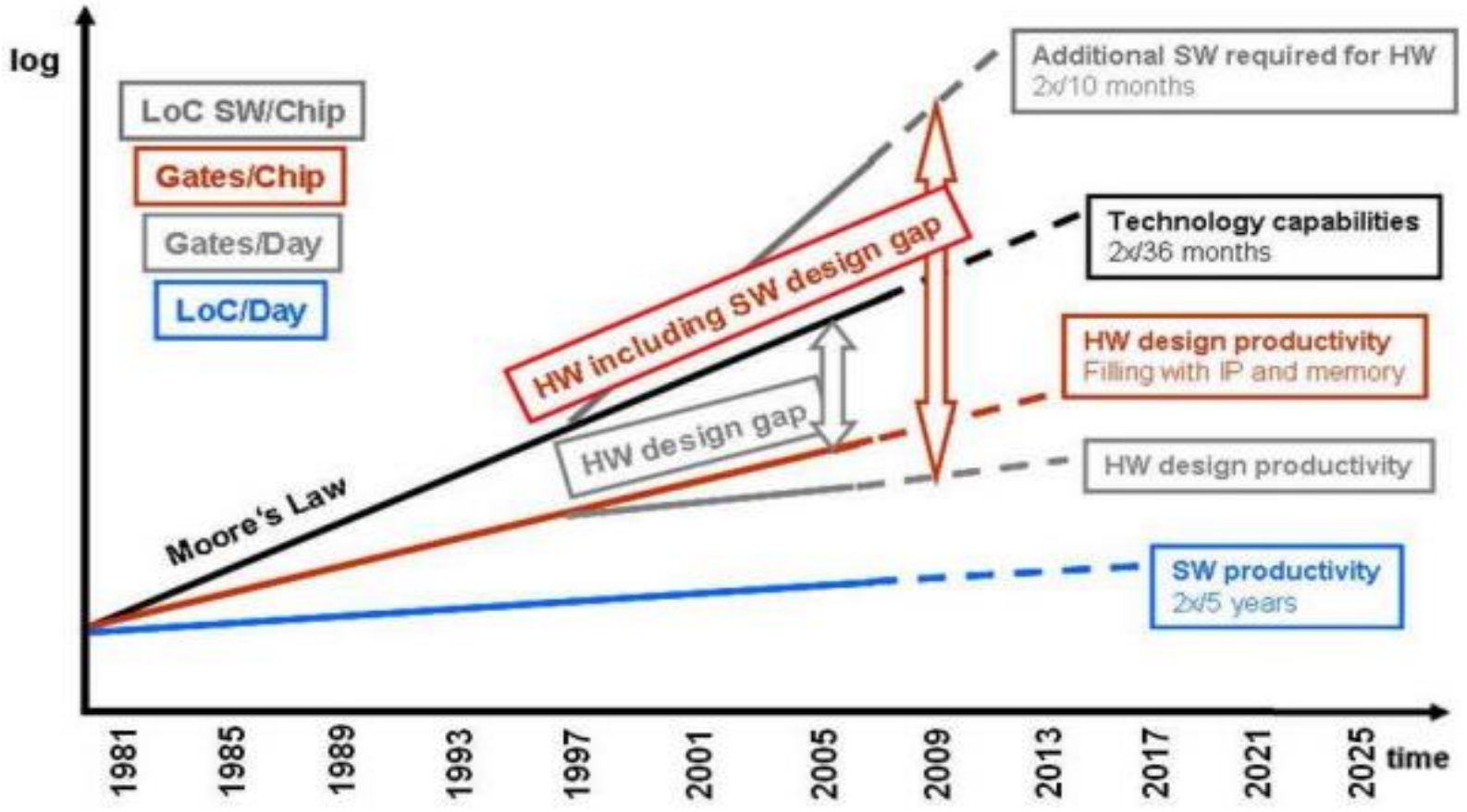


Figure DESN3 Hardware and Software Design Gaps versus Time[5]

Source: THE INTERNATIONAL TECHNOLOGY ROADMAP FOR SEMICONDUCTORS: *2009*

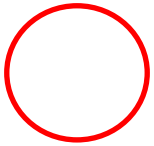
Abstraction levels

Traditional design methods, in which systems are designed directly at the low hardware or software levels, are fast becoming infeasible.

One solution for closing the productivity gap is to raise the level of abstraction in the design process.

Abstraction levels

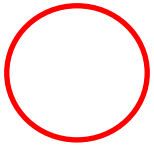
Circuit



What the components generated at circuit level are?

Abstraction levels

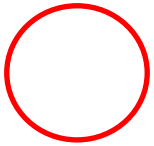
Circuit



Cells, formed by P-type and N-type transistors.

Abstraction levels

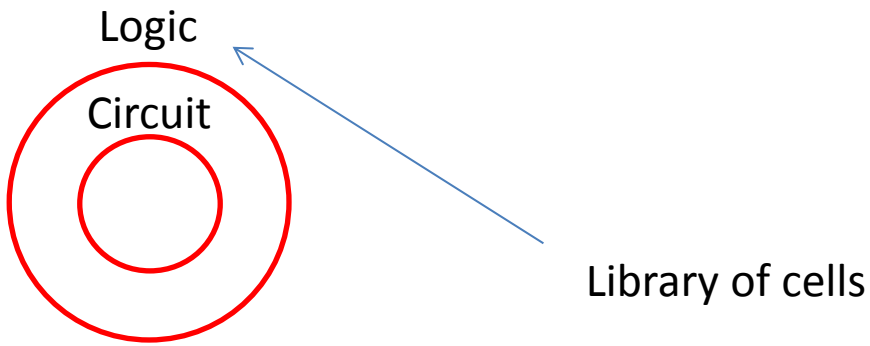
Circuit



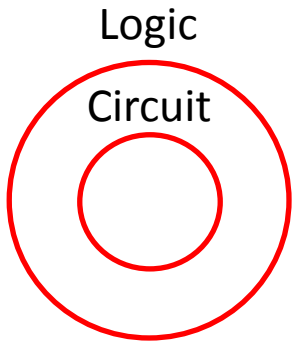
Cells, formed by P-type and N-type transistors.

Library of cells

Abstraction levels

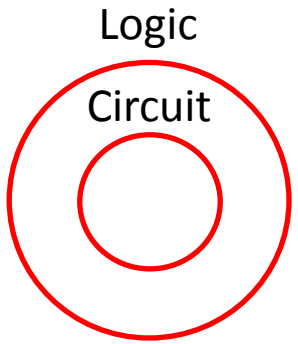


Abstraction levels



Logic gates and flip-flops to generate register-transfer components

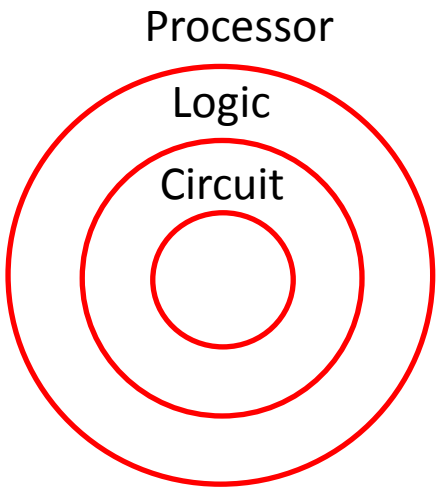
Abstraction levels



Logic gates and flip-flops to generate register-transfer components

Library of logic gates to create components at RTL level, such as registers, register files, ALUs, multipliers, and other components for processor micro architecture

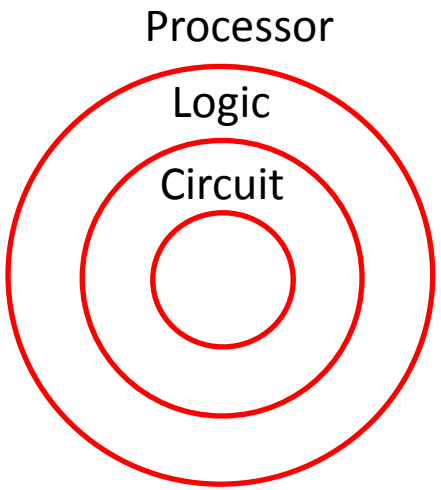
Abstraction levels



On the processor level, standard and custom processors, or special-hardware components (memory controllers, arbiters, bridges, routers, and various interface components) are generated.

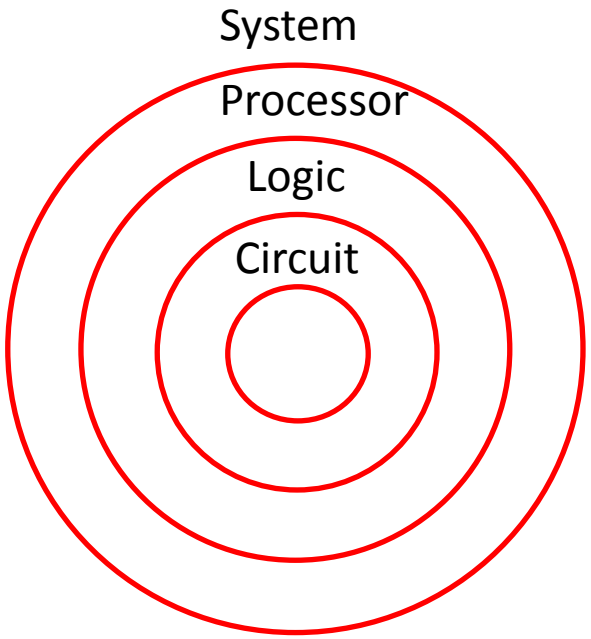
PE Processing Elements
CE Communication Elements

Abstraction levels



At this level it is also performed the floorplanning, placement, and routing of these PEs or CEs using the components from RTL library, and they are stored in a Processor library.

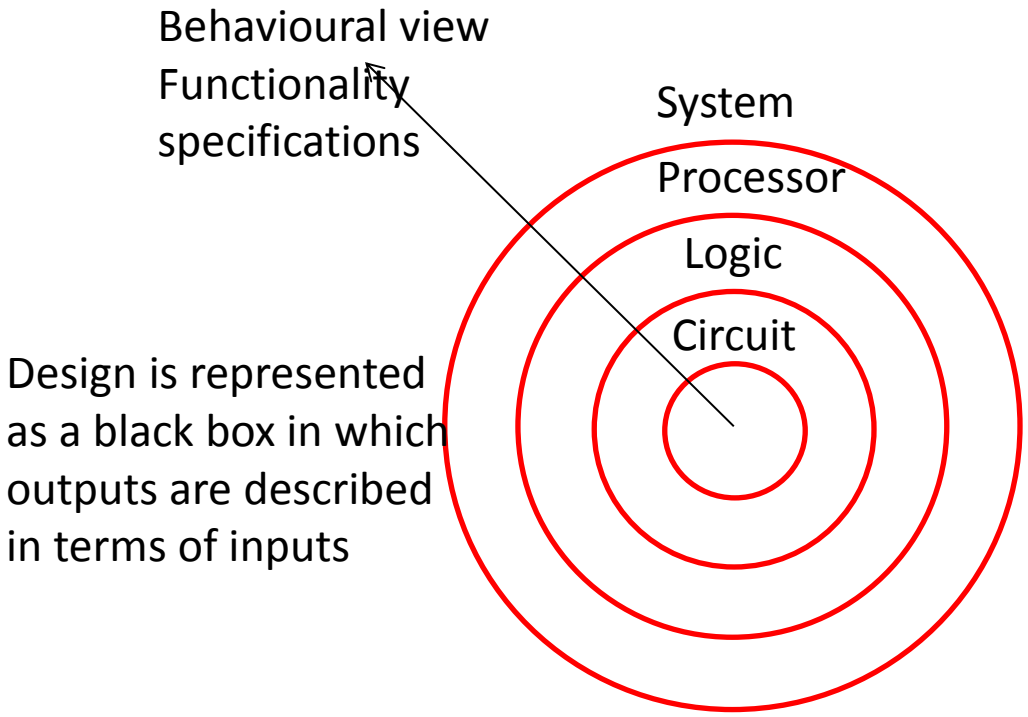
Abstraction levels



Finally, on the system level, standard or embedded systems formed by processors, buses, memories, and other processor components are designed.

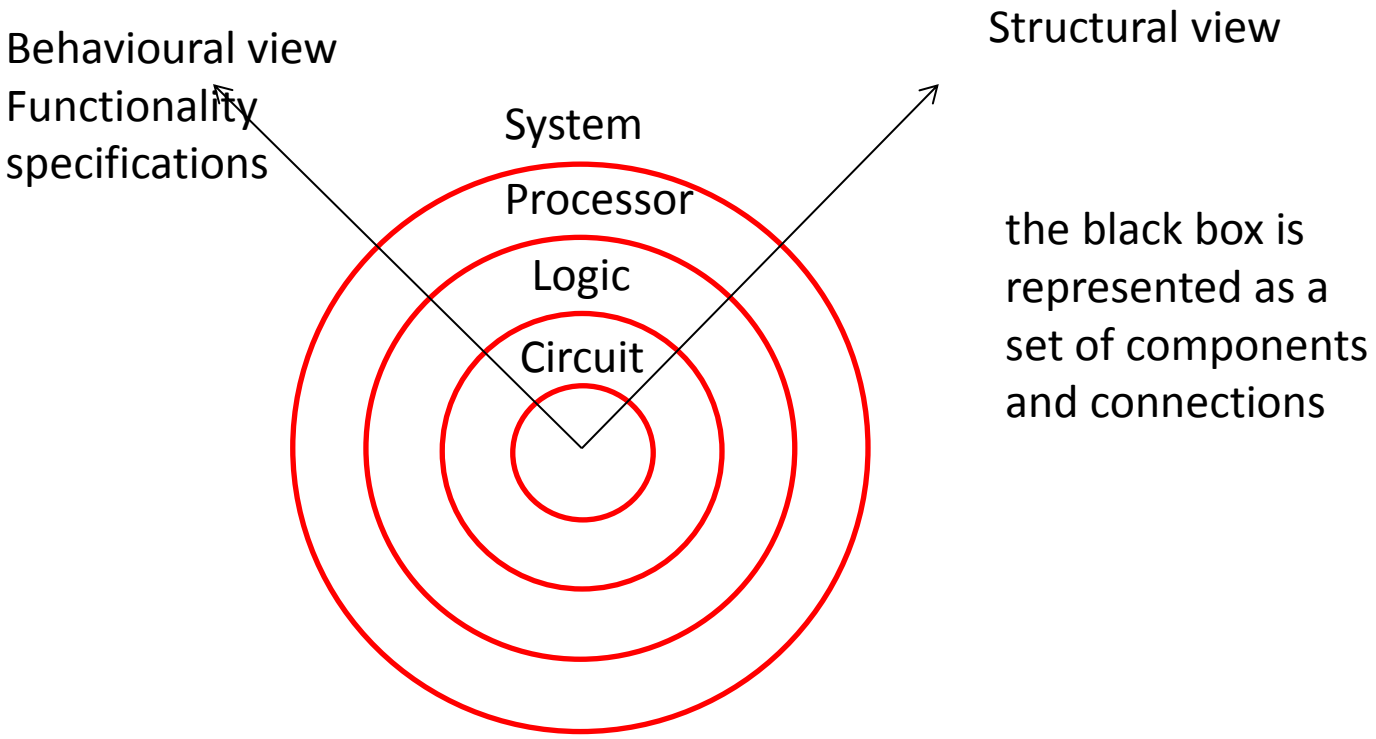
Y-chart

Three different views



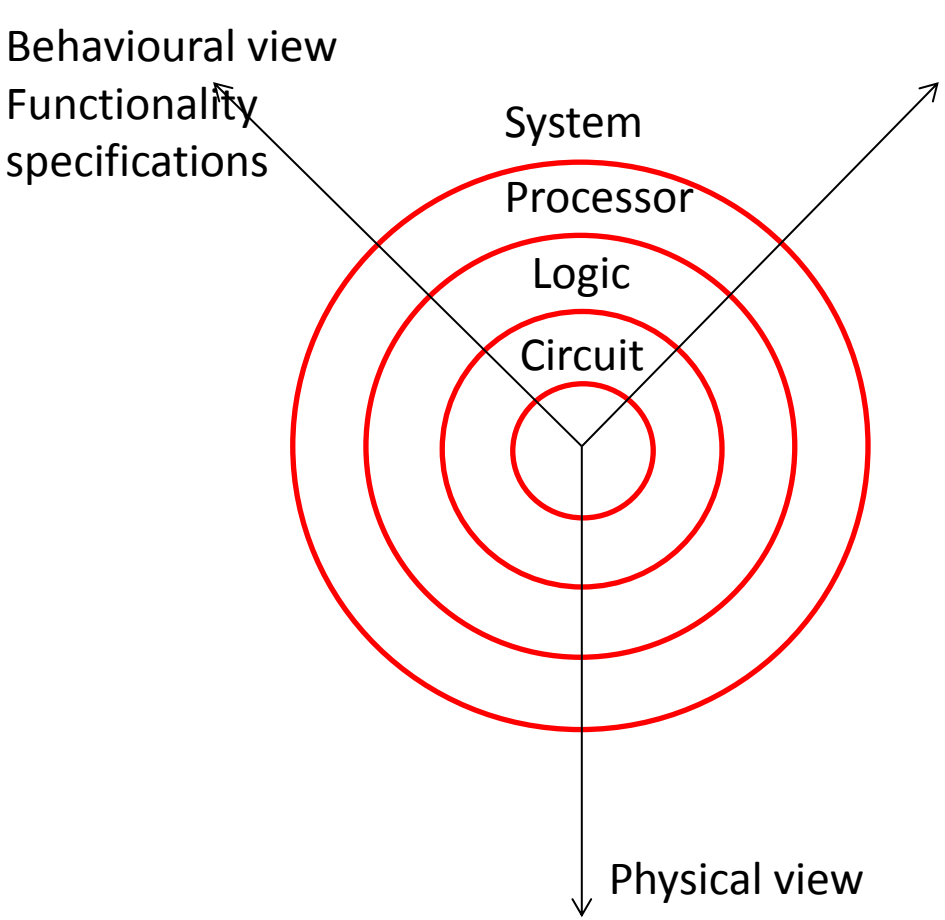
Y-chart

Three different views



Y-chart

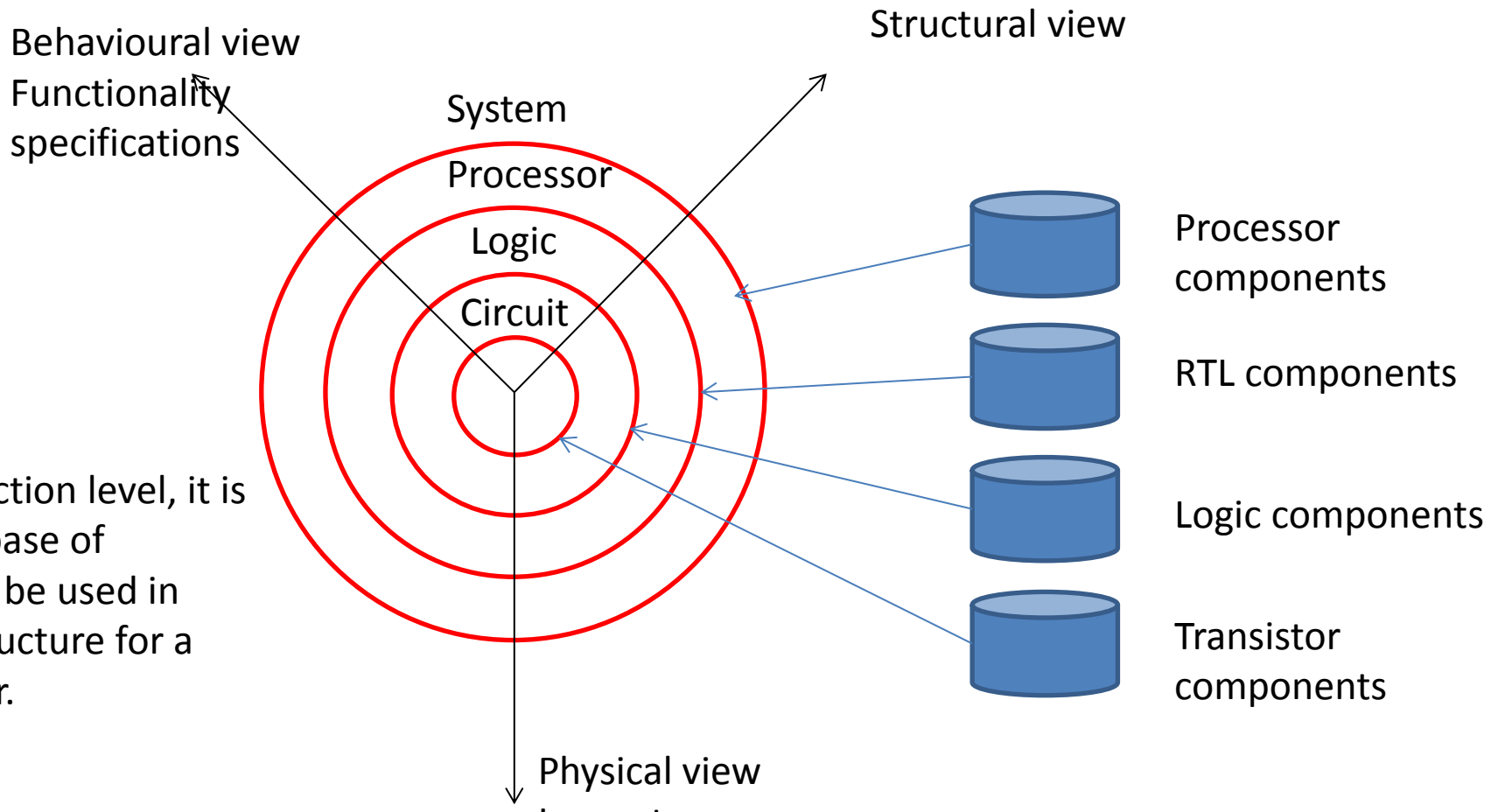
Three different views



Specifies the size and the position of each component, as well as each port and its connection.

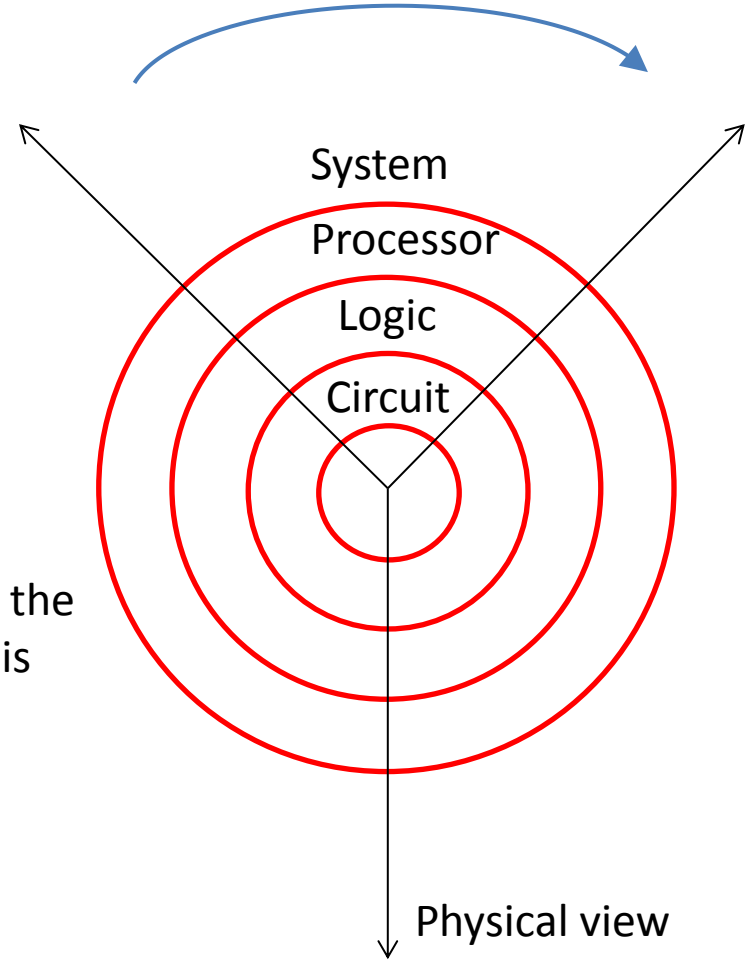
Y-chart

Three different views



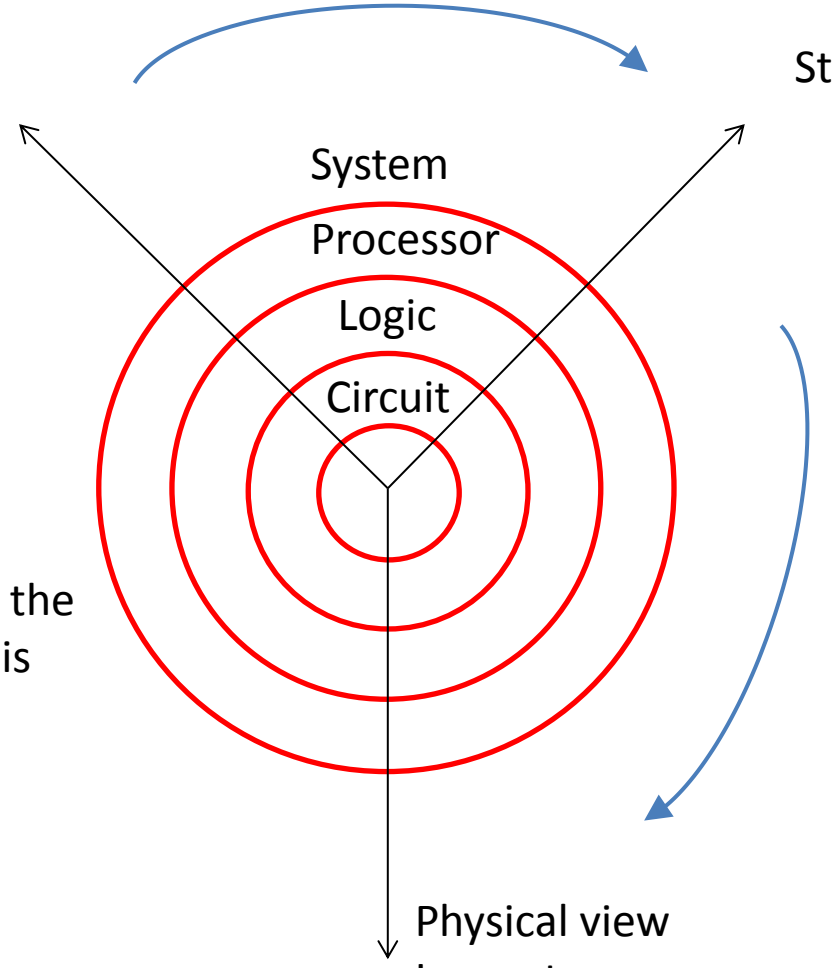
On each abstraction level, it is needed a database of components to be used in building the structure for a given behaviour.

Y-chart



The process of converting the behaviour in an structure is called **Synthesis**

Y-chart



The process of converting the behaviour in an structure is called **Synthesis**

Physical design consists in floorplanning, placement and routing on the chip

Y-chart

Three different views

Behavioural view

Structural view

System

Processor

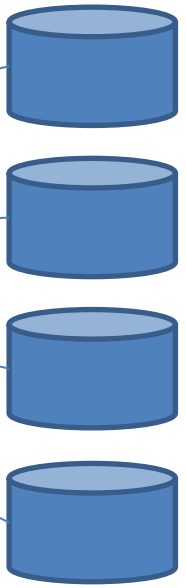
Logic

Circuit

Physical view

Lay-out

Each component in the database needs a model representing the different view.



Processor components

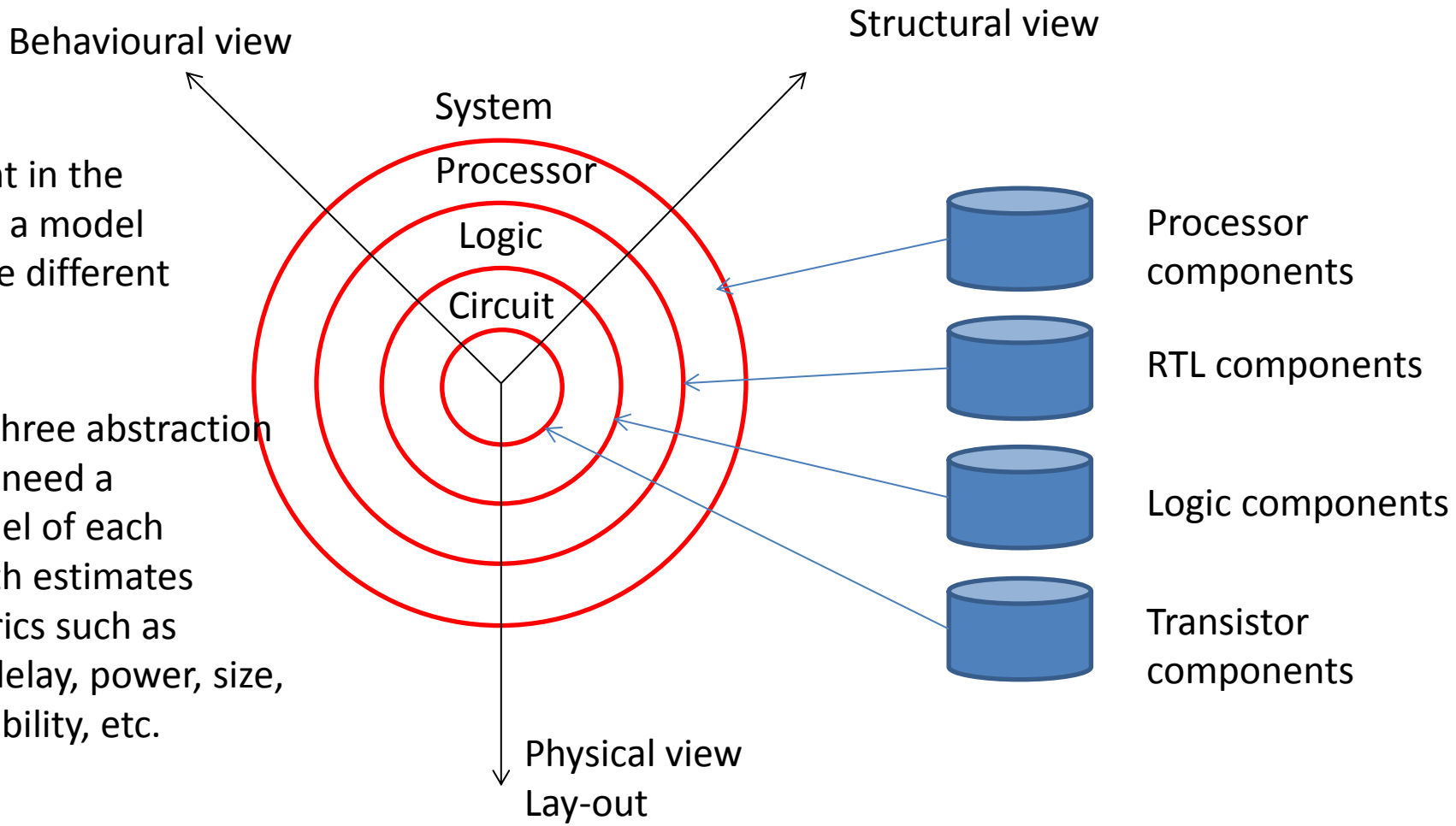
RTL components

Logic components

Transistor components

Y-chart

Three different views



Each component in the database needs a model representing the different view.

for the higher three abstraction levels, we only need a functional model of each component with estimates of the key metrics such as performance, delay, power, size, reliability, testability, etc.

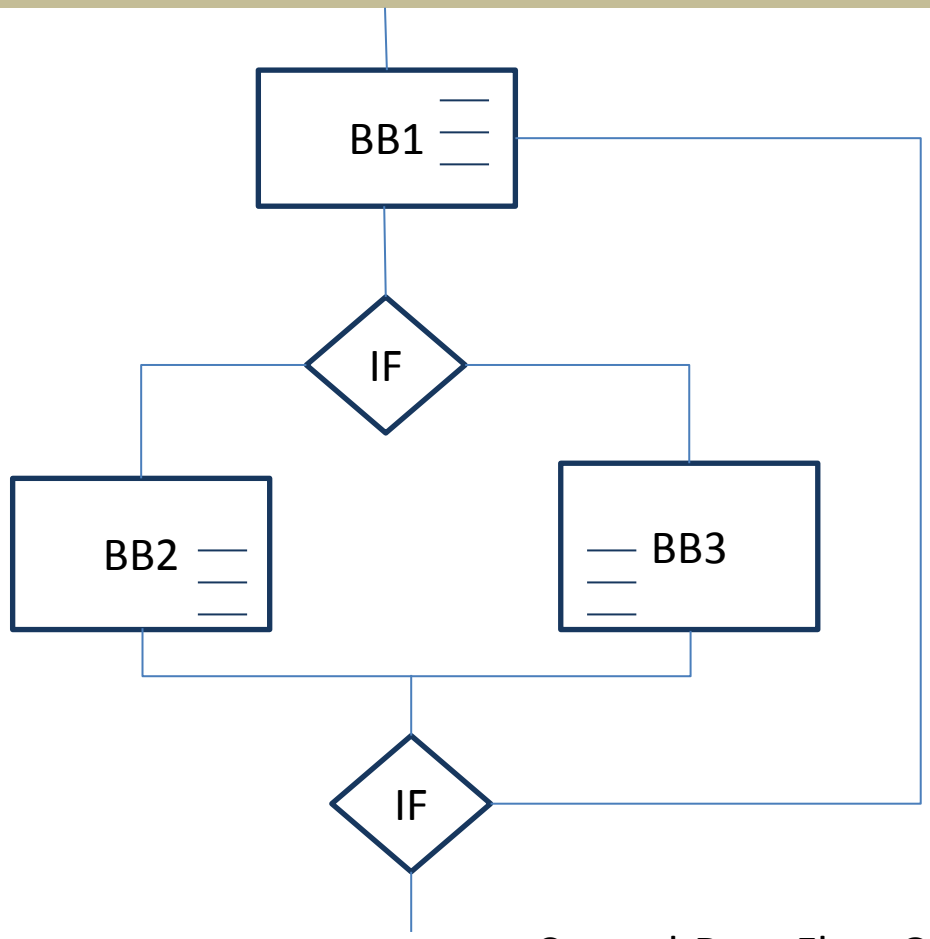
Processor-level behavioural model

On the processor level we design processing elements (PE)
PE can be a custom component or a standard Processor

The behaviour or the functionality of PE can be described using

- **FSM: Finite state machine** – can be made clock-accurate if each state takes one clock cycle.
- **FSMD: Finite state machine with data**. Extended version of FSM with integer or floating point variable.
  FSMD model is usually not clock-accurate since computation in each state may take more than one clock cycle.
  FSMD model is not adequate to represent the computation expressed by standard programming languages such as C.
- **CDFG: Control-Data Flow Graph**. It represents any programming-language code
- **ISFC: Instruction Set Flow Chart** describing the fetch, decode and execute stage of each instruction of standard PE.

Processor-level behavioural model

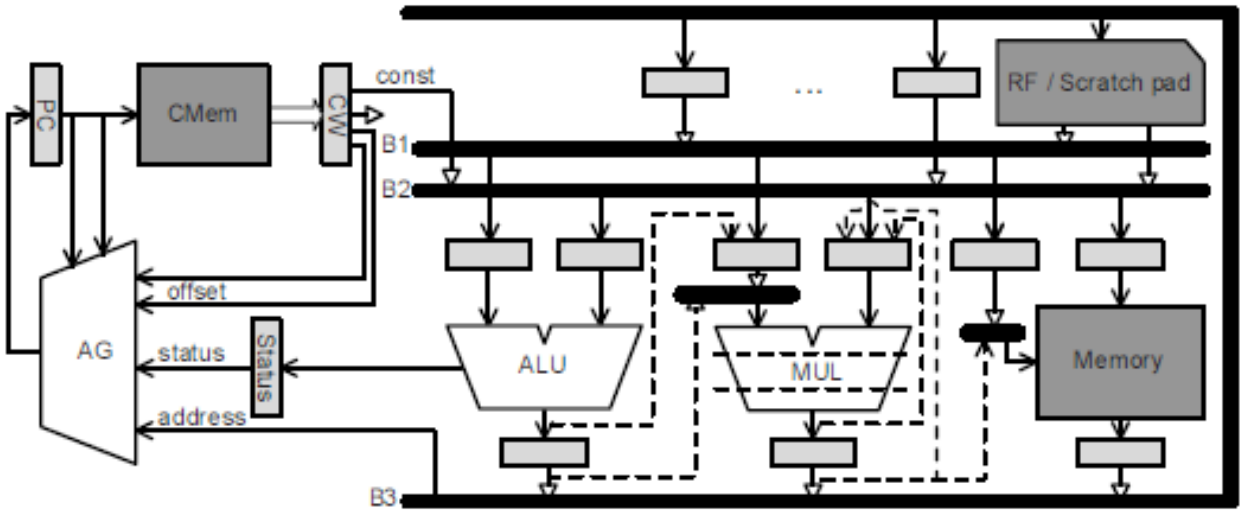


Control-Data Flow Graph

Processor-level structural model

Processor structural model usually consists of
a controller and
a datapath

A datapath consists of a set of storage elements (such as registers, register files, and memories), a set of functional units (such asALUs, multipliers, shifters, and other custom functional units), and a set of busses.

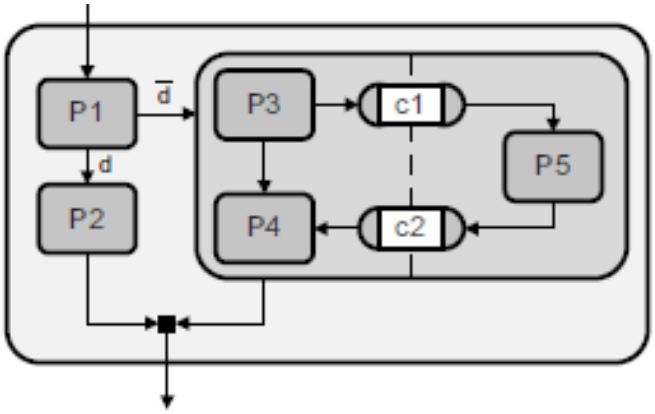


Source:Gajski D.D. et al. Embedded System Design - Modeling, Synthesis and Verification

System-level behavioural model

Multiple processes running in parallel in Sw and Hw require a system-level model.

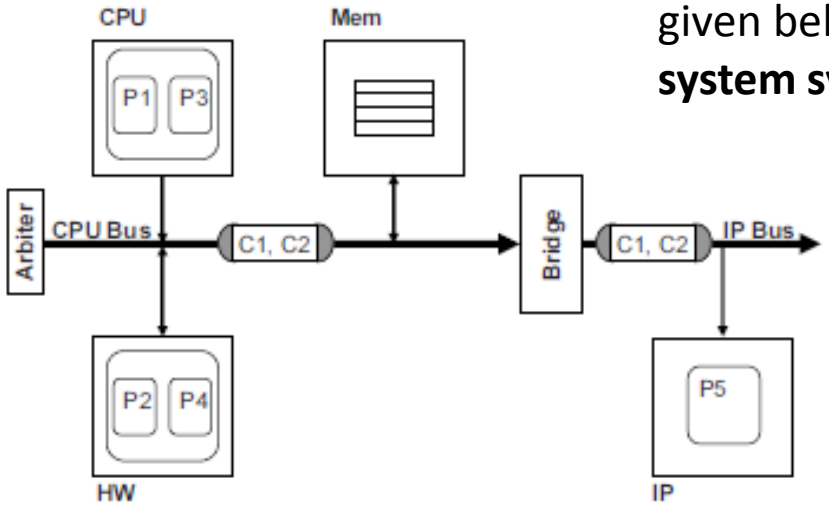
In order to represent a many-processor platform working in parallel or in pipelined mode, we must introduce concurrency and pipelining in the model.
If processes are working concurrently it is necessary a synchronization mechanism for data exchange.
The model must support Hierarchy.



Source:Gajski D.D. et al. Embedded System Design - Modeling, Synthesis and Verification

System-level structural model

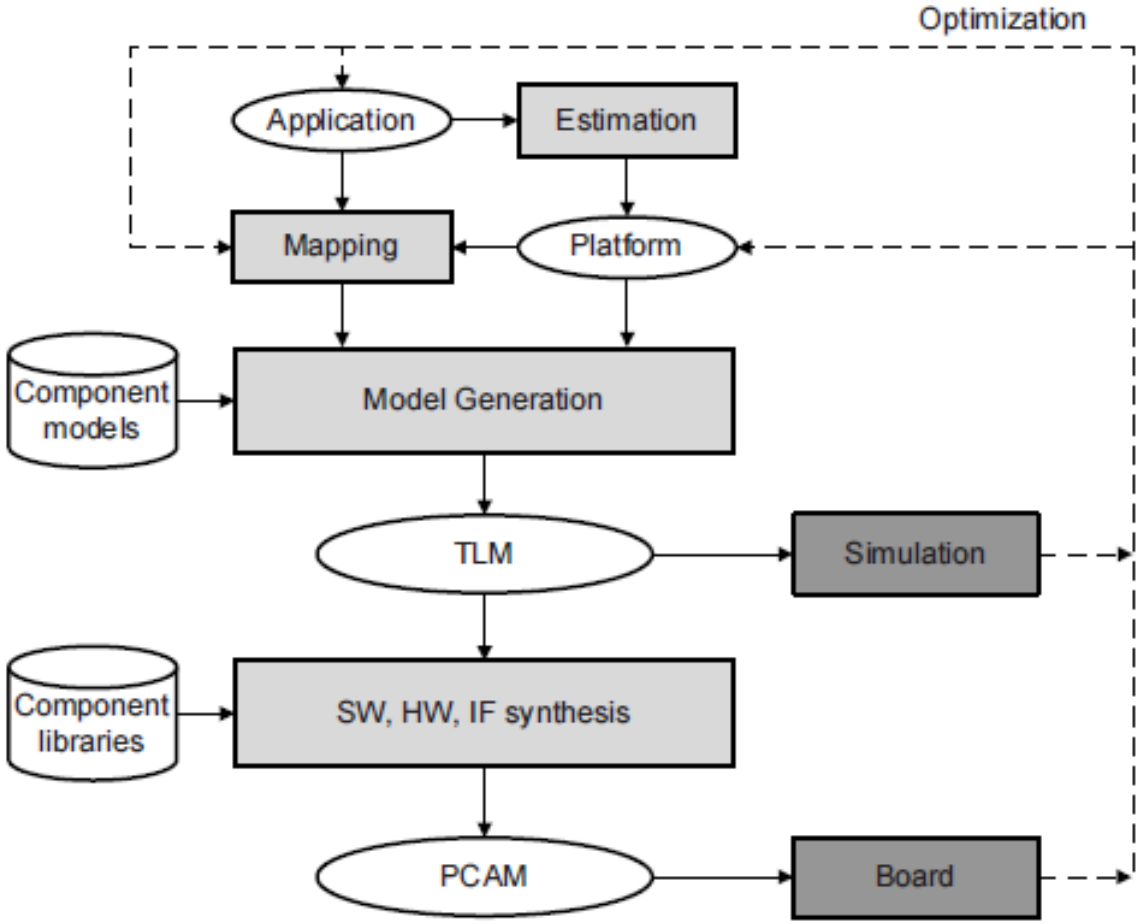
It is a netlist of system components used for computation, storage, and communication



A system structural model is generated from the given behavioural model by the process called **system synthesis.**

Source:Gajski D.D. et al. Embedded System Design - Modeling, Synthesis and Verification

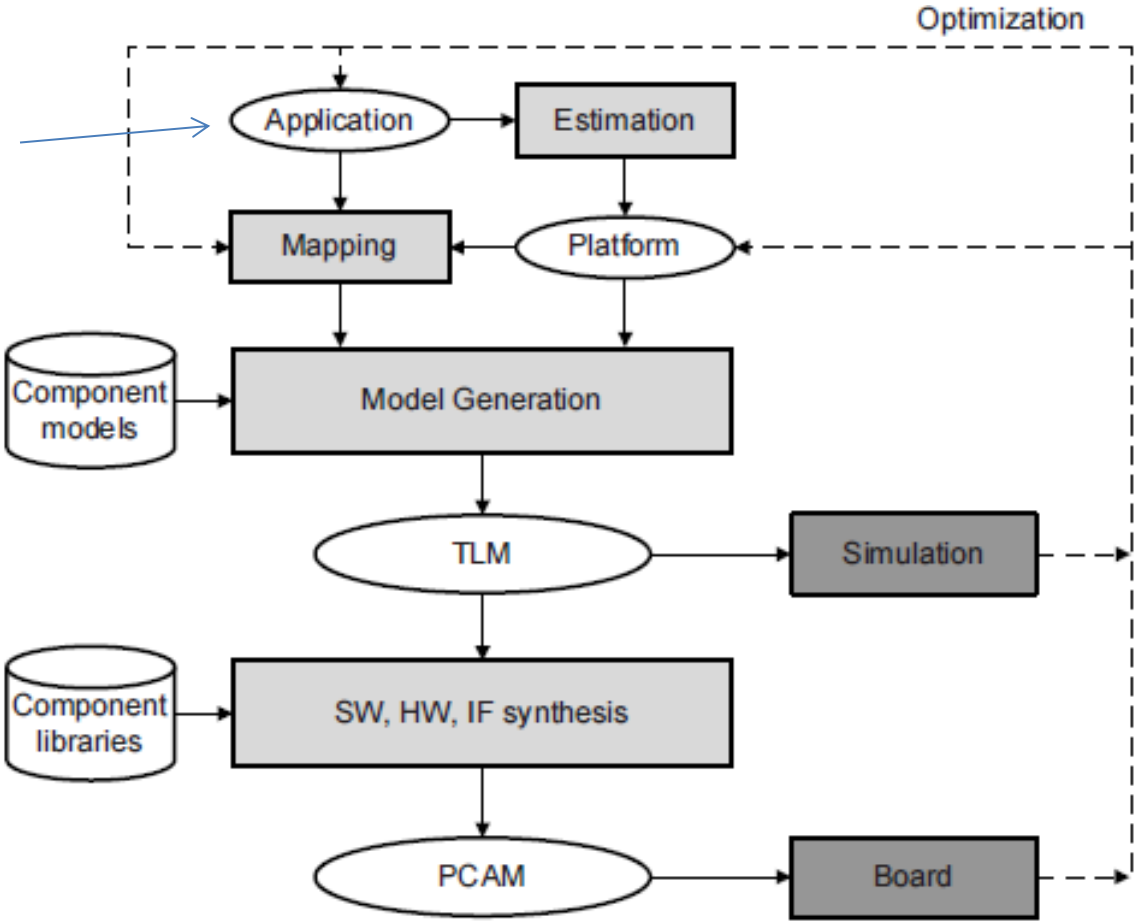
System-level synthesis



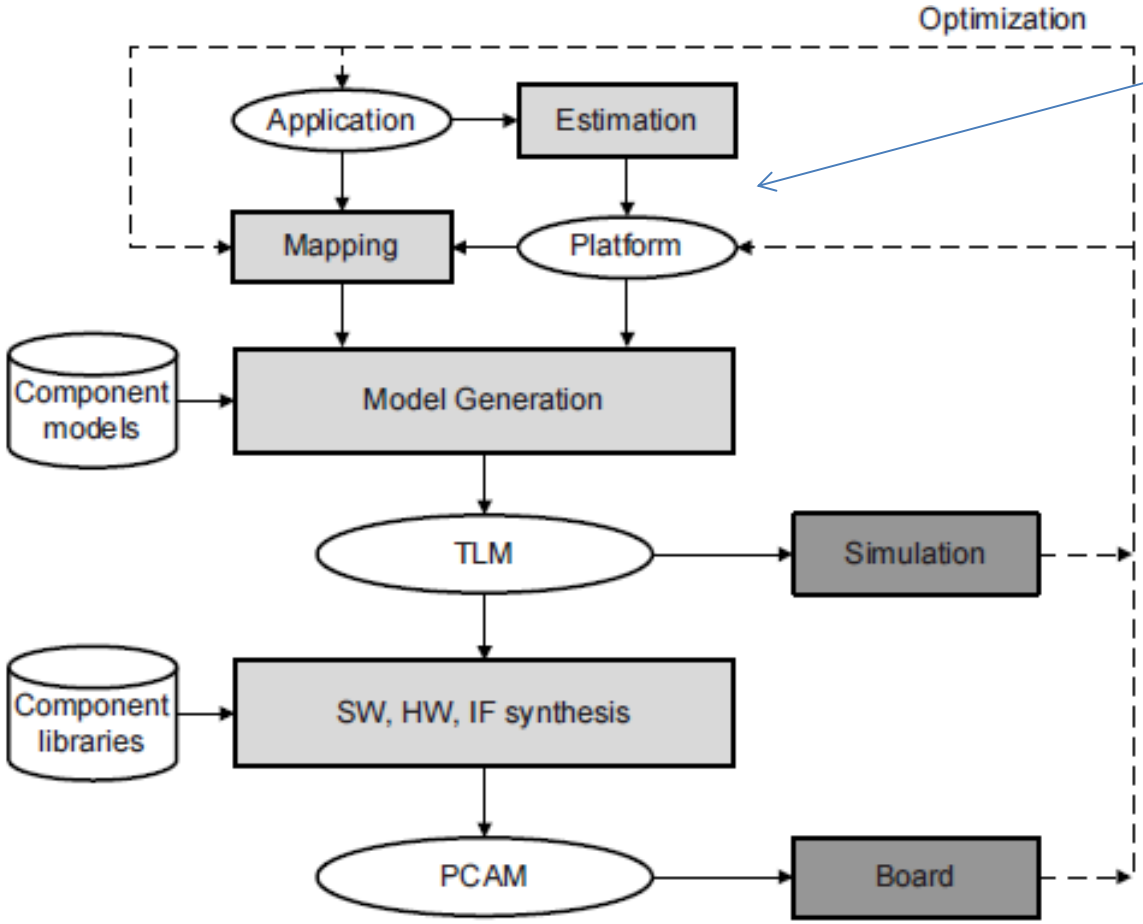
Source:Gajski D.D. et al. Embedded System Design - Modeling, Synthesis and Verification

System-level synthesis

Set of sequential and parallel processes communicating through message-passing channels



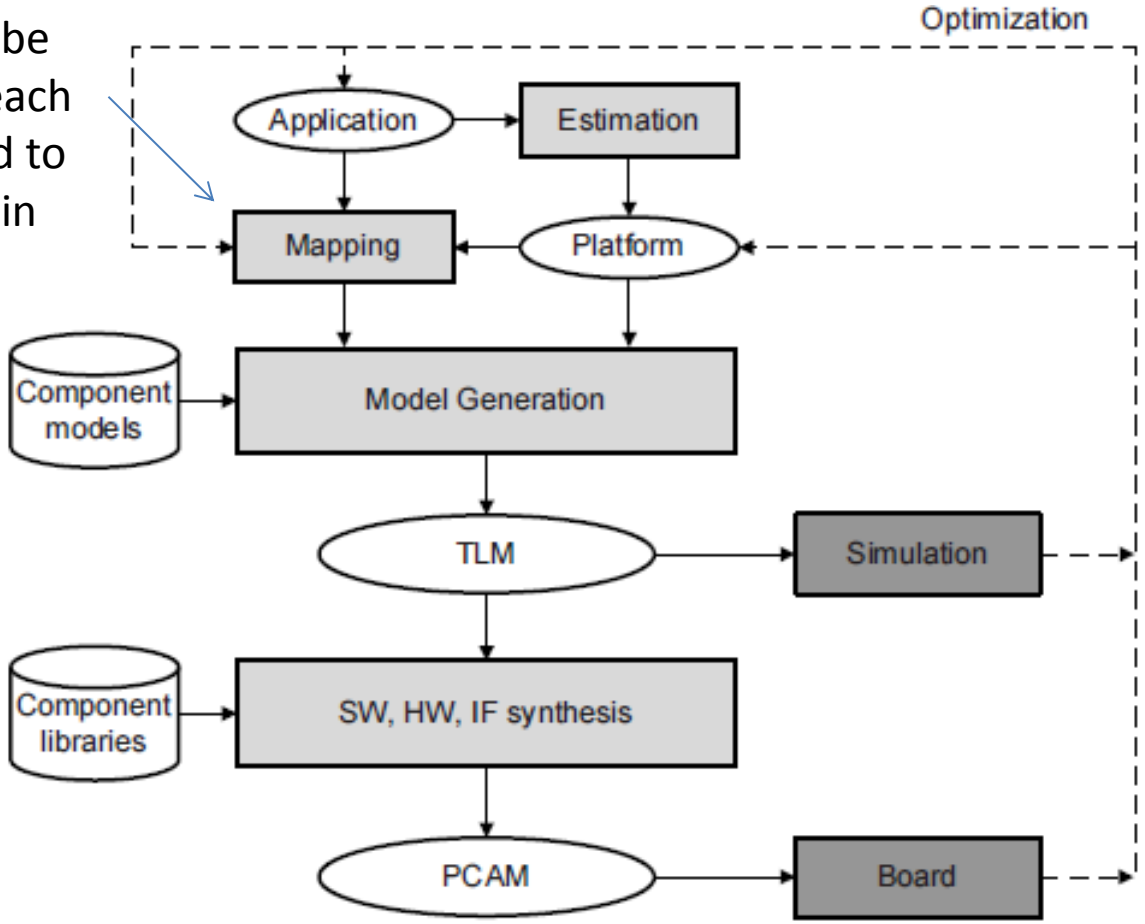
System-level synthesis



The platform can be defined through the estimation of performace, power, reconfigurability, etc

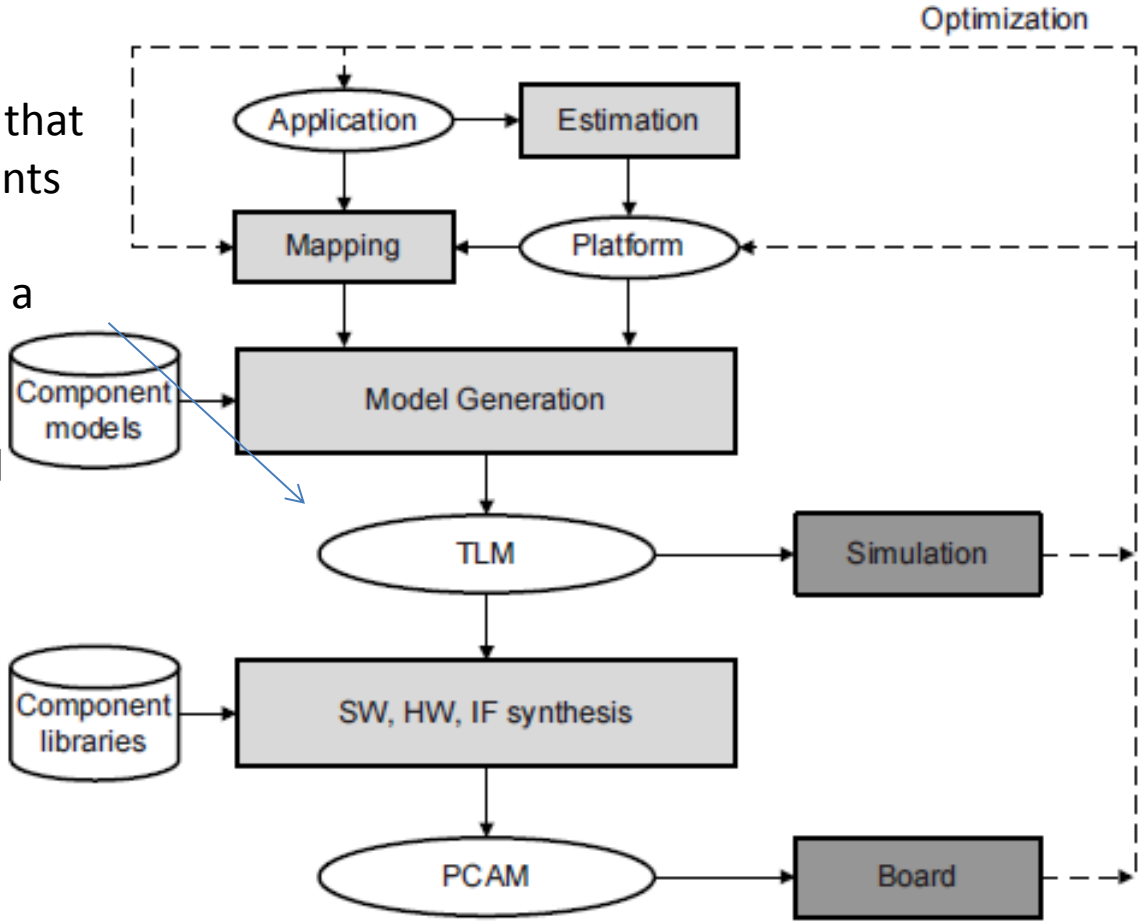
System-level synthesis

Once the platform is defined, an application must be partitioned and each partition assigned to a processor or IP in the platform

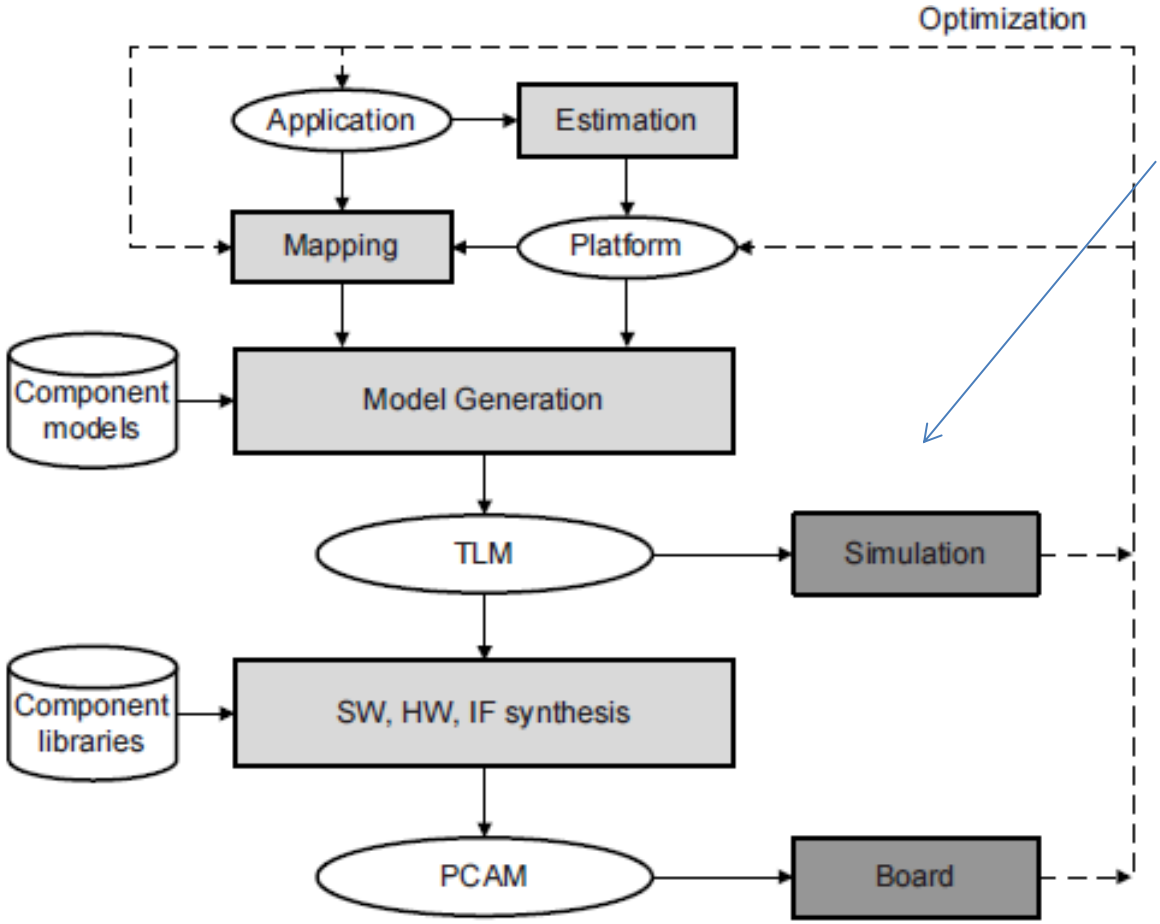


System-level synthesis

In order to verify that all the requirements are met, we need to generate a model such as a timed Transaction-Level Model (TLM)

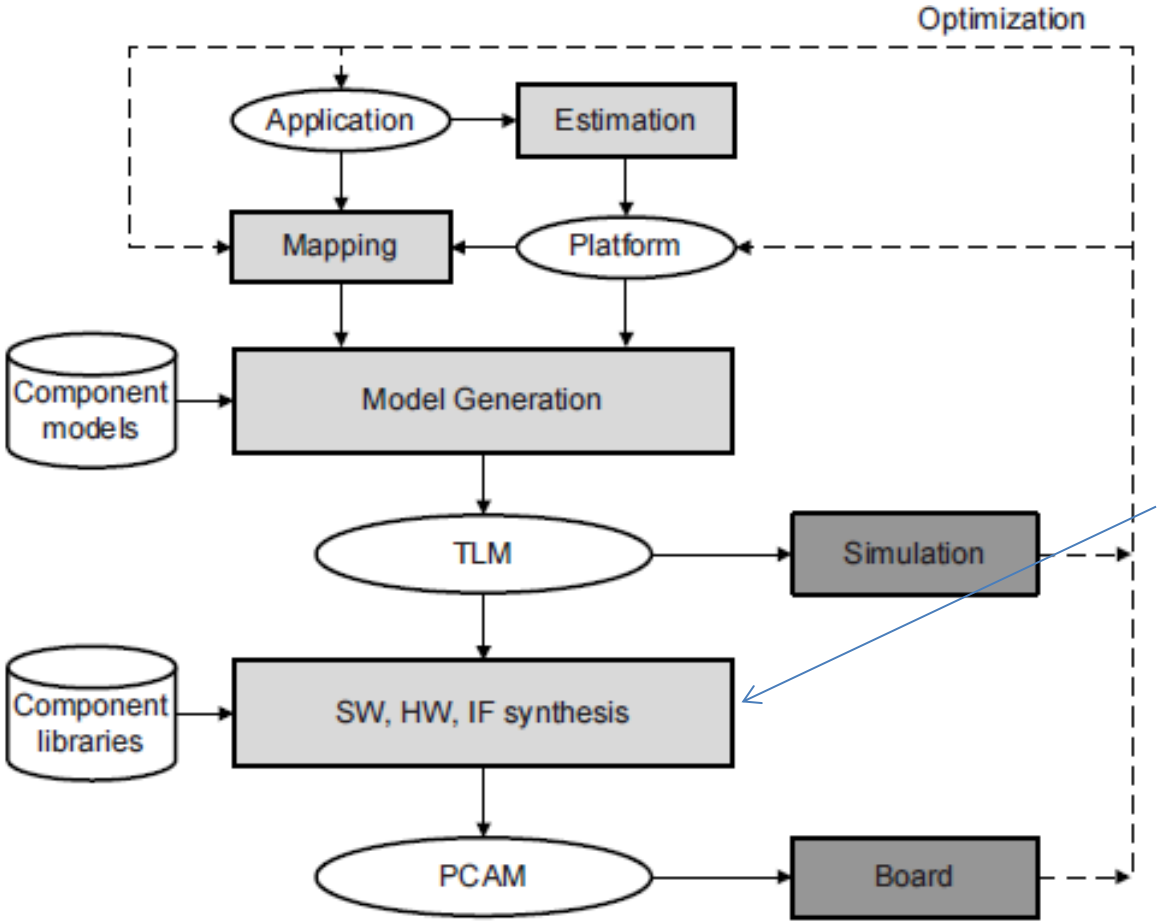


System-level synthesis



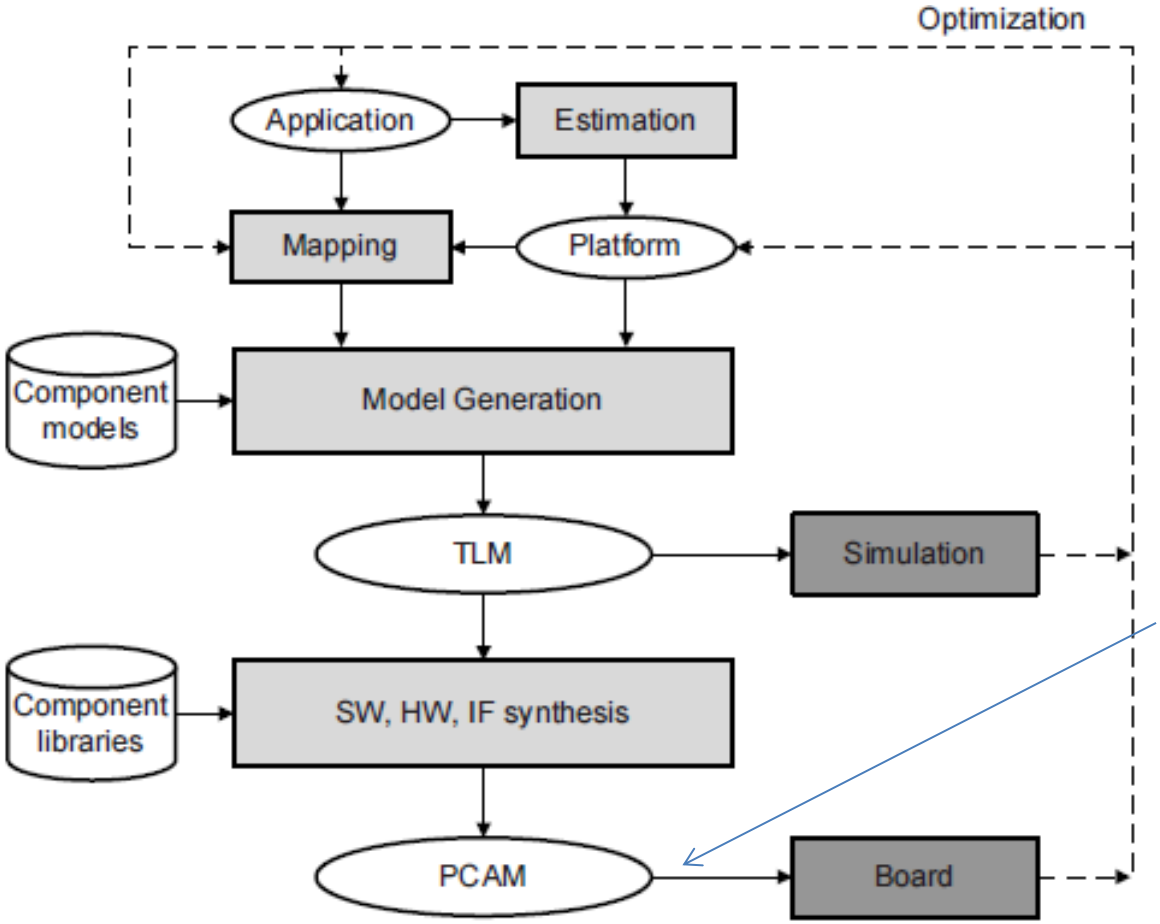
After simulation, the design can be optimized

System-level synthesis



After we obtain a satisfactory application code, platform, and mapping, we can synthesize each component

System-level synthesis



After synthesis, we need to generate a CAM (Cycle accurate model) model that contains binaries for downloading to processors and RTL descriptions for the HW parts in the platform.

Modelling

System behaviour and models can be defined using

- Models of computation (MoC)
- Design languages

MoCs are generally based on a decomposition of behavior into pieces and their relationships in the form of well-defined objects and composition rules.

Process – based: data oriented, concurrent processes
State – based: control-dominated applications

Modelling

System behaviour and models can be defined using

- Models of computation (MoC)
- Design languages

System Design Languages:

Netlist and schematic: structural representations of the design as a set of components and their connectivity. i.e: Electronic Design Interchange Format (EDIF), at gate level. Variant of XML to capture netlist of system platforms.

HDL: To represent not only the netlist and structure of designs but also their behaviour (VHDL, VERILOG)

System-level design language: to model the hardware side of a design and parts of the system implemented in software. (C, C+HDL – System Verilog, SystemC, SpecC)

Processor modelling

Orthogonality between Computation and communication

On the computation side, the basic system component is a processor. The processor executes part of an application.

At the specification level, the application is modelled as a network of communicating processes.
In case of software processor, applications run on top of an OS. Then, a model of OS is needed to introduce accurate representations of the scheduling of parallel processes on the inherently sequential processors.

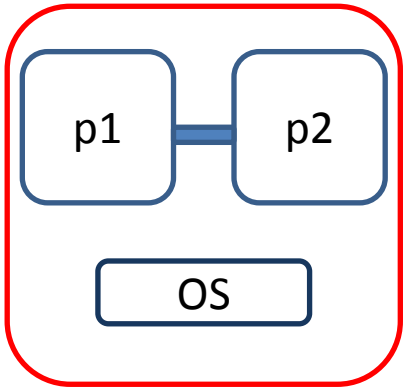
Processor modelling

Computation modelling

At the specification level, the application is modelled as a network of communicating processes.

Processor modelling

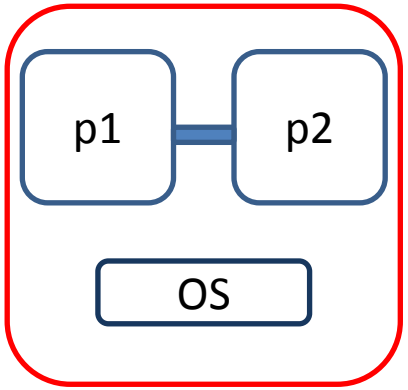
Computation modelling



In case of software processor, applications run on top of an OS. Then, a model of OS is needed to introduce accurate representations of the scheduling of parallel processes on the inherently sequential processors.
The model of the OS provides dynamic scheduling and multi-tasking services

Processor modelling

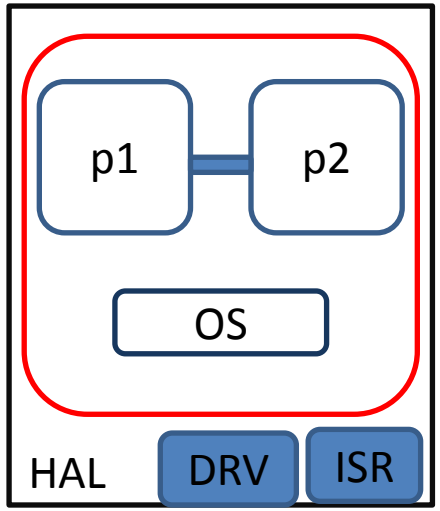
Computation modelling



Application and OS software has to run on top of the actual processor hardware, which realizes physical bus interfaces and interrupts (including processor suspension and interrupt timing) for communication with the external world

Processor modelling

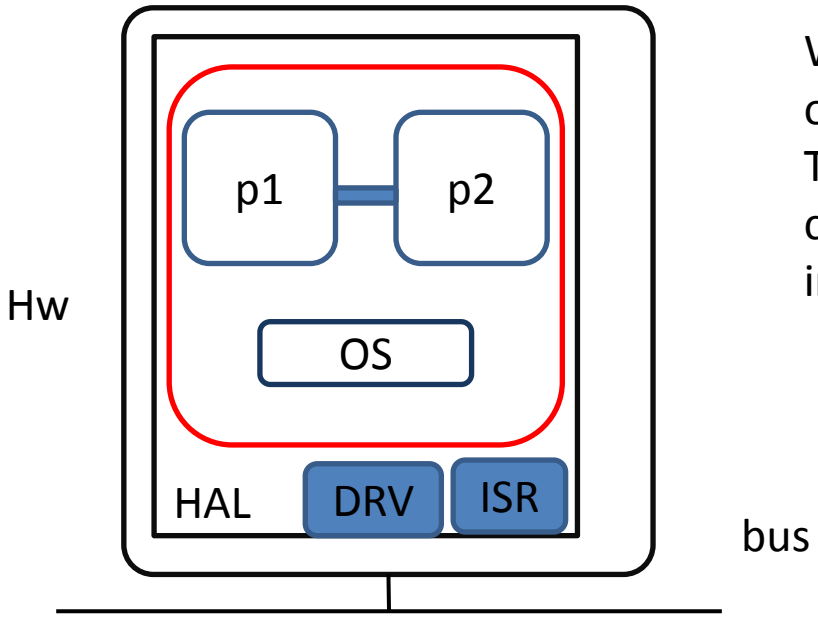
Computation modelling



Then a hardware abstraction layer (HAL) is introduced providing interfaces, such as bus drivers and interrupt service routines (ISRs) for accessing the processor hardware from the software (i.e., application and OS) side.
HAL is the lowest level of functionality implemented in software

Processor modelling

Computation modelling



With the final hardware layer, an accurate model of the actual processor hardware is included. The processor hardware model specifically captures details of physical bus interfaces and of interrupt handling behaviour.

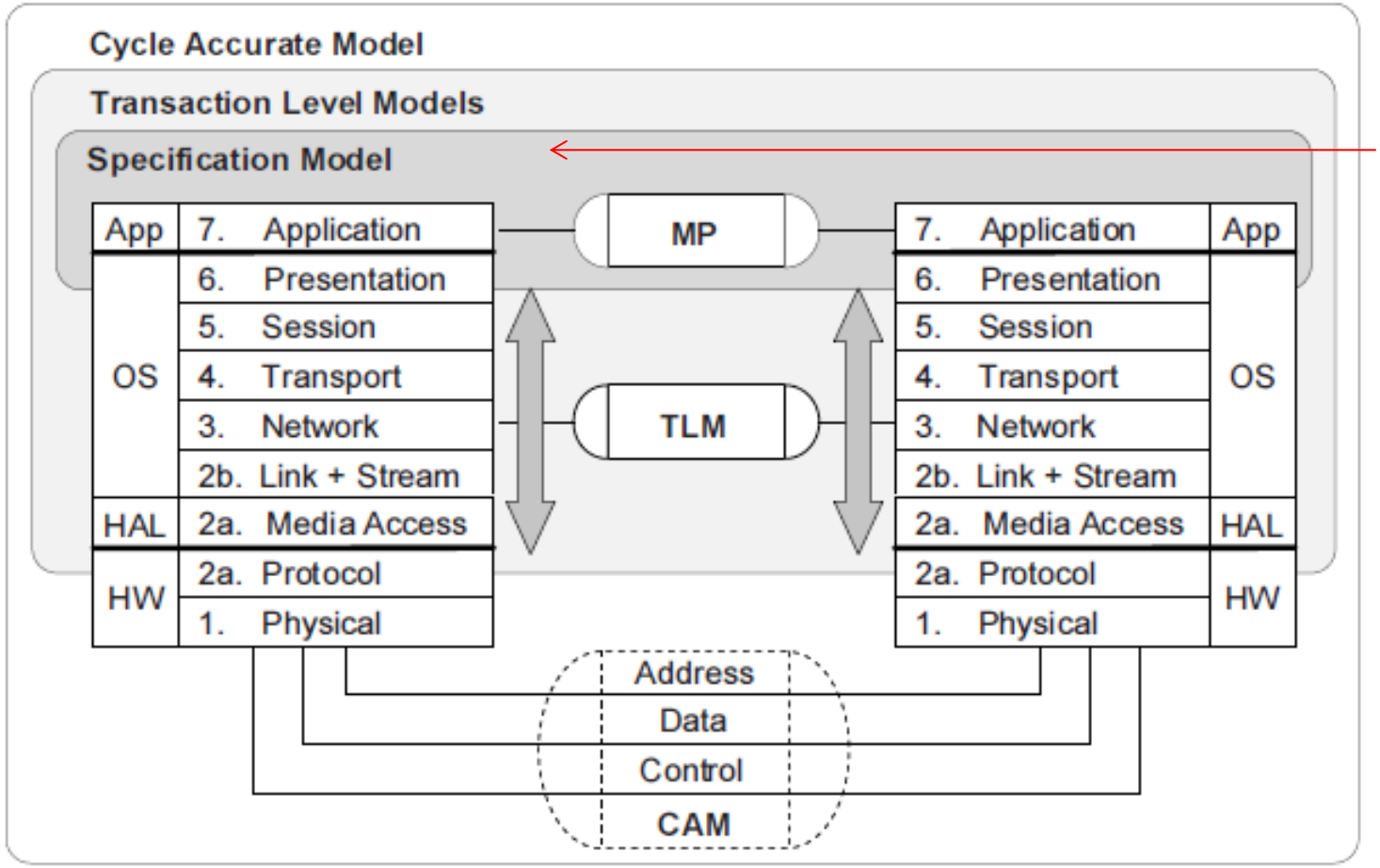
Processor modelling

Communication modelling

Based on ISO/OSI 7 layer model.

Layer	**Functionality**	**Implementation**	**OSI**
Application	Computation	Application	7
Presentation	Data formatting	OS	6
Session	Synchronization, multiplexing	OS	5
Transport	Packeting, flow control	OS	4
Network	Subnet bridging, routing	OS	3
Link	Point-to-point logical links	Driver	2b
Stream	Multiplexing, addressing	Driver	2b
Media access	Data slicing, arbitration	HAL	2a
Protocol Media	Protocol timing	Hardware	2a
Physical	Driving, sampling	Interconnect	1

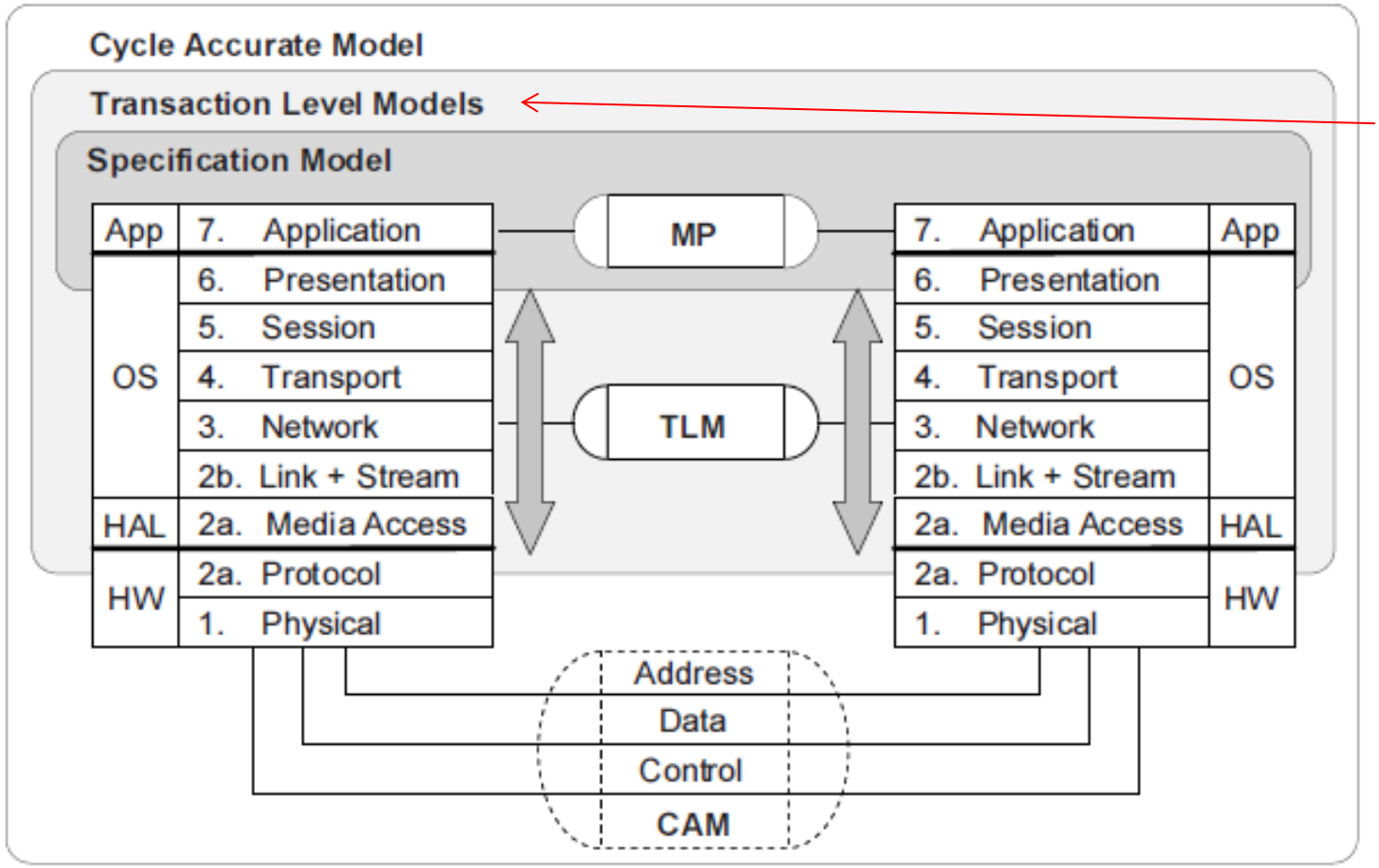
System Model



For application designers

Source:Gajski D.D. et al. Embedded System Design - Modeling, Synthesis and Verification

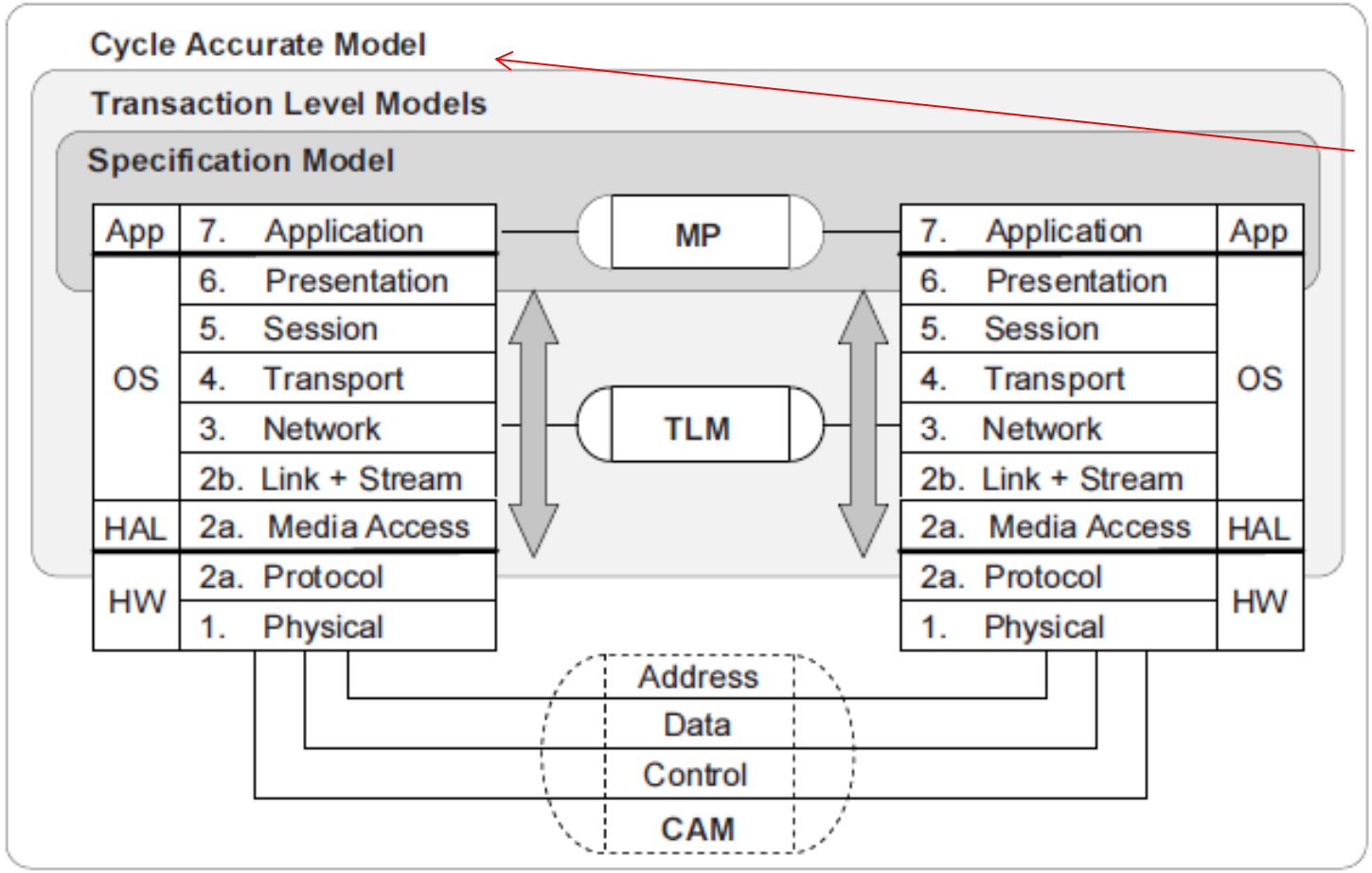
System Model



For system designers

Source:Gajski D.D. et al. Embedded System Design - Modeling, Synthesis and Verification

System Model



For implementation designers

Source:Gajski D.D. et al. Embedded System Design - Modeling, Synthesis and Verification

Thank you and
Good luck!