Design and Verification in the ForSyDe Methodology

Tarvo Raudvere
Royal Institute of Technology
tarvo@imit.kht.se

Motivation for ForSyDe

- Trends
  - Increasing capacity of integrated circuits allows to integrate more and more functions on a single chip
  - SoC-architectures may include a variety of components: microprocessors, DSP cores, memories, custom hardware and analog parts
- Conclusion
  - Simulation is not enough
  - System-level design methodologies have to incorporate formal methods

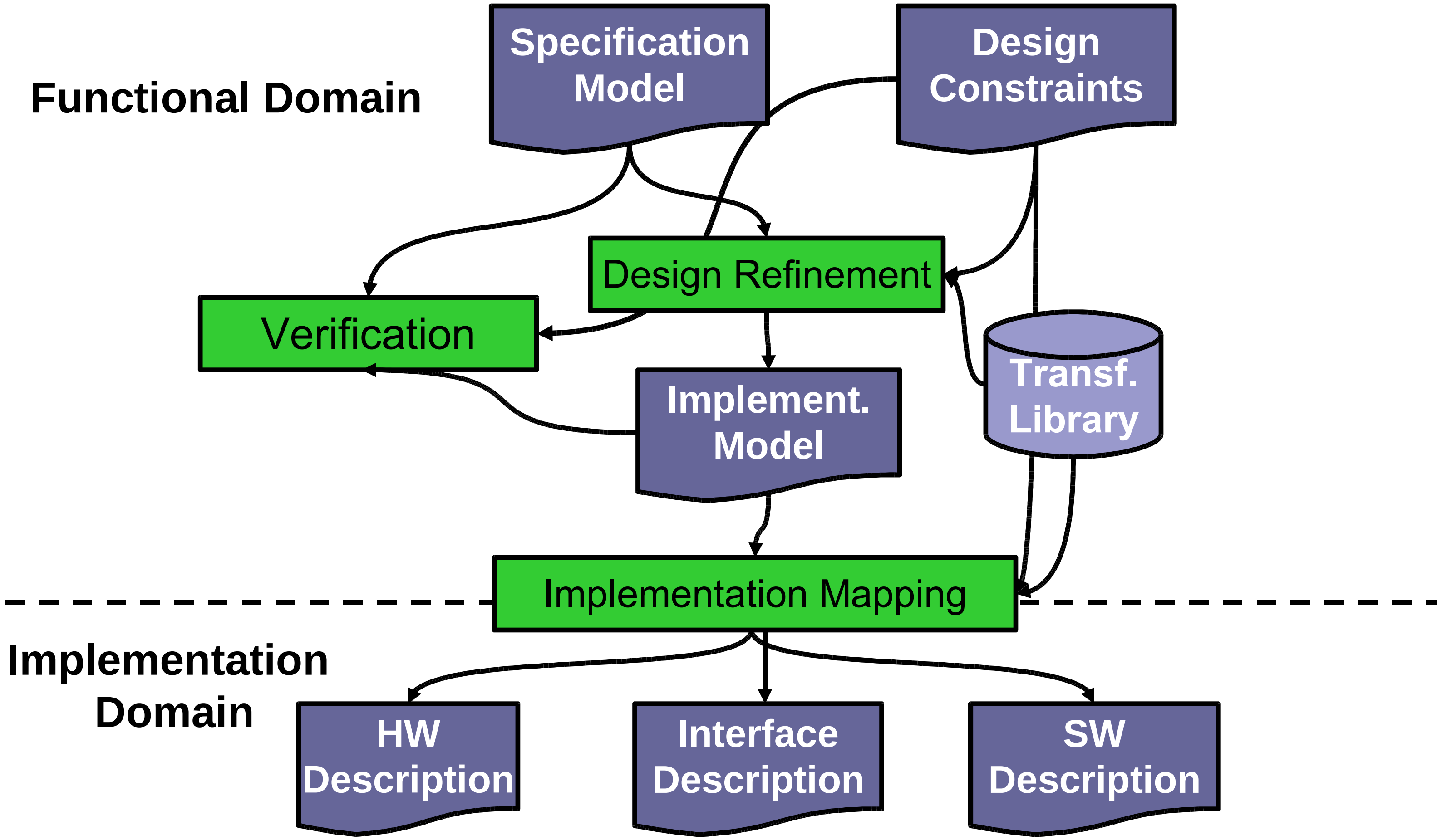
Outline

- Motivation
- Design Flow
- Design Refinement
- Gradual Design Verification
- Verification Example
- Conclusion

ForSyDe (Formal System Design)

- Objective
  - Addresses the design of reactive system-on-chip applications with control and data flow parts
- Foundations
  - Start with a purely functional and deterministic system specification
  - Uses a synchronous model of computation
  - Uses a functional modeling language with formal semantics
  - Allows for formally defined design transformation
  - Allows to interpret the system model into a hardware and software interpretation

The ForSyDe Design Flow

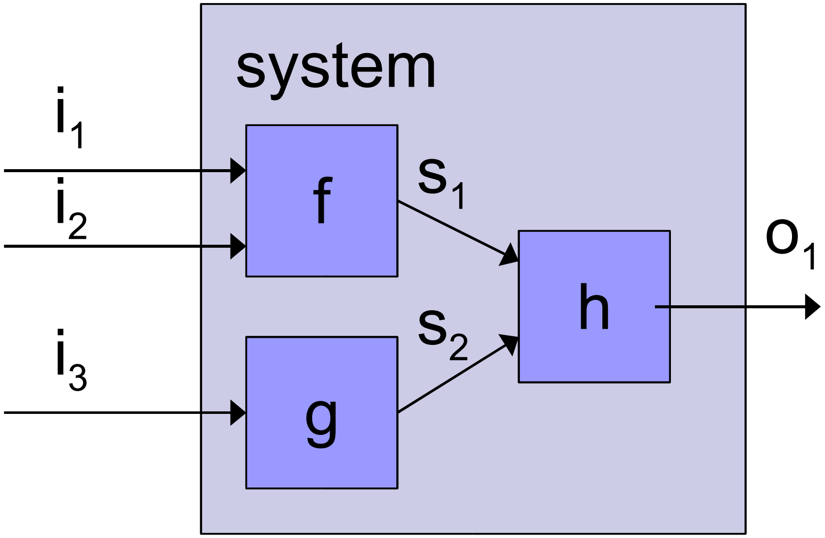


Specification Model

- Is based on a synchronous model of computation
- Is purely functional and deterministic
- Uses process constructors
- Uses possibly infinite/ideal data types
- Implies a concurrent process model
- Can be refined during design transformations

Functional Specification

- A system is specified as a function of the system inputs
- A function can be composed of the other functions
- A function has no side effects
- There is no global state
- System inputs are signals
- A functional system specification is deterministic
- Concurrency is implicit



o_1=system (i_1,i_2,i_3)

where

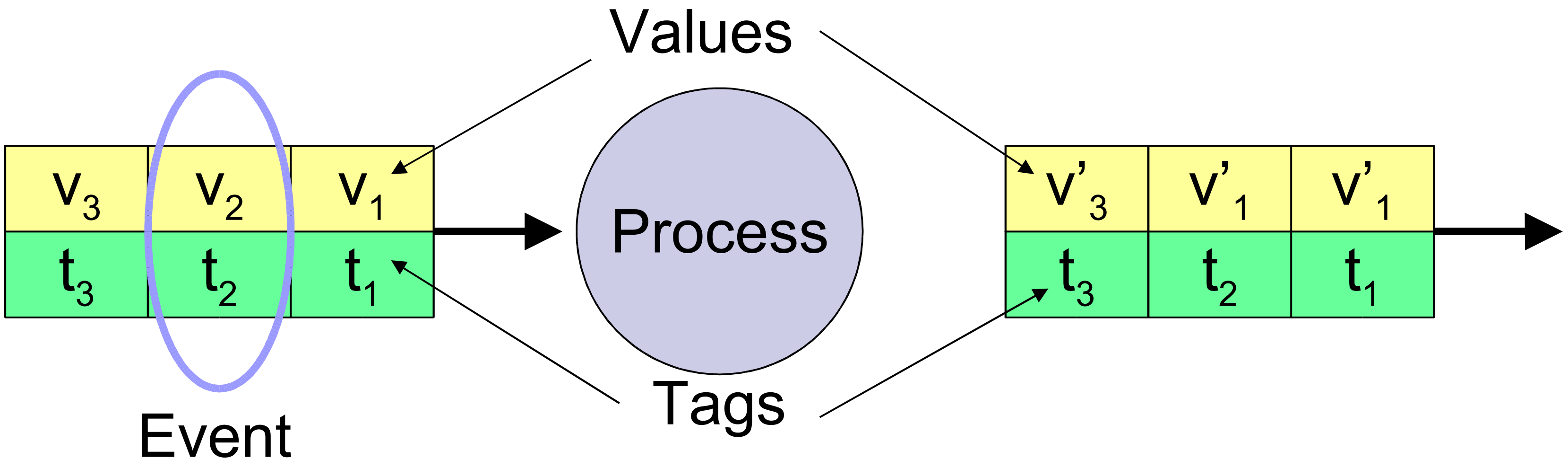
$s_1 = f\ (i_1,i_2)$

$s_2 = g\ (i_3)$

$o_1 = h\ (s_1,s_2)$

Synchronous Assumption

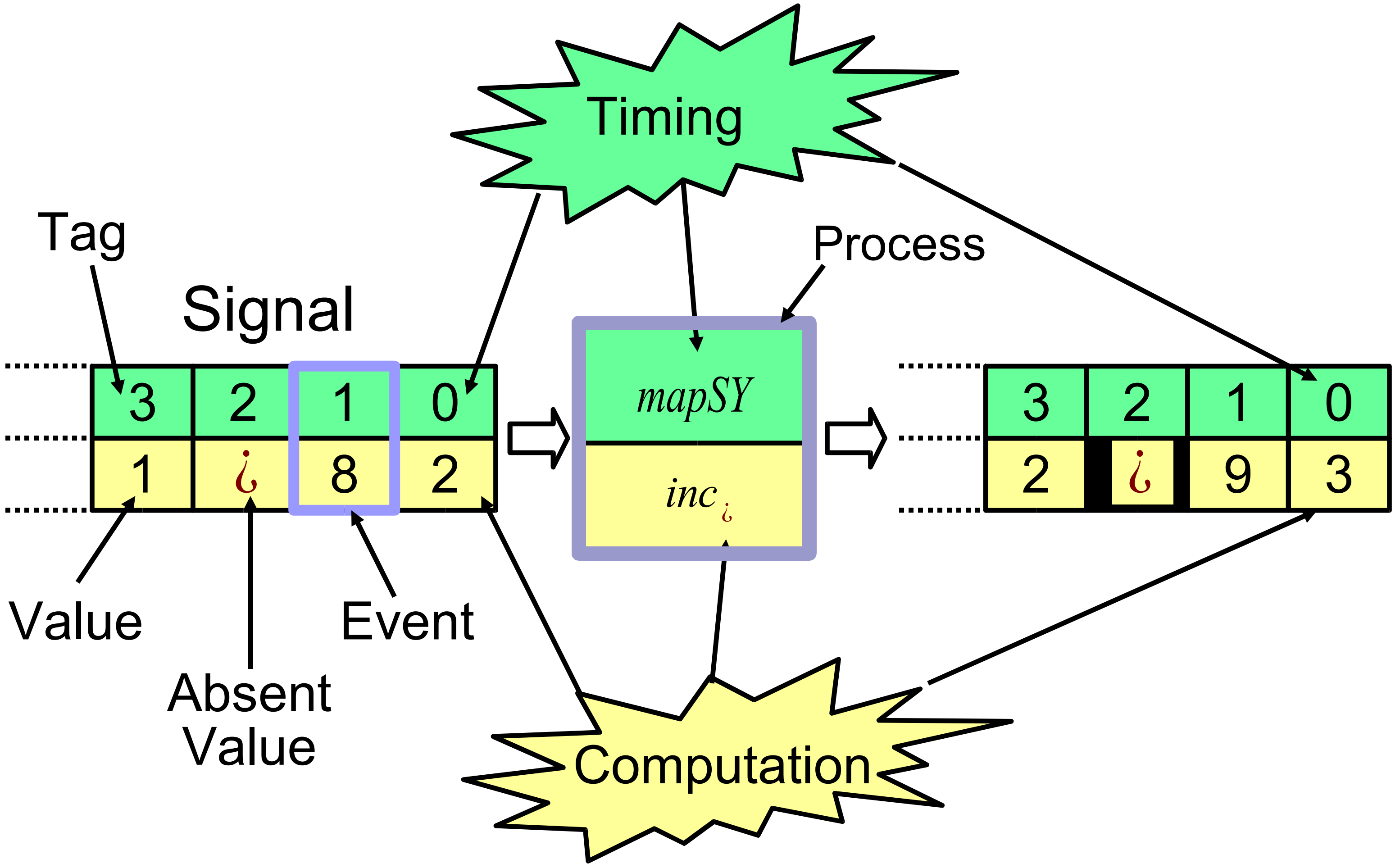
- The system computes infinitely quickly
- Each reaction divides time into a sequence of discrete instants
- A system reaction to an input appears at the same time as the input
- The synchronous assumption leads to a clean separation between communication and computation



Process Constructor

- is a higher-order function
- is used for synchronization
- has a hardware and software interpretation
- allows for design transformations
- is used to create processes

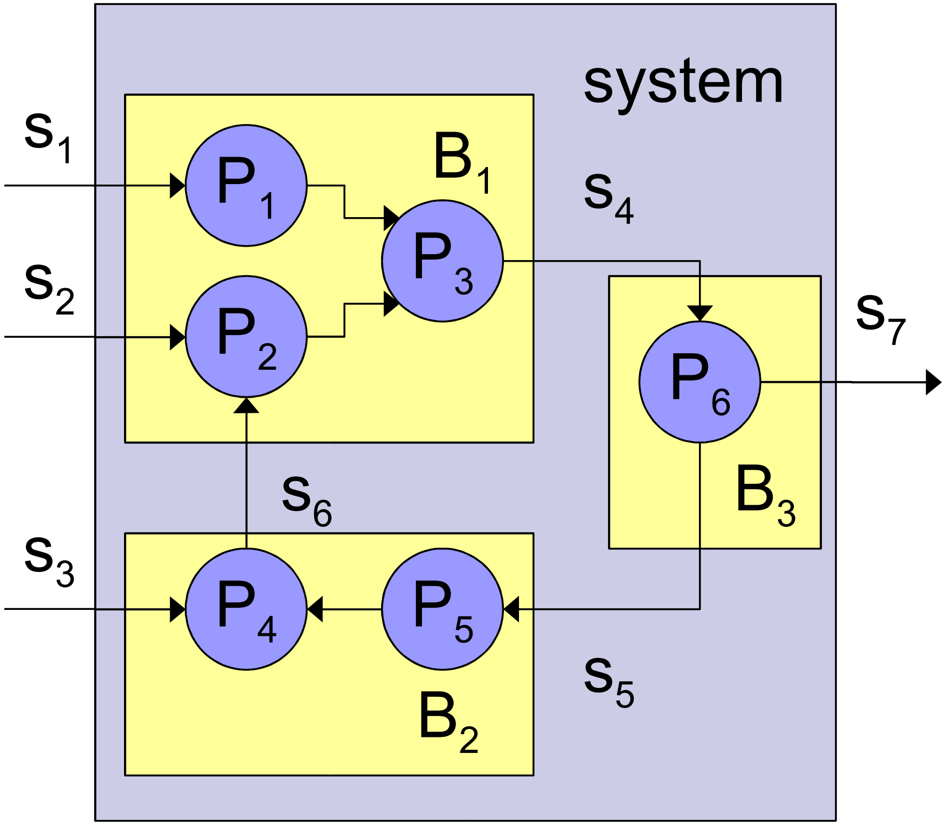
Process Constructor mapSY



3	2	1	0
1	¿	8	2

3	2	1	0
2	¿	9	3

System as a process network



system $(s_1,s_2,s_3) = s_7$

where

$s_4 = B_1 (s_1,s_2,s_6)$

$s_6 = B_2 (s_3,s_5)$

$(s_7,s_5) = B_3 (s_4)$

$B_3 (i_1) = (o_1,o_2)$

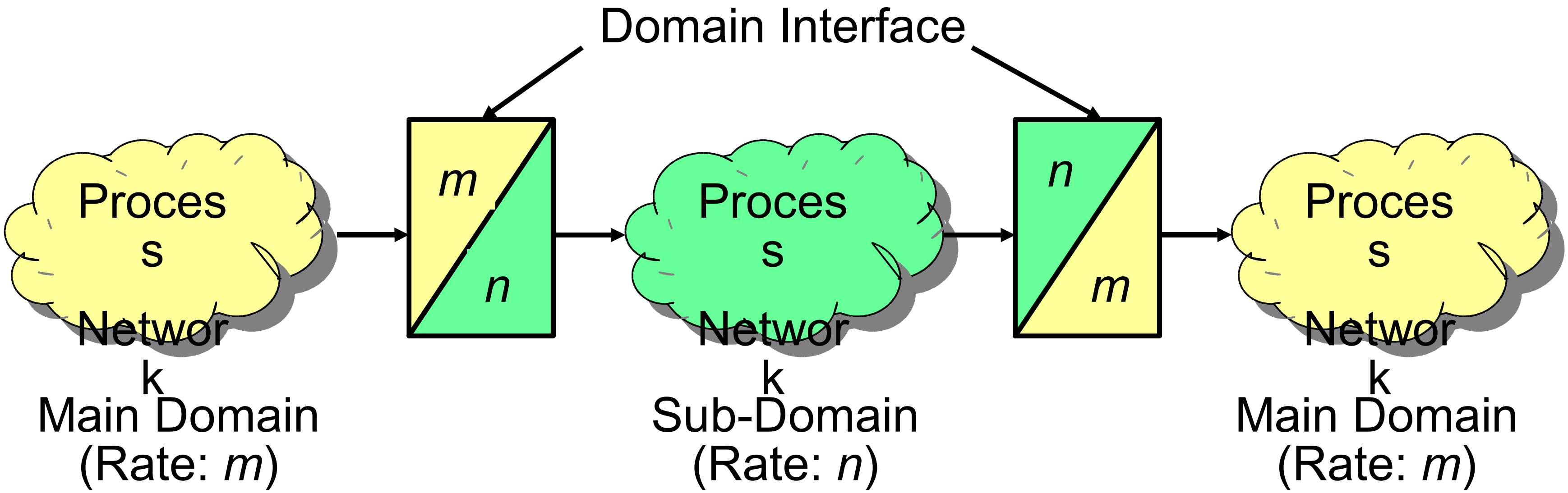
where

$(o_1,o_2) = P_6 (i_1)$

P_6=mealySY(f,g,m_0)

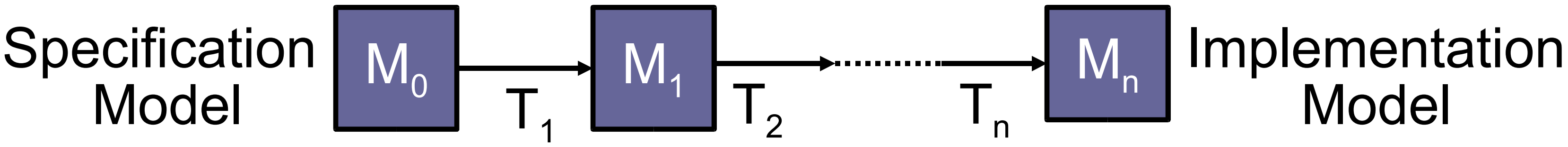
f::Int->Int->Int

Implementation Model



- may contain synchronous sub-domains
- is described in terms of limited resources
- can be mapped to hard- and software (VHDL,C)

Refinement of the Specification Model



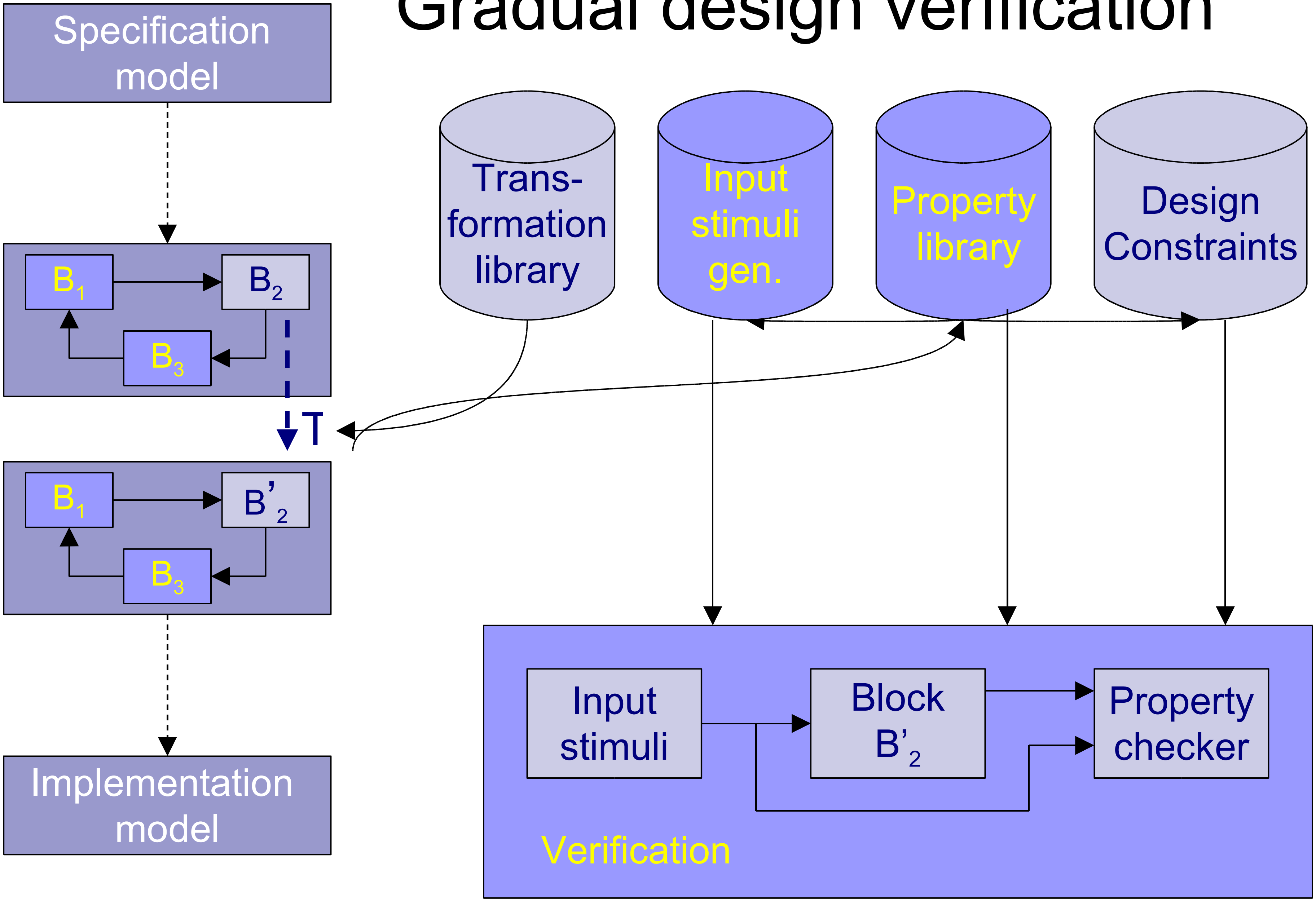
The specification model (M_0) is stepwise refined by the use of well defined design transformation (T_i) into an implementation model (M_n)

- Semantic Preserving Transformation
  - Do not change the meaning of the model
  - Used mainly for process merging and splitting
- Design Decision Transformation
  - Change the meaning of the model
  - Introduce a design decision
  - Examples are refinement of data types, constraining buffer sizes

Verification in ForSyDe

- Global properties are verified by simulation
- Local properties of design blocks can be checked through model checking
- Design transformation contains information about the changes in the model, that can be used to create relevant verification properties

Gradual design verification



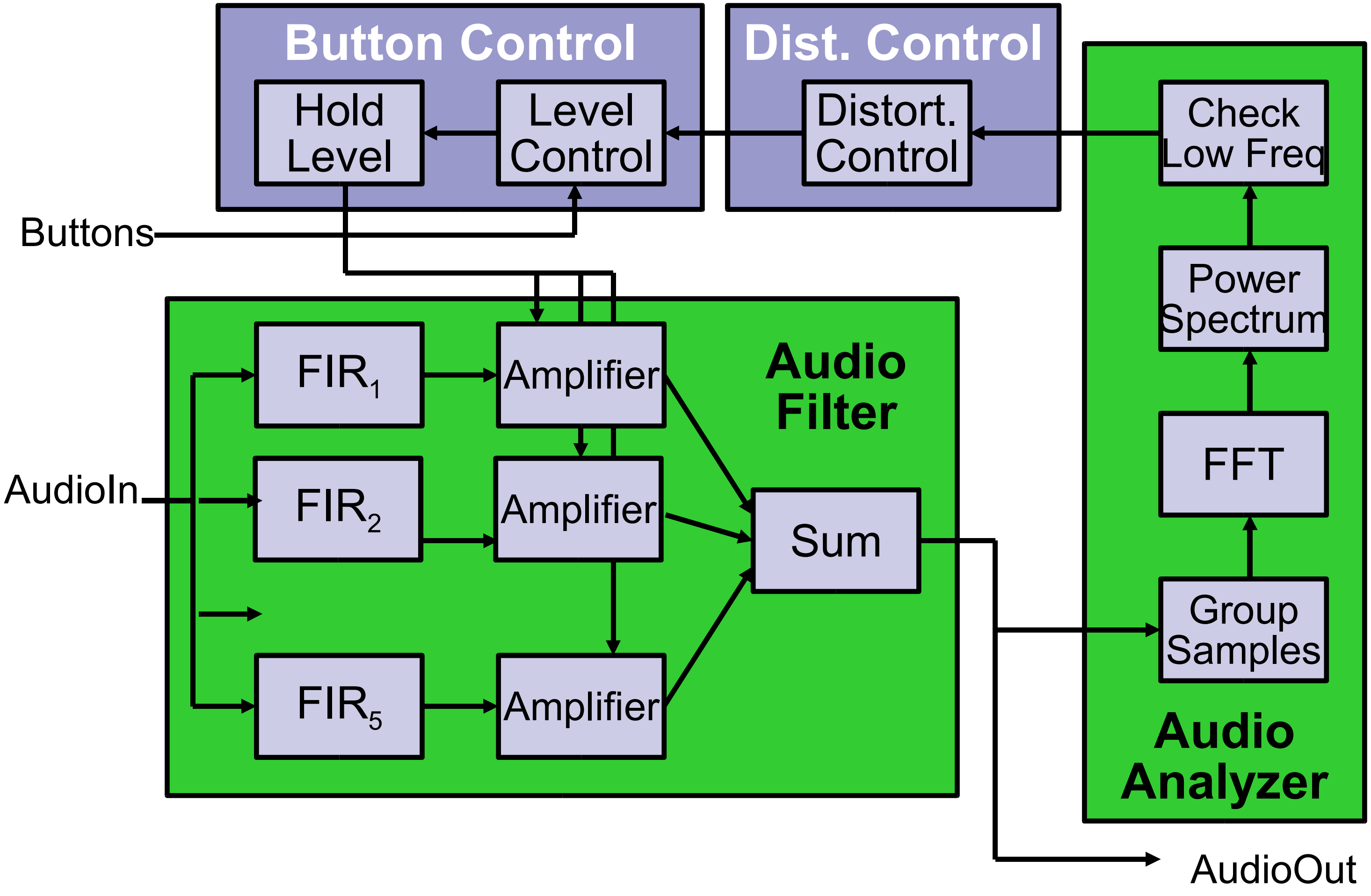
Assumptions for gradual verification

- The specification model is correct
- Only the local correctness of the system is verified through model checking
- The designer is aware of the constraints for every design block
- Design blocks are small enough, that existing verification tools manage the tasks in reasonable time

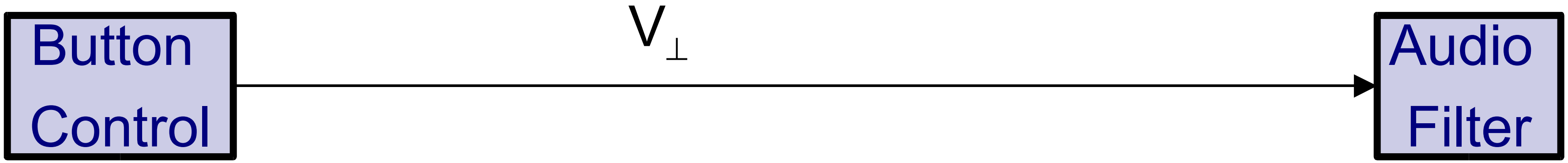
Verification Details

- For every design transformation there is a set of predefined properties
- There are templates to assist the user to model the design environment for design blocks
- Abstraction has to be done by the user
- The Cadence version of SMV (Symbolic Model Verifier) is used for verification
- Translation from ForSyDe into SMV is straight forward

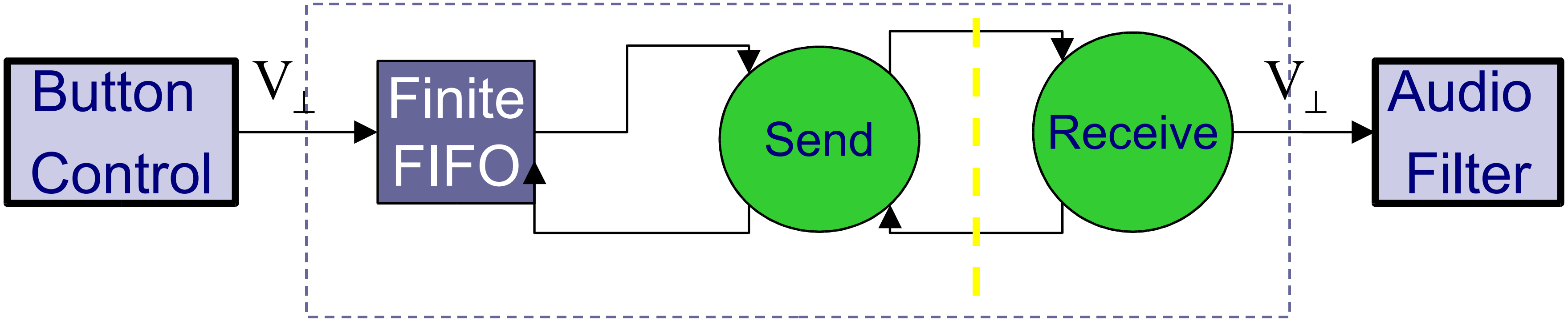
Refinement of the Equalizer System Model



Refinement into a Handshake Protocol



Transformation: *ChannelToHandshake*



- The handshake protocol introduces a delay between Send and Receive
- The FIFO buffer stores the input data if the channel is busy with the previous arrival
- The FIFO size must correspond to the load on the channel

Verified properties

Reliability: The only data loss is caused by overflow. (1 sec.)

Latency: It takes a fixed number of clock cycles to transport data through the channel. (0.2 sec.)

Bandwidth: An input stream with a certain data rate causes no FIFO overflow. (0.2 sec.)

Order: Present values on the channel output preserves the same order they have on the input. (400 sec.)

Requirements

- For every new design transformation a set of properties has to be defined, which are obligatory to verify
- In order to avoid *state space explosion,* abstraction has to be applied.
- The proper abstraction technique should be selected according to:
  - specific design transformation
  - the properties that we verify

Conclusion

- The ForSyDe methodology supports formal system design
- Design flow starts at a high abstraction level
- Design is refined through the well-defined design transformations
- Implementation can be mapped to soft- and hardware
- Verification is applied during the design refinement

- http://www.ele.kth.se/ForSyDe

Thanks for your attention!