Convertibility Verification and Converter Synthesis

Roberto Passerone
University of California at Berkeley
Cadence Design Systems, Inc.

Jerry R. Burch
Synopsys, Inc.

Luca de Alfaro
University of California, Santa Cruz

Tom A. Henzinger
Alberto L. Sangiovanni-Vincentelli
University of California, Berkeley

Outline

- Motivations
- Interface verification
- Correctness specification
- Converter synthesis
  - Automata based
  - Game-theory based
- Summary and Conclusions
Motivations

- **Re-use strategy critical for cost and time-to-market**
- **Systems assembled from internal and third party IPs**
- **Correctness of composition must be verified**
  - Costly simulations may still miss problems
  - Safety critical applications require a formal correctness proof
- **Abstract component models used to specify the requirements**
  - Transaction Level Models shorten time-to-verification
  - Standards used to simplify the problem
- **Formal proofs usually based on type systems**
  - Typically only limited to static information

Behavioral Types

- **Define the protocol of interaction**
  - Includes dynamic behavior as well as static typing information
- **Distinguishes I/O behavior so that**
  - it defines assumptions on the accepted inputs,
  - it provides guarantees on the generated outputs
- **Compatibility defined**
  - Two IPs are compatible if the output guarantees of one satisfy (or imply) the input assumptions of the other
Example

Data partitioned into two parts: a and b

Producer
Send b immediately after a

Consumer
Possibly wait between a and b

Observations

- The problem of checking compatibility can be set up as a game
- Here reduced to checking trace containment
  - Producer Outputs \( \subseteq \) Consumer Inputs
- For open systems the procedure must include the environment
  - Helpful environments are used to decide compatibility and to compute the input assumptions and output guarantees of the composite
- Symbols are used to represent data
  - Data must be represented explicitly when the protocol depends on the values
- Some mechanism in the implementation must signal whether a or b is being transferred
  - We don’t need to be specific at this level of description
  - Any mechanism will do (toggling bits, additional signal, etc.)
Example revisited

Consumer
Must receive b immediately after a

Producer
Possibly wait between a and b

Data partitioned into two parts: a and b

Compatibility

- The protocols are incompatible
  - Direct connection leads to (possible) failure
- The interaction can be mediated by an adapter
  - Potentially makes the system globally compatible
- Compatibility redefined
  - Two IPs are compatible if the output guarantees of one can be used to satisfy the input assumptions of the other
- There are many possible adapters
  - Liberally generate legal transactions on the receiver side and accept all transactions on the producer side
  - Probably not what we want!
- Need a strategy to design a correct adapter
  - Need to understand what the word “correct” really means
Converter Synthesis

- **Borriello et al, 1988**
  - Timing diagram based
- **Narayan et al, 1995**
  - Language based
- **Passerone et al, 1998**
  - Automata based
- **Smith et al, 1998**
  - FIFO based
- **In all cases the semantics of a correct conversion is embedded in the algorithm**

Correctness Specification

- **Extend converter synthesis with a correctness specification**
  - Provides the notion of compatibility
- **Correctness embodied by a transaction monitor**
  - Defines the correct interactions
  - Monitors signals from both the producer and the consumer

October 2, 2003
Observations

- **The converter must conform to the correctness specification**
  - But the specification does not define how the conversion should be done

- **Example of specification**
  - No symbol should be discarded or duplicated
  - Symbols must be delivered in the order in which they are received
  - Only one symbol can be in flight at any time

- **But does not require that, for example**
  - b follows a, and a follows b

---

**Example**

No data in transit

- "a" transmitted but not received
- "b" transmitted but not received
Outline

- Motivations
- Interface verification
- Correctness specification
- Converter synthesis
  - Automata based
  - Game-theory based
- Summary and Conclusions

Converter Synthesis

- **Start from the product of the interacting protocols**
  - Most general form of the converter
  - It adapts the producer and consumer protocols without synchronization

- **Make the converter conform to the specification**
  - Must remove transitions from the product that are not allowed by the specification

- **Ensure that the converter is responsive (receptive) to the producer protocol**
  - It must accept all possible transactions
Product Computation

Conformance to Specification

Converter Specification
Outline

- Motivations
- Interface verification
- Correctness specification
- Converter synthesis
  - Automata based
  - Game-theory based
- Summary and Conclusions
Game theoretic formulation

Game played between the protocols and the specification on one side, and the converter on the other.

Game structure

Transition system such that each state
- gives the available moves for the producer,
- gives the available responses for the converter

Some states in the game structure have an empty set of available responses
- They correspond to the illegal states in the product machine
Playing the game

- Player 1 starts the game by choosing a move available from the producer.
- Player 2 responds with a move allowed by consumer and specification.
- The game transitions to a new state given the two moves.

Winning the game

- **Winning the game**
  - Player 1 wins if it can steer the game to a state where Player 2 (the converter) has no moves.
  - Player 2 wins if it can always steer the game to a state where it has moves.
  - Players can play according to a strategy.
- **A converter is a winning strategy for Player 2**
  - If a winning strategy does not exist, then the protocols are incompatible.
  - Game solved via traditional game theory results.
  - Complexity linear in the size of the game structure.
Game theory: advantages

- **Game theory a more general basis for the definition of the problem**
  - The approach is abstract and generic
  - Can easily be extended to multi-player scenarios
  - Limited information scenarios also studied in the literature

- **Generalizes to more expressive specifications**
  - Can add fairness constraints without changing the theory
  - Omega-regular games are well studied
  - Computational complexity increases

- **Tools for solving games already available**

Fairness Example
Applications

What are the potential applications?
- Composition verification
- Protocol conversion
- Domain conversion/mix-mode simulation
- Design of communication independent IPs
- Test bench generation (master and slave)
- Mixed transaction/signal level simulation for accuracy/performance tradeoffs
- Stack layer synthesis
- Bus bridge synthesis

Summary

Compatibility rephrased in terms of the existence of an adapter
- Interface verification requires synthesizing the converter

Correctness expressed in terms of a specification
- Reordering, buffering, latency, etc.

Converter synthesis extended to account for the specification
- Synthesis problem cast and solved as a game
- Game theory a more general basis for formulating the problem

Current work
- Exploring formulation for both synchronous and asynchronous models using trace theory
- Generalization to different models of computation
- Tool support