Interface Synthesis: Convertibility Verification and Converter Synthesis
Outline

- Motivations
- Interface verification
- Correctness specification
- Converter synthesis
  - Automata based
  - Game-theory based
  - Trace-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

Producer
Send b immediately after a

Consumer
Possibly wait between a and b

Data partitioned into two parts: a and b
Observations

- The problem of checking compatibility can be set up as a game
- Here reduced to checking trace containment
  - Producer Outputs ⊆ 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

Data partitioned into two parts: a and b

Producer
Possibly wait between a and b

Consumer
Must receive b immediately after a
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
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
Motivations

Interface verification

Correctness specification

Converter synthesis
  – Automata based
  – Game-theory based
  – Trace-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

T' \quad b' \quad a'

T/T' \quad T/a'

T/b' \quad b/a'

a/a'

a/T' \quad b/T' \quad b/b'
Conformance to Specification

Converter

Specification
Final converter

T' b' a'

T/T' b/a' a/b'

T/T' a/T' T/b'

T b a
Outline

- Motivations
- Interface verification
- Correctness specification
- Converter synthesis
  - Automata based
  - Game-theory based
  - Trace-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.
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

- **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

\[ (T, T') \ (a, a') \ (b, b') \]

\[ (T, a') \ (a, T') \ (T, a') \ (b, T') \]

\[ (a, a') \ (T, T') \ (a, a') \ (b, a') \]

\[ (b, a') \ (b, a') \ (b, b') \ (b, b') \]

\[ \text{October 22, 2009} \]
Outline

- Motivations
- Interface verification
- Correctness specification
- Converter synthesis
  - Automata based
  - Game-theory based
  - Trace-theory based

Summary and Conclusions
Receptiveness and failures

- The models described so far are not receptive
  - This is intended to constrain the behaviors of the environment to only those that are “acceptable”
  - This is unlike, for example, I/O Automata
- We would like to recover receptiveness by explicitly modeling the occurrence of a failure
- Dill’s trace structures
  - A trace is either a success or a failure
  - A trace structure contains a set of success traces and a set of failure traces
  - Trace structures must be receptive
Example of failures

Receiving an a or b at the wrong time causes a failure
Failures and composition

A trace structure that has no failures is said to be “failure-free”

A trace structure that has failures can still be used!
- It is enough to compose it with an environment that does not excite the failure
- We also refer to them as “helpful” environments

Successes and failures thus implicitly function as the input requirements and the output guarantees of a behavior type
- We can use the property of failure-freedom to define the notion of satisfaction of a specification
T conforms to T' if and only if, for all possible environments E
- if T' makes E failure free
- then T makes E failure free
Conformance

T conforms to T' if and only if, for all possible environments E
- if T' makes E failure free
- then T makes E failure free
T conforms to T’ if and only if, for all possible environments E
– if T’ makes E failure free
– then T makes E failure free
Checking conformance involves considering all possible environments
  – Too complex

Conformance can be characterized by a single trace structure
  – The maximal environment that makes the composition failure-free
  – This environment is called a mirror

Result
  – \( T \preceq T' \) if and only if
  – \( T \parallel \text{mirror}(T') \) is failure-free.
Result

- $T \leq T'$ if and only if
- $T \parallel \text{mirror}( T' )$ is failure-free.
Conversion as Rectification

\[
(a \ b) \xrightarrow{\text{Correctness Specification}} (a' \ b')
\]

\[
\begin{align*}
\text{Producer Protocol (handshake)} & \quad \rightarrow \quad \text{Converter} \\
\text{Consumer Protocol (serial)} & \quad \rightarrow \quad (a' \ b')
\end{align*}
\]

\[
C \leq \text{mirror}( \ H \ || \ S \ || \ \text{mirror}( \ \text{spec} ) \ )
\]
General formulation

- Experimented with Dill’s trace theory verifier
  - Applicable to both synchronous and asynchronous systems

- Generalized trace theory to arbitrary models of computation
  - The model must satisfy the axioms of trace algebras
  - The axioms provide the necessary assumptions to prove the rectification in a more general setting

- Future research
  - Models as algebras can be related by homomorphisms
  - Considering rectification across models of computation
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
Tool support

Tool support is important!
- Have demonstrated a prototype in 1998
- Have been focusing mostly on the theory

Need complementary technologies
- Shimizu et al. presented monitor specs for protocols
- Siegmund et al. presented work on transaction based verification in SystemC based on regular expressions
- Need to put all these technologies into a coherent framework for IP-based design
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
Abstract Correctness Specification

Producer Protocol (handshake) (a b) 

Converter (synthesized) 

Consumer Protocol (serial) (a’ b’)

Correctness Specification (transaction monitor) (a b) (a’ b’)

October 22, 2009