Communication Validation
CS 4271 Lecture 7

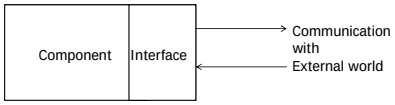
Abhik Roychoudhury
National University of Singapore

http://www.comp.nus.edu.sg/~abhik/

What is it about?

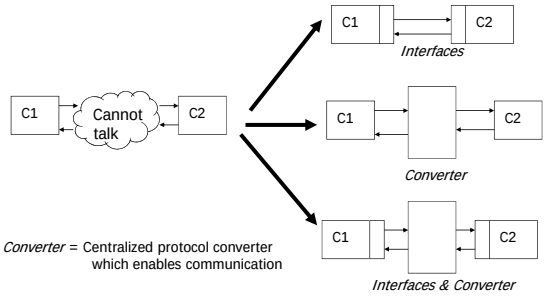
- Communication compatibility among different components of an embedded system.
- Is it that important?
  - Pieces of a complex embedded system may be taken from off-the-shelf components.
  - Communication protocol in each component may be different.
  - Need a protocol converter to allow communication
  - YES !

Notion: Interface



- Two views of interface
  - Passive entity: provides pin-mappings!
  - Active entity: a separate process in charge of communication on the component's behalf.
    - Control flow of its own.

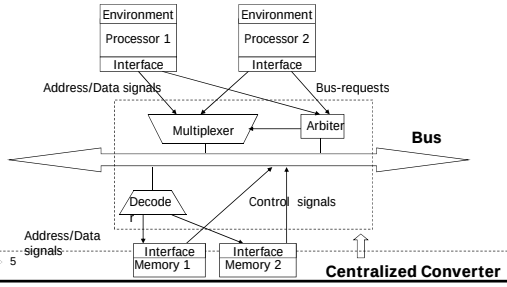
Enabling communication



Converter = Centralized protocol converter which enables communication

Why converter *and* interfaces?

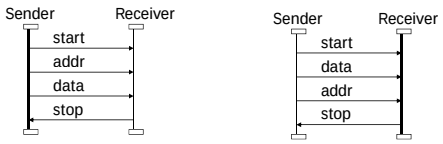
- Bus-based communication - Each component hooked to the bus, has its own bus interface.
  - Common in system-on-chip bus protocols such as AMBA (in ARM).
- Bus controller or arbiter acts as the central converter, enabling communication among the bus interfaces of the components.



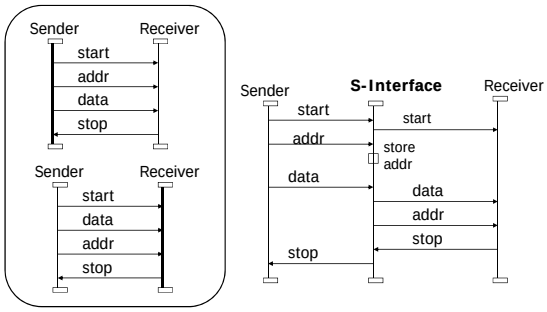
Organization

- Task of converter/intrafaces
  - Enable communication.
  - Resolve protocol incompatibilties.
- What are the common incompatibilties?
- Converter synthesis

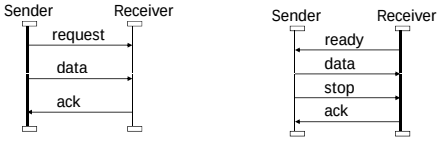
1. Signal ordering mixed up



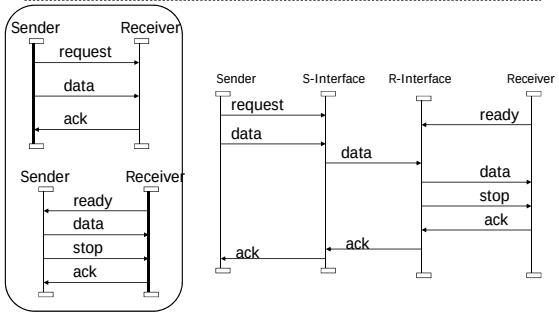
Signal ordering mixed up

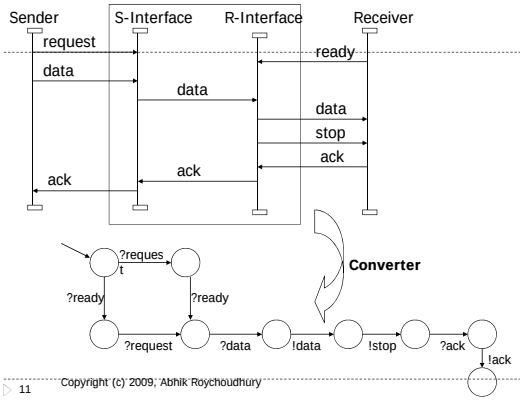


2. Different Signal Alphabet

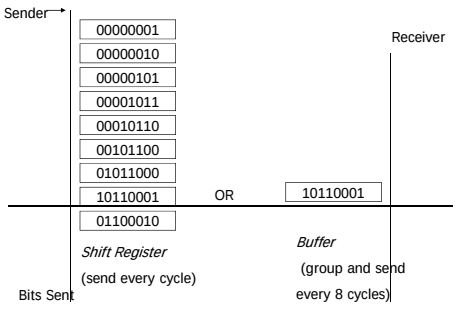


Different Signal Alphabet

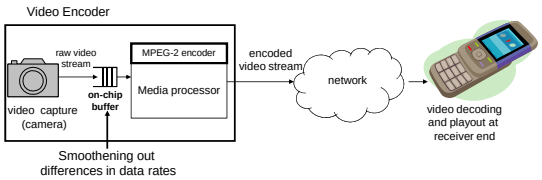




3. Mismatch in data format



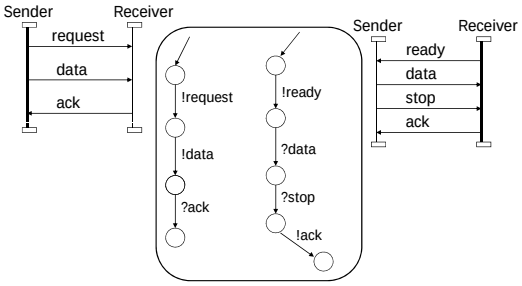
4. Mismatch in data rates



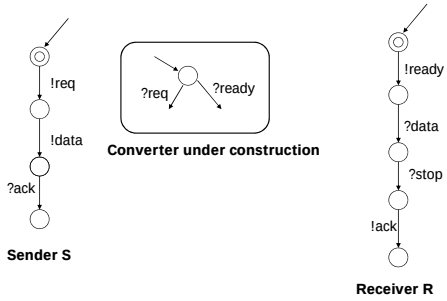
Organization

- Task of converter/intrafaces
  - Enable communication.
  - Resolve protocol incompatibilties.
- What are the common incompatibilties?
- Converter synthesis
  - Represent native protocols and converter as FSM.

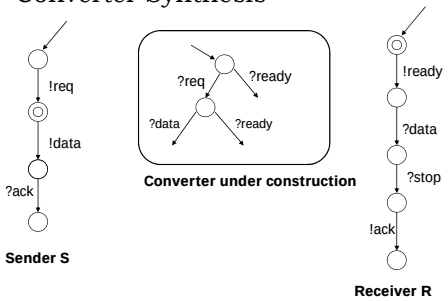
Example: Incompatible protocols

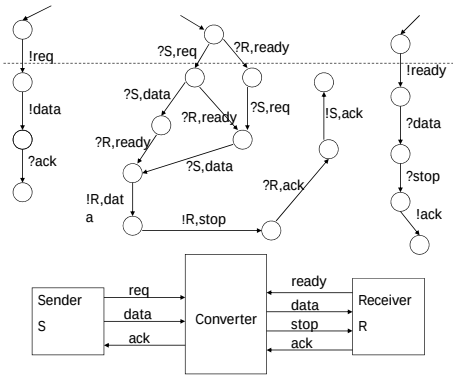


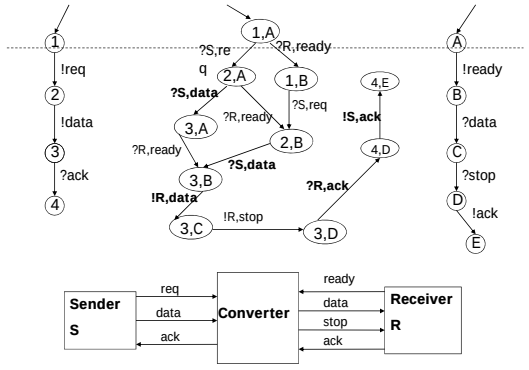
Converter Synthesis



Converter Synthesis

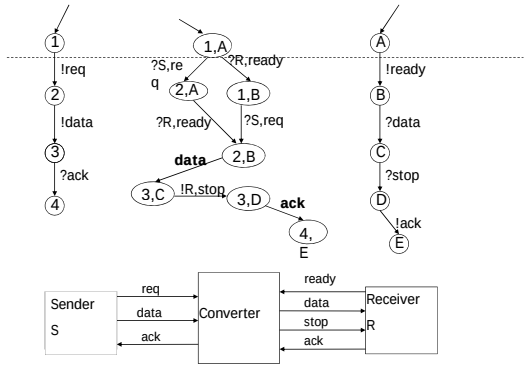






Converter

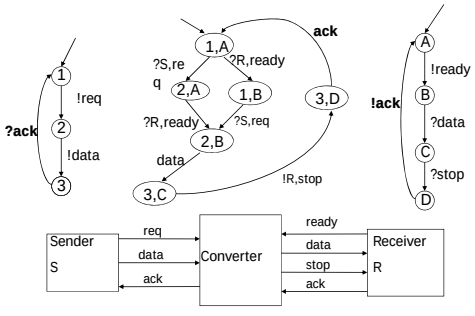
- Product of the incompatible protocol FSMs.
  - Sending and receiving of each signal de-linked.
  - Sending and receiving are separate events.
- What about shared signals?
  - Signals in the common alphabet.
  - Typically include "data" signals.
  - If the sending and receiving happen together, we can avoid storing these signals in converter.
    - This restricts behaviors of converter - fewer exec. traces.



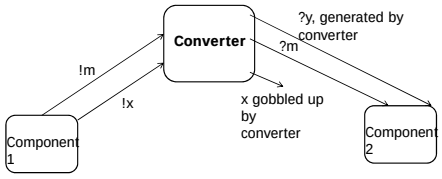
Converter

- Protocol FSMs may contain infinite length traces.
  - Protocol FSM = Interface of a component.
  - Ongoing interaction with other components.
  - FSM with cycles.
- In that case,
  - Converter also needs to be a cyclic FSM.
  - "Sessions" of the protocol interactions across components need to be synchronized.
    - Explicitly (adding new messages), or
    - Implicitly (same msg. at session end in each component).

Synchronizing the "sessions"



Summary: Converter synthesis



Message *m* between *component1* and *component2* flows through converter.
No direct interaction between components.