ReChannel: A Library for Describing and Simulating Reconfigurable Hardware in SystemC

Andreas Raabe  Armin Felke

Technical Computer Science
Outline

- Related Approaches
- What do we want?
- ReChannel
Reconfiguration Bus

Properties

- Interface fixed.
- either:
  - IP Reuse → Tailored Bus
  - Bus fixed → No IP Reuse
- Topology different to static design.
Reconfiguration Bus

Properties
- Interface fixed.
- either:
  - IP Reuse → Tailored Bus
  - Bus fixed → No IP Reuse
- Topology different to static design.
Reconfiguration Bus

Properties
- Interface fixed.
- either:
  - IP Reuse → Tailored Bus
  - Bus fixed → No IP Reuse
- Topology different to static design.
Related Approaches

Polymorphism

Basic change of development paradigm.

Component Based → Object Oriented

- Intuitive for system designers.
- Not so intuitive for hardware developers.
- OO-Extension necessary.

More important:

- Reuse of large SystemC code-base not possible for reconfiguration.
- Utilization of standard SystemC tools not possible.
Polymorphism

Basic change of development paradigm.
Component Based $\rightarrow$ Object Oriented

- Intuitive for system designers.
- Not so intuitive for hardware developers.
- OO-Extension necessary.
- More important:
  - Reuse of large \texttt{SYSTEMC} code-base not possible for reconfiguration.
  - Utilization of standard \texttt{SYSTEMC} tools not possible.
What do we want?

Features

- Description still looks like SYSTEMC.
- IP Reuse
- Standard Compliance (no meddling with kernel)
- Use SYSTEMC tools.
What do we want?

**Features**

- Description still looks like `SYSTEMC`.
- IP Reuse
- Standard Compliance (no meddling with kernel)
- Use `SYSTEMC` tools.
What do we want?

Features

- Description still looks like \texttt{SYSTEMC}.
- IP Reuse
- Standard Compliance (no meddling with kernel)
- Use \texttt{SYSTEMC} tools.
The ReCHANNEL Approach - Portals

Properties

- Topology remains unchanged.
- Reconfigurable modules’ implementations remain unchanged → IP Reuse.
- Language construct.
- Exportals are also provided.

Supports all channel types → Description on all abstraction levels.

Usage Example

```cpp
rc_portal< sc_fifo_out<int> > my_portal;

my_portal.bind_static( my_fifo_channel );
my_portal.bind_dynamic( A.out );
my_portal.bind_dynamic( B.some_other_out );
```
The ReCHANNEL Approach - Portals

Properties
- Topology remains unchanged.
- Reconfigurable modules’ implementations remain unchanged → IP Reuse.
- Language construct.
- Exportals are also provided.

Supports all channel types → Description on all abstraction levels.

Usage Example

```c
rc_portal< sc_fifo_out<int> > my_portal;

my_portal.bind_static( my_fifo_channel );
my_portal.bind_dynamic( A.out );
my_portal.bind_dynamic( B.some_other_out );
```
The ReCHannel Approach - Portals

Properties

- Topology remains unchanged.
- Reconfigurable modules’ implementations remain unchanged → IP Reuse.
- Language construct.
- Exportals are also provided.

Supports all channel types → Description on all abstraction levels.

Usage Example

```c
rc_portal< sc_fifo_out<int> > my_portal;

my_portal.bind_static( my_fifo_channel );
my_portal.bind_dynamic( A.out );
my_portal.bind_dynamic( B.some_other_out );
```
The ReChannel Approach - Reconfigurable Modules

**Properties**

- Enable central controlling of (ex-)portals.
- (Static) IP components can be extended with reconfiguration properties (i.e., delay, behavior).

**Usage Example**

```cpp
RC_RECONFIGURABLE_MODULE_DERIVED(A_rc, A) {
    RC_RECONFIGURABLE_CTOR_DERIVED(A_rc, A) {
        rc_set_delay(RC_LOAD, sc_time(1, SC_MS));
    }
}
```
The **ReChannel** Approach - Reconfigurable Modules

**Properties**
- Enable central controlling of (ex-)portals.
- (Static) IP components can be extended with reconfiguration properties (i.e., delay, behavior).

**Usage Example**

```c
RC_RECONFIGURABLE_MODULE_DERIVED(A_rc, A) {
    RC_RECONFIGURABLECTOR_DERIVED(A_rc, A) {
        rc_set_delay(RC_LOAD, sc_time(1, SC_MS));
    }
}
```
Reconfiguration Properties

Properties

- Configuration Control is modelled as a normal module.
- rc_control provides language constructs for registering and controlling reconfigurable modules.
The ReCHannel Approach - Controlling

Usage

```cpp
rc_control ctrl;
ctrl.add(A + B);
ctrl.load(A);
ctrl.activate(A);
ctrl.unload(A);
ctrl.load(B);
ctrl.activate(B);
```
The ReChannel Approach - Controlling

Usage

```c
rc_control ctrl;
ctrl.add(A + B);
ctrl.load(A);
ctrl.activate(A);
ctrl.unload(A);
ctrl.load(B);
ctrl.activate(B);
```
The ReCHANNEL Approach - Controlling

Usage

```c
rc_control ctrl;
ctrl.add(A + B);
ctrl.load(A);
ctrl.activate(A);
ctrl.unload(A);
ctrl.load(B);
ctrl.activate(B);
```
The ReCHANNEL Approach - Controlling

Usage

```c
rc_control ctrl;
ctrl.add(A + B);
ctrl.load(A);
ctrl.activate(A);
ctrl.unload(A);
ctrl.load(B);
ctrl.activate(B);
```
The ReChannel Approach - Controlling

**Usage**

```c
rc_control ctrl;
ctrl.add(A + B);
ctrl.load(A);
ctrl.activate(A);
ctrl.unload(A);
ctrl.load(B);
ctrl.activate(B);
```
Summary

Main Properties

ReChannel ...

... combines IP Reuse with reconfiguration.
... enables description and simulation of reconfiguration on all levels of abstraction featured by SystemC.
... preserves the topology of the design.
... extends SystemC in a natural way.
... complies to SystemC language standard → should work with any kernel implementation.

Additional Properties

... provides its own process control.
... provides features for variable and process reset, more accurate timing simulation, mobility, synchronisation, etc.
... was tested in a realistic case study (Demo Wed. 14:00).
Summary

Main Properties

ReChannel ...

... combines IP Reuse with reconfiguration.
... enables description and simulation of reconfiguration on all levels of abstraction featured by SystemC.
... preserves the topology of the design.
... extends SystemC in a natural way.
... complies to SystemC language standard → should work with any kernel implementation.

Additional Properties

... provides its own process control.
... provides features for variable and process reset, more accurate timing simulation, mobility, synchronisation, etc.
... was tested in a realistic case study (Demo Wed. 14:00).
Thank you for your attention.

**ReChannel**: A Library for Describing and Simulating Reconfigurable Hardware in **SystemC**

Andreas Raabe    Armin Felke
Technical Computer Science

[universität bonn]
Developing a standard compliant extension library would have been easier, if the following was part of the standard:

- Every Port should provide a typedef `IF_type`.
- Port and Export should have a common base type.
- `sc_port_base` should have a method, that returns its interface.
- It should be possible to ask the kernel which is the current (standardized) kernel phases.
Developing a standard compliant extension library would have been easier, if the following was part of the standard:

- Every Port should provide a typedef IF_type.
- Port and Export should have a common base type.
- `sc_port_base` should have a method, that returns its interface.
- It should be possible to ask the kernel which is the current (standardized) kernel phases.