

SystemC – Processes (02A)

SystemC

Copyright (c) 2012 Young W. Lim.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using OpenOffice and Octave.

Based on the following original work

- [1] Aleksandar Milenkovic, 2002
CPE 626 The SystemC Language – VHDL, Verilog Designer's Guide
<http://www.ece.uah.edu/~milenska/ce626-02S/lectures/cpe626-SystemC-L2.ppt>
- [2] Alexander de Graaf, EEMCS/ME/CAS, 2010
SystemC: an overview ET 4351
ens.ewi.tudelft.nl/Education/courses/et4351/SystemC-2010v1.pdf
- [3] Joachim Gerlach, 2001
System-on-Chip Design with System of Computer Engineering
<http://www2.cs.uni-paderborn.de/cs/ag-hardt/Forschung/Data/SystemC-Tutorial.pdf>
- [4] Martino Ruggiero, 2008
SystemC
polimage.polito.it/~lavagno/codes/SystemC_Lezione.pdf
- [5] Deepak Kumar Tal, 1998-2012
SystemC Tutorial
<http://www.asic-world.com/systemc/index.html>

SystemC Processes (1)

- Basic unit of concurrent execution
- Encapsulates functionality
- Have sensitivity lists
- Triggered by events on sensitive signals

- Member functions are registered as processes by a process declaration in **SC_CTOR**
- No input arguments, No output

SystemC Processes (2)

- Expressing concurrency and parallel activities in the system
- Contained in modules
- Access external channel interfaces through the ports
- Not hierarchical → cannot call another process directly
- Can call methods and functions that are not registered as processes

Types of Processes

- Method processes
- Thread processes
- Clocked thread processes (deprecated)

SC_METHOD

- Executed repeatedly
- Run completely and then return
- Cannot be suspended : wait() X
- Should avoid using calls to blocking methods

Registration →

```
SC_METHOD(process_name);  
sensitivity << signal1 << signal2 << .... ;
```

SC_THREAD

- Executed only once and only once by the simulator
- Have complete control on the simulation until return to the simulator
- `exit()`: the process is terminated for the rest of simulation
- `wait()`: suspend process execution until a next trigger
(continue execution until the next `wait()`)

Registration

```
SC_THREAD(process_name);  
sensitivity << signal1 << signal2 << .... ;
```


SC_THREAD v.s SC_METHOD

SC_THREAD

most general process

used to model nearly anything

slower than a SC_METHOD

(→ wait() induces a context switch)

SC_METHOD

faster

Static Sensitivity

- Static sensitivity provides the parameters, which would trigger a process statically
- Specified during design.

```
SC_METHOD(add);  
sensitive << A << B << Cin;
```

Dynamic Sensitivity for SC_METHOD

```
next_trigger(event);
```

```
next_trigger(event1 | eventi, ...);
```

```
next_trigger(event1 & eventi, ...);
```

```
next_trigger(timeout, event);
```

```
next_trigger(timeout, event1 | eventi, ...);
```

```
next_trigger(timeout, event1 & eventi, ...);
```

```
next_trigger(timeout);
```

Dynamic Sensitivity for SC_THREAD

`wait(event);`

`wait(event1 | eventi, ...);`

`wait(event1 & eventi, ...);`

`wait(timeout, event);`

`wait(timeout, event1 | eventi, ...);`

`wait(timeout, event1 & eventi, ...);`

`wait(timeout);`

Process Communications

Communication at the same level

- (a) Processes may communicate with other processes via **channels**
- (a) Processes may be synchronized with other processes via **events**.

Communication with different level

- (a) Processes may communicate with processes outside the local design module through **ports** bound to **channels** by way of **interfaces**.
- (b) Processes may also communicate with processes in sub-module instances via **interfaces** to **channels** connected to the sub-module **ports** or by way of **interfaces** through the module itself of an **sc_export**.

Communication with Processes

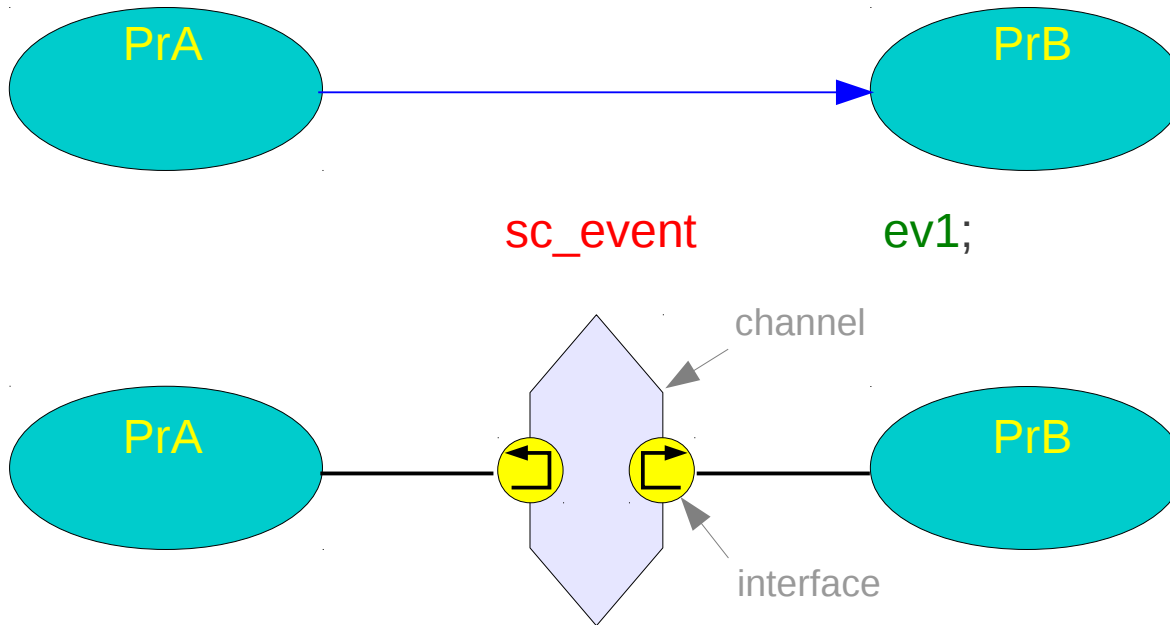
SC_METHOD (PrA) or SC_THREAD(PrA)

SC_METHOD (PrB) or SC_THREAD(PrB)

Communication at the same level

(a) via *channels*

(a) via *events*.



```
trigger(ev1),
sensitive << ev1,
wait(ev1),
next_trigger(ev1),
```

```
sc_signal      sig1;
sc_fifo        fifo1;
sc_mutex       mu1;
sc_semaphore   sema1;
```

```
sig1.read(), sig1.write(),
fifo1.read(), fifo1.write(), ...
mu1.lock(), mu1.unlock(), ...
sema1.wait(), sema1.post(), ...
```

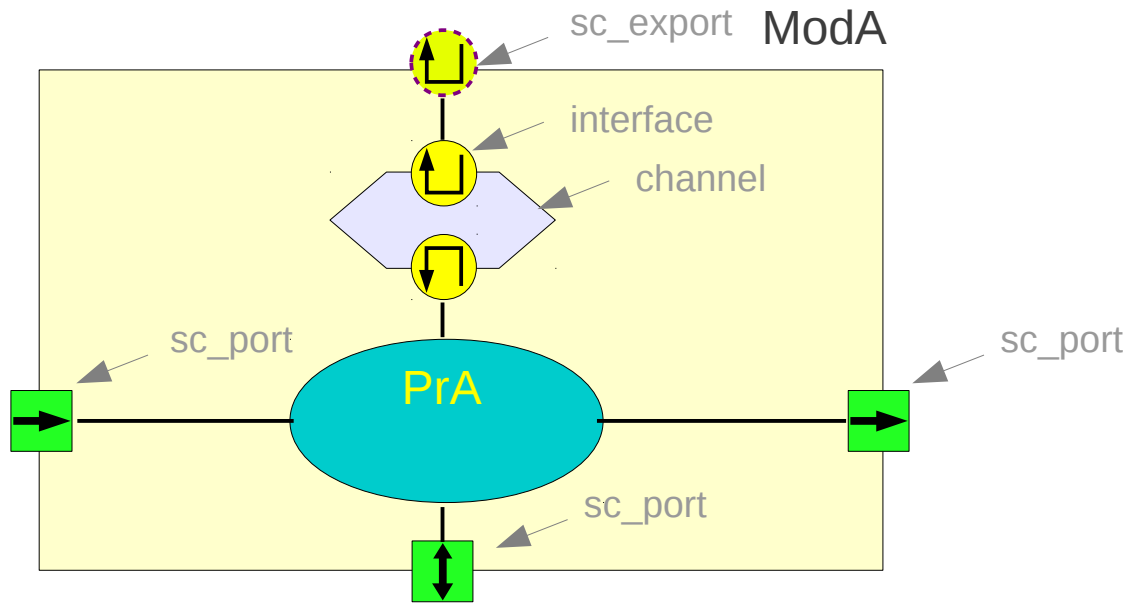
Communication with Outside Modules

Communication with different level

(a) Through **ports** bound to **channels** by way of **interfaces**.

(b) via **interfaces** to **channels** connected to the sub-module **ports**

(c) by way of **interfaces** through the module itself of an **sc_export**.



sc_signal
sc_fifo
sc_mutex
sc_semaphore

sig1;
fifo1;
mu1;
sema1;

sig1.read(), sig1.write(),
fifo1.read(), fifo1.write(), ...
mu1.lock(), mu1.unlock(), ...
sema1.wait(), sema1.post(), ...

Communication with Sub-Modules

SC_METHOD (PrA) or SC_THREAD(PrA)

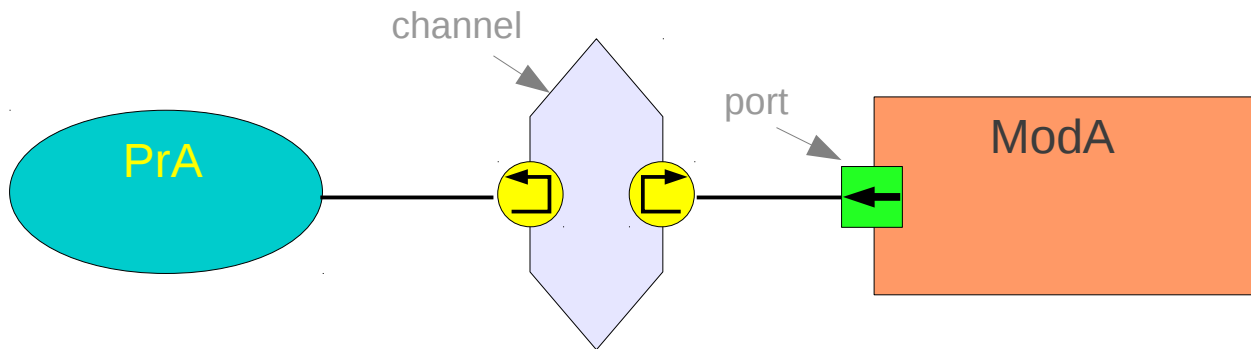
SC_MODULE (ModA)

Communication with different level

(a) Through **ports** bound to **channels** by way of **interfaces**.

(b) via **interfaces** to **channels** connected to the sub-module **ports**

(c) by way of **interfaces** through the module itself of an **sc_export**.

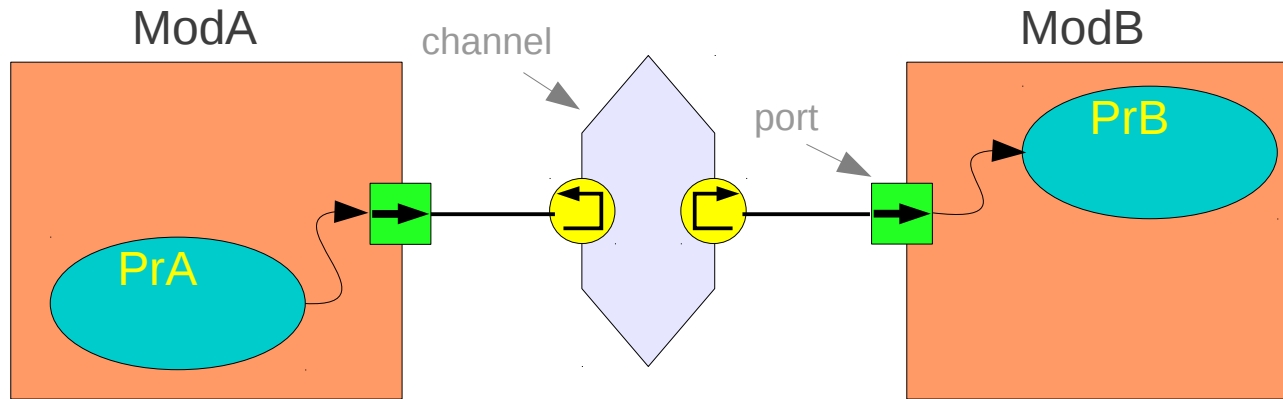


sc_signal
sc_fifo
sc_mutex
sc_semaphore

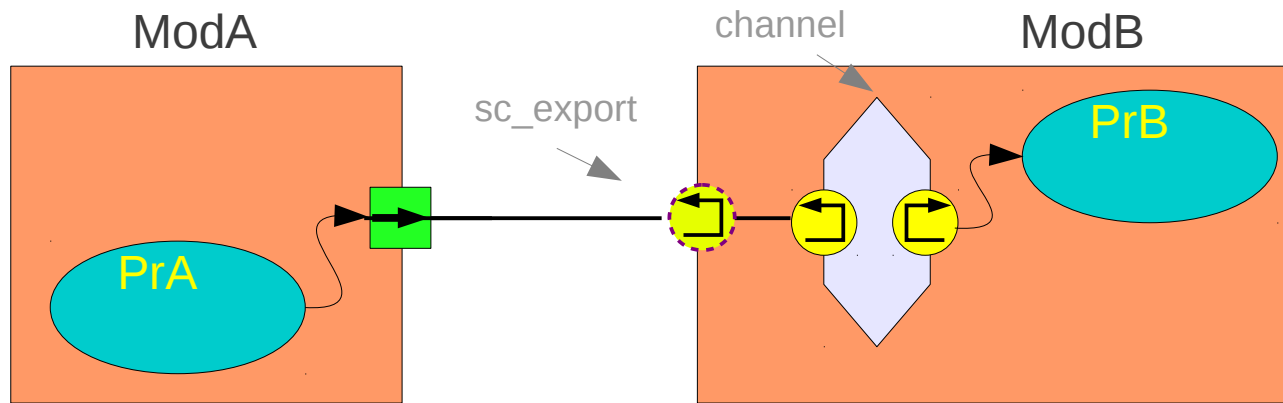
sig1;
fifo1;
mu1;
sema1;

sig1.read(), sig1.write(),
fifo1.read(), fifo1.write(), ...
mu1.lock(), mu1.unlock(), ...
sema1.wait(), sema1.post(), ...

Communication via sc_ports



Communication via sc_exports



References

- [1] Aleksandar Milenkovic, 2002
CPE 626 The SystemC Language – VHDL, Verilog Designer’s Guide
<http://www.ece.uah.edu/~milenska/ce626-02S/lectures/cpe626-SystemC-L2.ppt>

- [2] Alexander de Graaf, EEMCS/ME/CAS, 2010
SystemC: an overview ET 4351
ens.ewi.tudelft.nl/Education/courses/et4351/SystemC-2010v1.pdf

- [3] Joachim Gerlach, 2001
System-on-Chip Design with System of Computer Engineering
<http://www2.cs.uni-paderborn.de/cs/ag-hardt/Forschung/Data/SystemC-Tutorial.pdf>

- [4] Martino Ruggiero, 2008
SystemC
polimage.polito.it/~lavagno/codes/SystemC_Lezione.pdf

- [5] Deepak Kumar Tal, 1998-2012
SystemC Tutorial
<http://www.asic-world.com/systemc/index.html>