

SystemC - Events (08A)

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>

Event

Events are occurrence of signal values and changes.

- Events are meant to trigger processes of modules.
- An event has no duration or value.

It can be used for

- static sensitivity of processes, or
- dynamic sensitivity of processes

Event Trigger

Triggering events: event.notify()

- Events occur explicitly by calling `.notify()` method
- When an event notification is scheduled,
the previous outstanding scheduled event is canceled

Canceling events: event.cancel()

Events can be explicitly canceled by calling `.cancel()` method

sc_event Queue

sc_event queue

- sc_event_queue lets a single event be scheduled repeatedly even for the same time
- when events are scheduled for the same time, each happens in a different delta cycle
- sc_event_queue objects do not support immediate notification
- .cancel() is replaced with .cancel_all()

Sensitivity

The **sensitivity** of a process instance is the set of **events** and **time-outs** that can potentially cause the process to be resumed or triggered.

The **static sensitivity** of an unspawned process instance is **fixed** *during elaboration*.

The **static sensitivity** of a spawned process instance is **fixed** when the function *sc_spawn* is called.

The dynamic sensitivity of a process instance may **vary** over time under the control of the process itself.

A process instance is said to be **sensitive to an event** if the **event** has been added to the static sensitivity or dynamic sensitivity of the process instance.

A **time-out** occurs when a given time interval has elapsed.

Dynamic Sensitivity - sc_method

A method process instance
may have static sensitivity.

A method process, and only a method process,
May call the function **next_trigger** to create **dynamic sensitivity**.

next_trigger

does not suspend the process

temporarily sets a sensitivity list only for next time the process executes again

may be called repeatedly, overriding the previous calls

Without a **next_trigger** or a **static sensitivity**,
such process will never be executed again

Static Sensitivity - sc_method

Data member **sensitive** of class `sc_module` can be used to create the static sensitivity of an unspawned process instance using operator `<<` of class `sc_sensitive`. (the only way)

However, static sensitivity may be enabled or disabled
By calling function **next_trigger()** (\rightarrow `sc_method`) or
By calling function **wait()** (\rightarrow `sc_thread`).

Dynamic Sensitivity - sc_thread

Data member **sensitive** of class `sc_module` can be used to create the static sensitivity of an unspawned process instance using operator `<<` of class `sc_sensitive`. (the only way)

However, static sensitivity may be enabled or disabled
By calling function `next_trigger()` (\rightarrow `sc_method`) or
By calling function `wait()` (\rightarrow `sc_thread`).

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>