Modelado y simulación de sistemas usando SystemC

Cesar Armando Fuguet Tortolero

Universidad de Los Andes (ULA)

Outline I

Outline II

Qué es SystemC?

- SystemC es una librería del lenguaje C++.

Qué es SystemC?

- SystemC es una librería del lenguaje C++.
- Está pensado para hacer frente a Hardware y Software, y permitir modelar y/o simular grandes sistemas.

Qué es SystemC?

- SystemC es una librería del lenguaje C++.
- Está pensado para hacer frente a Hardware y Software, y permitir modelar y/o simular grandes sistemas.
- Tiene como objetivo proveer al equipo de diseño con una especificación ejecutable del sistema.

Qué es SystemC?

- SystemC es una librería del lenguaje C++.
- Está pensado para hacer frente a Hardware y Software, y permitir modelar y/o simular grandes sistemas.
- Tiene como objetivo proveer al equipo de diseño con una especificación ejecutable del sistema.
- SystemC es desarrollado y soportado por OSCI (Open SystemC Initiative), la cual es una organización sin fines de lucro, compuesta por una gran cantidad de compañias, universidades e individuos.

Qué es SystemC?

- SystemC es una librería del lenguaje C++.
- Está pensado para hacer frente a Hardware y Software, y permitir modelar y/o simular grandes sistemas.
- Tiene como objetivo proveer al equipo de diseño con una especificación ejecutable del sistema.
- SystemC es desarrollado y soportado por OSCI (Open SystemC Initiative), la cual es una organización sin fines de lucro, compuesta por una gran cantidad de compañias, universidades e individuos.

Librería SystemC.h

SystemC agrega al lenguaje C++ las siguientes características:

Librería SystemC.h

SystemC agrega al lenguaje C++ las siguientes características:

- Noción de tiempo.

Librería SystemC.h

SystemC agrega al lenguaje C++ las siguientes características:

- Noción de tiempo.
- Concurrencia.

Librería SystemC.h

SystemC agrega al lenguaje C++ las siguientes características:

- Noción de tiempo.
- Concurrencia.
- Tipos de datos especiales para modelado de Hardware.

Librería SystemC.h

SystemC agrega al lenguaje C++ las siguientes características:

- Noción de tiempo.
- Concurrencia.
- Tipos de datos especiales para modelado de Hardware.
- Jerarquía modular para manejar la estructura y conectividad.

Librería SystemC.h

SystemC agrega al lenguaje C++ las siguientes características:

- Noción de tiempo.
- Concurrencia.
- Tipos de datos especiales para modelado de Hardware.
- Jerarquía modular para manejar la estructura y conectividad.

Fortalezas de SystemC

Algunas de las ventajas que ofrece el uso de SystemC son:

Fortalezas de SystemC

Algunas de las ventajas que ofrece el uso de SystemC son:

- Abstracción: Ofrece varios niveles de abstracción para el modelado de sistemas.

Fortalezas de SystemC

Algunas de las ventajas que ofrece el uso de SystemC son:

- Abstracción: Ofrece varios niveles de abstracción para el modelado de sistemas.
- Reutilización del diseño: Al igual que C++, el diseño puede ser reutilizado.

Fortalezas de SystemC

Algunas de las ventajas que ofrece el uso de SystemC son:

- Abstracción: Ofrece varios niveles de abstracción para el modelado de sistemas.
- Reutilización del diseño: Al igual que C++, el diseño puede ser reutilizado.
- Permite especificar al mismo tiempo el software y el hardware de un sistema electrónico.

Fortalezas de SystemC

Algunas de las ventajas que ofrece el uso de SystemC son:

- Abstracción: Ofrece varios niveles de abstracción para el modelado de sistemas.
- Reutilización del diseño: Al igual que C++, el diseño puede ser reutilizado.
- Permite especificar al mismo tiempo el software y el hardware de un sistema electrónico.

Comparación con otros LDH

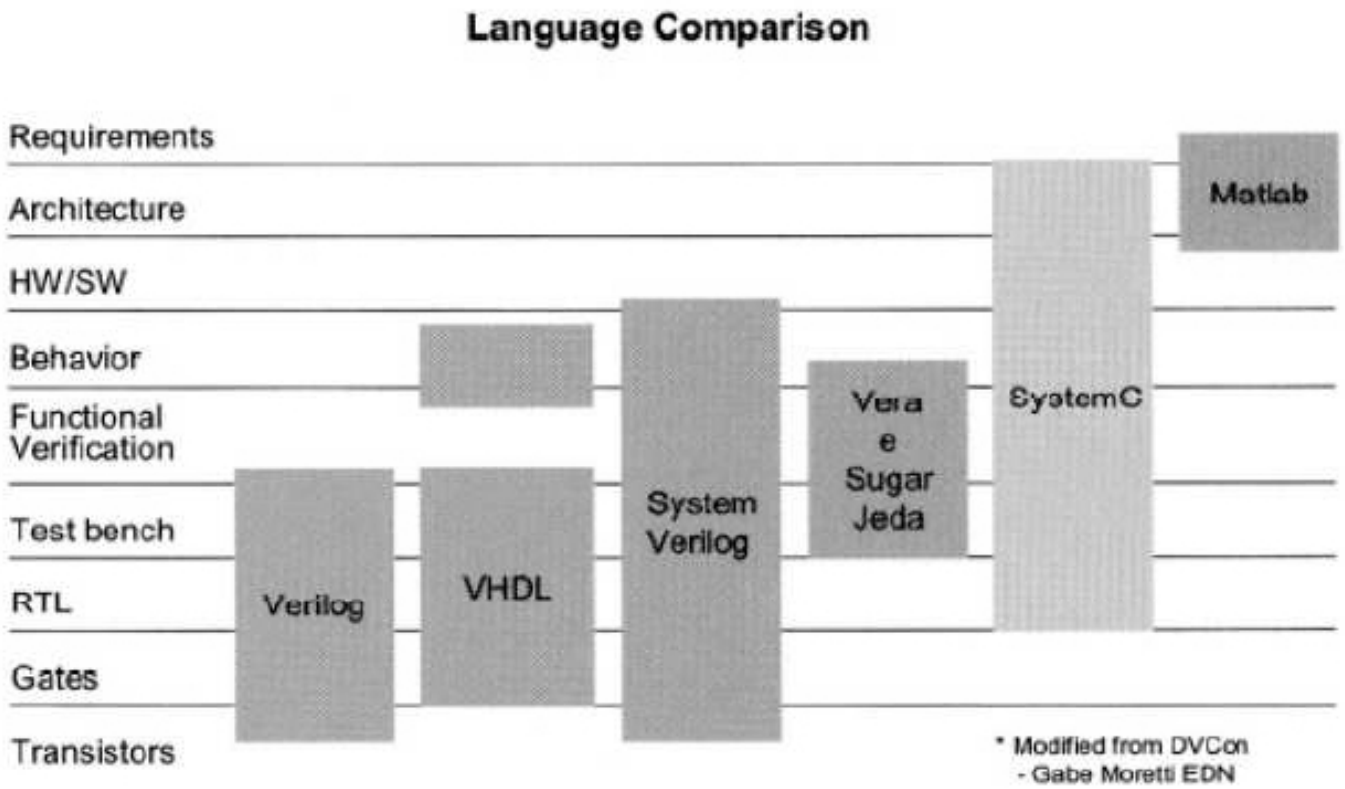

Figure: Niveles de abstracción manejados por SystemC respecto al de otros lenguajes.

Vista General

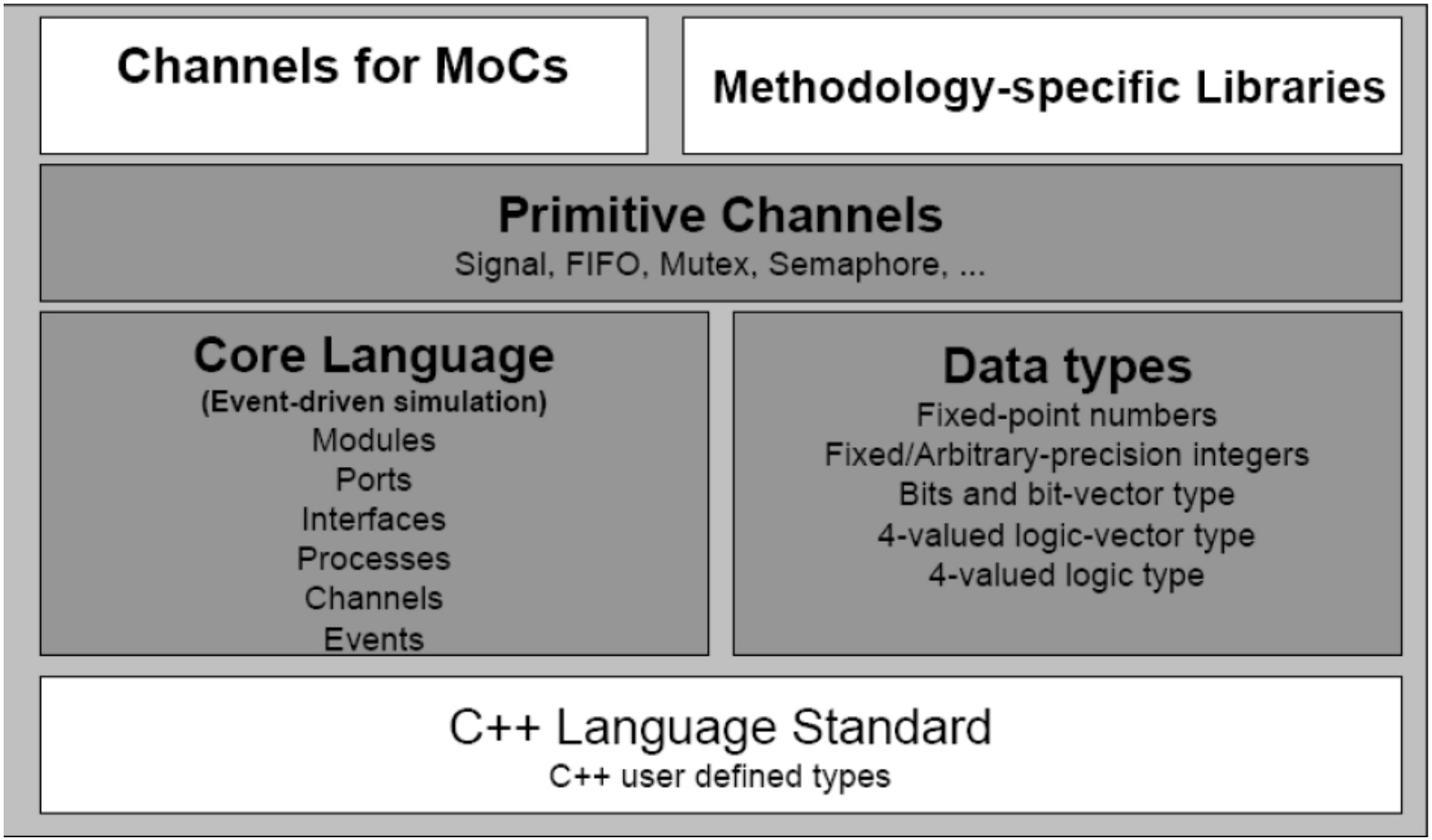

Figure: Elementos de la librería SystemC

Módulos *(Modules)*

Los módulos son la unidad básica de diseño en SystemC, y se caracterizan por lo siguiente:

Módulos *(Modules)*

Los módulos son la unidad básica de diseño en SystemC, y se caracterizan por lo siguiente:

- Pueden representar un sistema, un bloque, un tablero, un chip y así sucesivamente.

Módulos *(Modules)*

Los módulos son la unidad básica de diseño en SystemC, y se caracterizan por lo siguiente:

- Pueden representar un sistema, un bloque, un tablero, un chip y así sucesivamente.
- Permiten dividir un sistema complejo en pequeños, manejables y reutilizables bloques.

Módulos *(Modules)*

Los módulos son la unidad básica de diseño en SystemC, y se caracterizan por lo siguiente:

- Pueden representar un sistema, un bloque, un tablero, un chip y así sucesivamente.
- Permiten dividir un sistema complejo en pequeños, manejables y reutilizables bloques.
- Pueden contener puertos, canales, procesos, variables internas, métodos internos y otros módulos.

Módulos *(Modules)*

Los módulos son la unidad básica de diseño en SystemC, y se caracterizan por lo siguiente:

- Pueden representar un sistema, un bloque, un tablero, un chip y así sucesivamente.
- Permiten dividir un sistema complejo en pequeños, manejables y reutilizables bloques.
- Pueden contener puertos, canales, procesos, variables internas, métodos internos y otros módulos.

Declaración de Módulos

Para declarar módulos en un modelo realizado con SystemC, pueden utilizarse cualquiera de las siguientes sentencias:

- SC_MODULE(*nombremodulo*)
- class *nombremodulo* : public sc_module
- struct *nombremodulo* : public sc_module

Gráfico de un módulo

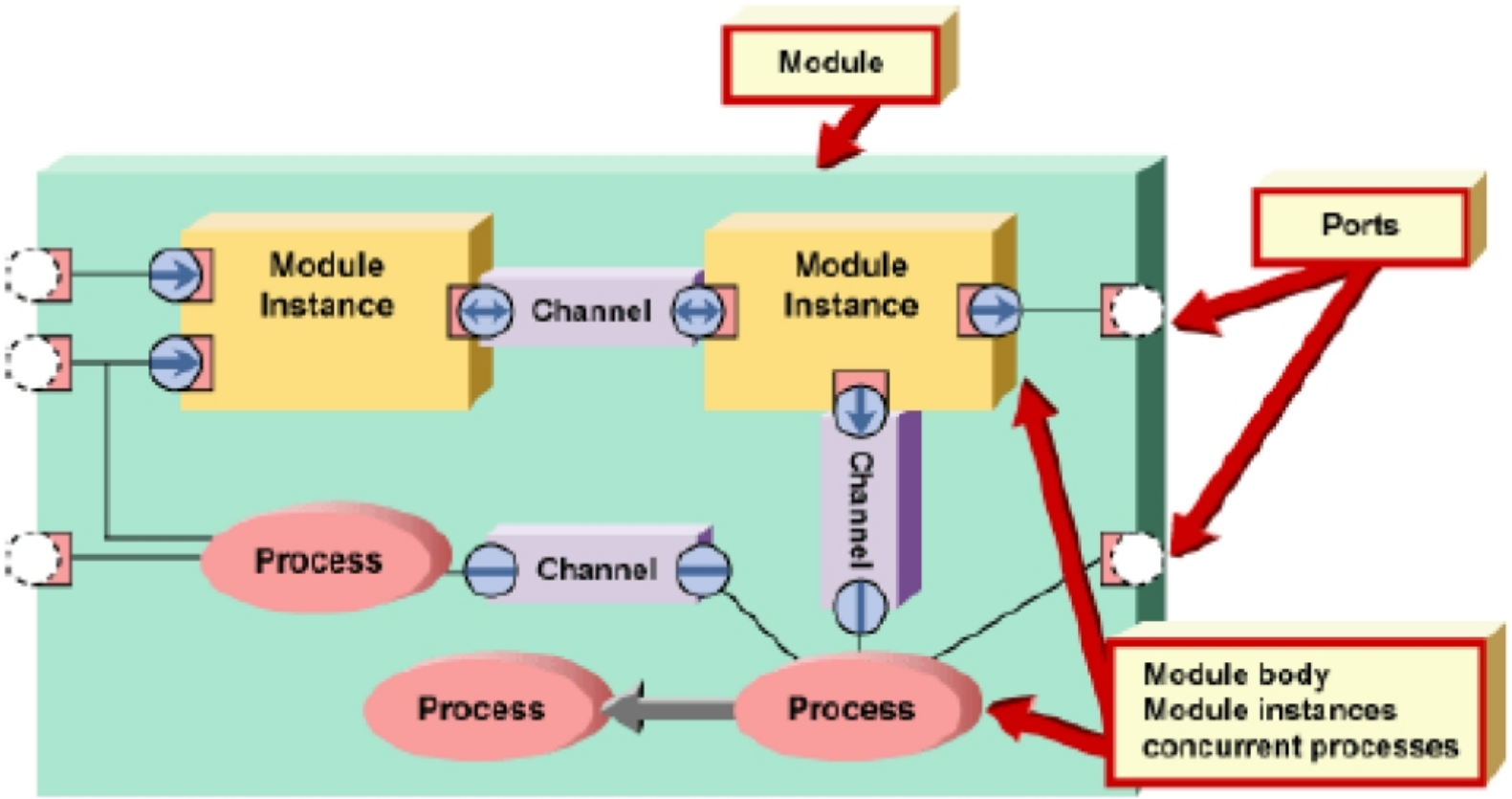

Figure: Elementos que puede contener un módulo

Procesos *(Process)*

Es una función o método de un módulo y posee las siguientes características:

Procesos *(Process)*

Es una función o método de un módulo y posee las siguientes características:

- Es invocado por el planificador del *kernel de simulación de SystemC*.

Procesos *(Process)*

Es una función o método de un módulo y posee las siguientes características:

- Es invocado por el planificador del *kernel de simulación de SystemC*.
- Representa la unidad básica de ejecución de un módulo.

Procesos *(Process)*

Es una función o método de un módulo y posee las siguientes características:

- Es invocado por el planificador del *kernel de simulación de SystemC*.
- Representa la unidad básica de ejecución de un módulo.
- Describen la funcionalidad del módulo.

Procesos *(Process)*

Es una función o método de un módulo y posee las siguientes características:

- Es invocado por el planificador del *kernel de simulación de SystemC*.
- Representa la unidad básica de ejecución de un módulo.
- Describen la funcionalidad del módulo.
- No toman argumentos ni retornan nada.

Procesos *(Process)*

Es una función o método de un módulo y posee las siguientes características:

- Es invocado por el planificador del *kernel de simulación de SystemC*.
- Representa la unidad básica de ejecución de un módulo.
- Describen la funcionalidad del módulo.
- No toman argumentos ni retornan nada.
- Pueden hacerse *sensibles* a ciertos *puertos o señales*, de forma que cuando los valores de estos cambien, el procesos sean invocado.

Procesos *(Process)*

Es una función o método de un módulo y posee las siguientes características:

- Es invocado por el planificador del *kernel de simulación de SystemC*.
- Representa la unidad básica de ejecución de un módulo.
- Describen la funcionalidad del módulo.
- No toman argumentos ni retornan nada.
- Pueden hacerse *sensibles* a ciertos *puertos o señales*, de forma que cuando los valores de estos cambien, el procesos sean invocado.

Procesos *(Process)*

- Poseen una lista de sensibilidad, en la cual se especifican los eventos que activarán la ejecución del proceso.

Procesos *(Process)*

- Poseen una lista de sensibilidad, en la cual se especifican los eventos que activarán la ejecución del proceso.
- Se comunican entre sí, a través de canales o eventos.

Procesos *(Process)*

- Poseen una lista de sensibilidad, en la cual se especifican los eventos que activarán la ejecución del proceso.
- Se comunican entre sí, a través de canales o eventos.

Declaración de Procesos

Son declarados usando una de las siguientes palabras reservadas, las cuales ofrecen diferentes funcionalidades:

- SC_METHOD(*nombreproceso*)
- SC_THREAD(*nombreproceso*)
- SC_CTHREAD(*nombreproceso*)

Puertos *(Ports)*

Los puertos son elementos esenciales en un modelo realizado con SystemC. Sus características y funciones son:

Puertos *(Ports)*

Los puertos son elementos esenciales en un modelo realizado con SystemC. Sus características y funciones son:

- Son usados por los módulos como puertas al mundo exterior.

Puertos *(Ports)*

Los puertos son elementos esenciales en un modelo realizado con SystemC. Sus características y funciones son:

- Son usados por los módulos como puertas al mundo exterior.
- Pueden representar: interfaz, pins, etc.

Puertos *(Ports)*

Los puertos son elementos esenciales en un modelo realizado con SystemC. Sus características y funciones son:

- Son usados por los módulos como puertas al mundo exterior.
- Pueden representar: interfaz, pins, etc.
- Se usan para interconectar los módulos a través de canales.

Puertos *(Ports)*

Los puertos son elementos esenciales en un modelo realizado con SystemC. Sus características y funciones son:

- Son usados por los módulos como puertas al mundo exterior.
- Pueden representar: interfaz, pins, etc.
- Se usan para interconectar los módulos a través de canales.
- Poseen una dirección (entrada, salida, entrada/salida).

Puertos *(Ports)*

Los puertos son elementos esenciales en un modelo realizado con SystemC. Sus características y funciones son:

- Son usados por los módulos como puertas al mundo exterior.
- Pueden representar: interfaz, pins, etc.
- Se usan para interconectar los módulos a través de canales.
- Poseen una dirección (entrada, salida, entrada/salida).

Declaración de Puertos

Son declarados usando una de las siguientes palabras reservadas:

- sc_in<*tipodedato*> *nombrepuerto*;
- sc_out<*tipodedato*> *nombrepuerto*;
- sc_inout<*tipodedato*> *nombrepuerto*;

Canales o señales *(channels,signals)*

Son mecanismos construidos en la librería SystemC, poseen las siguientes caracteríscas y funcionalidades:

Canales o señales *(channels,signals)*

Son mecanismos construidos en la librería SystemC, poseen las siguientes caracteríscas y funcionalidades:

- Son usados para manejar la comunicación entre módulos y procesos.

Canales o señales *(channels,signals)*

Son mecanismos construidos en la librería SystemC, poseen las siguientes caracteríscas y funcionalidades:

- Son usados para manejar la comunicación entre módulos y procesos.
- Encapsulan la complejidad de la comunicación.

Canales o señales *(channels,signals)*

Son mecanismos construidos en la librería SystemC, poseen las siguientes caracteríscas y funcionalidades:

- Son usados para manejar la comunicación entre módulos y procesos.
- Encapsulan la complejidad de la comunicación.
- Hay dos tipos de canales o señales:
  - Primitivos
  - Jerárquicos

Canales o señales *(channels,signals)*

Son mecanismos construidos en la librería SystemC, poseen las siguientes caracteríscas y funcionalidades:

- Son usados para manejar la comunicación entre módulos y procesos.
- Encapsulan la complejidad de la comunicación.
- Hay dos tipos de canales o señales:
  - Primitivos
  - Jerárquicos

Declaración de canales o señales

Son declarados usando una de las siguientes palabras reservadas:

- sc_signal<*tipodato*> *nombreseñal*;
- sc_buffer<*tipodato*> *nombreseñal*;
- sc_fifo<*tipodato*> *nombreseñal*;
- Otros.

Gráfico de una señal o canal

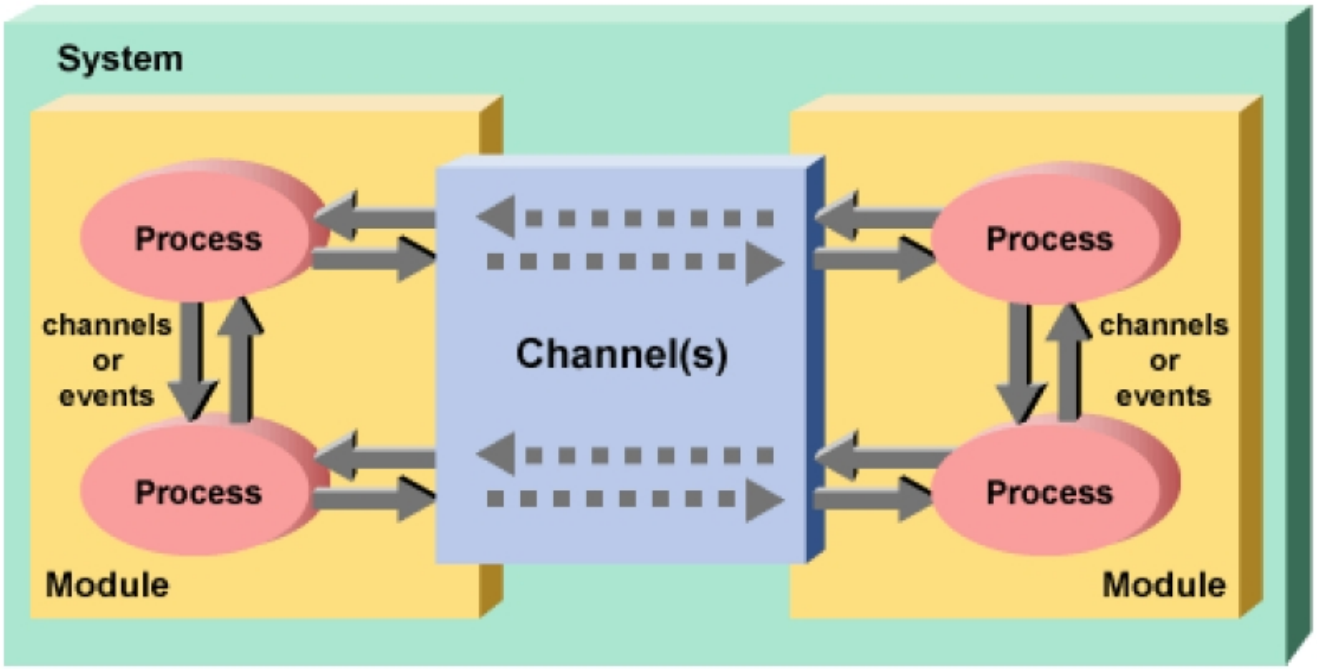

Figure: Canal o señal

Eventos *(Events)*

Eventos *(Events)*

- Los eventos en SystemC es la ocurrencia de una notificación de un sc_event.

Eventos *(Events)*

- Los eventos en SystemC es la ocurrencia de una notificación de un sc_event.
- Ocurre en un solo momento del tiempo.

Eventos *(Events)*

- Los eventos en SystemC es la ocurrencia de una notificación de un sc_event.
- Ocurre en un solo momento del tiempo.
- No tiene ni valor ni duración.

Eventos *(Events)*

- Los eventos en SystemC es la ocurrencia de una notificación de un sc_event.
- Ocurre en un solo momento del tiempo.
- No tiene ni valor ni duración.
- Estas ocurrencias pueden ser cambios en los valores de algún puerto o canal.

Eventos *(Events)*

- Los eventos en SystemC es la ocurrencia de una notificación de un sc_event.
- Ocurre en un solo momento del tiempo.
- No tiene ni valor ni duración.
- Estas ocurrencias pueden ser cambios en los valores de algún puerto o canal.
- Son destinados para activar los procesos en los módulos.

Eventos *(Events)*

- Los eventos en SystemC es la ocurrencia de una notificación de un sc_event.
- Ocurre en un solo momento del tiempo.
- No tiene ni valor ni duración.
- Estas ocurrencias pueden ser cambios en los valores de algún puerto o canal.
- Son destinados para activar los procesos en los módulos.

Captura de los eventos

Regla

- Para observar un evento, el observador debe estar esperando por el evento.

Interfaces *(interfaces)*

Representan uno de los elementos abstractos de la librería. No son usados directamente por el usuario durante el modelado o simulación de un sistema. Sus funciones son:

Interfaces *(interfaces)*

Representan uno de los elementos abstractos de la librería. No son usados directamente por el usuario durante el modelado o simulación de un sistema. Sus funciones son:

- Proveen la traducción y los mecanismos de comunicación necesarios entre los puertos y los canales o señales.

Interfaces *(interfaces)*

Representan uno de los elementos abstractos de la librería. No son usados directamente por el usuario durante el modelado o simulación de un sistema. Sus funciones son:

- Proveen la traducción y los mecanismos de comunicación necesarios entre los puertos y los canales o señales.
- Es una clase abstracta que solo provee declaraciones virtuales de métodos referenciados por los canales y puertos de la librería.

Interfaces *(interfaces)*

Representan uno de los elementos abstractos de la librería. No son usados directamente por el usuario durante el modelado o simulación de un sistema. Sus funciones son:

- Proveen la traducción y los mecanismos de comunicación necesarios entre los puertos y los canales o señales.
- Es una clase abstracta que solo provee declaraciones virtuales de métodos referenciados por los canales y puertos de la librería.

Gráfico de una interfaz

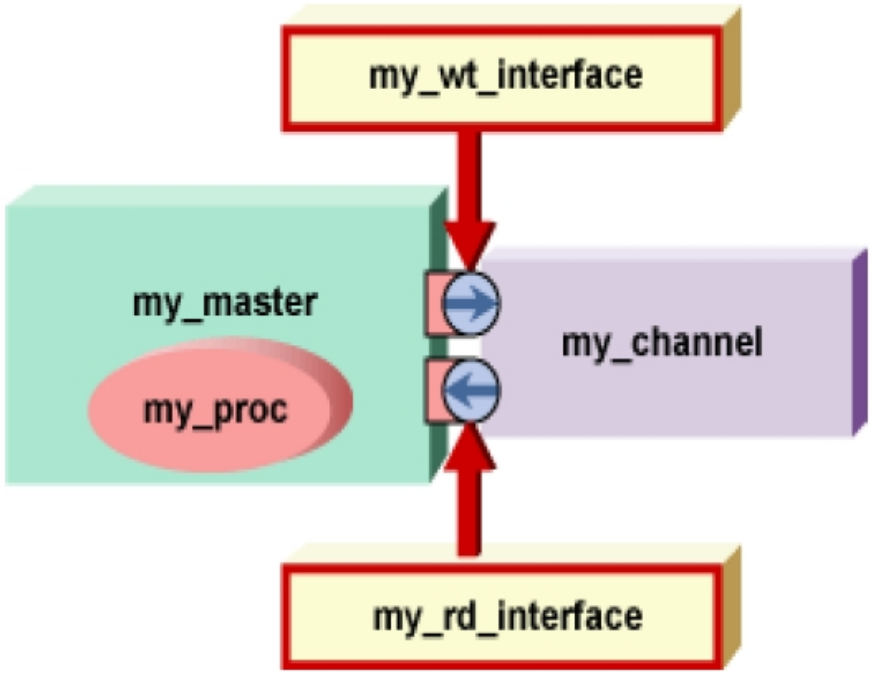

Figure: Interfaces

Estructura clásica de un módulo

Como ya se explico anteriormente, los módulos son la estructura básica y fundamental en un modelo realizado con la librería SystemC. A continuación se van a mostrar una serie de sugerencias para la codificación de un módulo.

Sugerencias

- Los módulos se definen en archivos con extensión *.h* y la implementación de los procesos en archivos con extensión *.cpp*.
- Luego de que todos los módulos estén hechos, se debe realizar un algoritmo para verificar el modelo. *(Testbench)*
- Por último todos los módulos son interconectados entre ellos y el módulo con los casos de prueba. Esta interconexión se realiza en el módulo de mas alto nivel, el *sc_main*.

```
// nombremodulo.h
// Libreria SystemC.
// Debe incluirse en todas las definiciones de modulos
#include<systemc.h>
// Puede añadirse cualquier otra librería.
// Ejemplo: iostream
class nombremodulo : public sc_module {
 public:
  // Declaracion de puertos
  SC_CTOR(nombremodulo){
    // Registro de procesos
    // Lista de sensibilidad estatica
    // Inicializacion de variables locales
    // Conexion de las senales entre los submodulos
  }
 private:
  // Declaracion de señales o canales locales
  // Declaracion de variables
  // Declaracion de procesos
  // Declaracion de submodulos
  // Declaracion de metodos locales
};
```

Kernel de simulación *(simulation kernel)*

Durante una simulación en SystemC, los procesos son ejecutados y los valores en las señales son actualizados en determinadas situaciones. Estas situaciones son las siguientes:

Kernel de simulación *(simulation kernel)*

Durante una simulación en SystemC, los procesos son ejecutados y los valores en las señales son actualizados en determinadas situaciones. Estas situaciones son las siguientes:

- Transiciones de reloj, en el caso de circuitos secuenciales.
- Cambios en las señales de entrada, en el caso de circuitos combinacionales.
- Combinación entre las situaciones anteriores.

Kernel de simulación *(simulation kernel)*

Durante una simulación en SystemC, los procesos son ejecutados y los valores en las señales son actualizados en determinadas situaciones. Estas situaciones son las siguientes:

- Transiciones de reloj, en el caso de circuitos secuenciales.
- Cambios en las señales de entrada, en el caso de circuitos combinacionales.
- Combinación entre las situaciones anteriores.

Por esta razón SystemC incluye un organizador, el cual realiza las siguientes tareas:

Kernel de simulación *(simulation kernel)*

Durante una simulación en SystemC, los procesos son ejecutados y los valores en las señales son actualizados en determinadas situaciones. Estas situaciones son las siguientes:

- Transiciones de reloj, en el caso de circuitos secuenciales.
- Cambios en las señales de entrada, en el caso de circuitos combinacionales.
- Combinación entre las situaciones anteriores.

Por esta razón SystemC incluye un organizador, el cual realiza las siguientes tareas:

- Manejar todos los eventos sobre las señales.
- Organizar los procesos para ser ejecutados cuando eventos apropiados ocurren en sus entradas.

Kernel de simulación *(simulation kernel)*

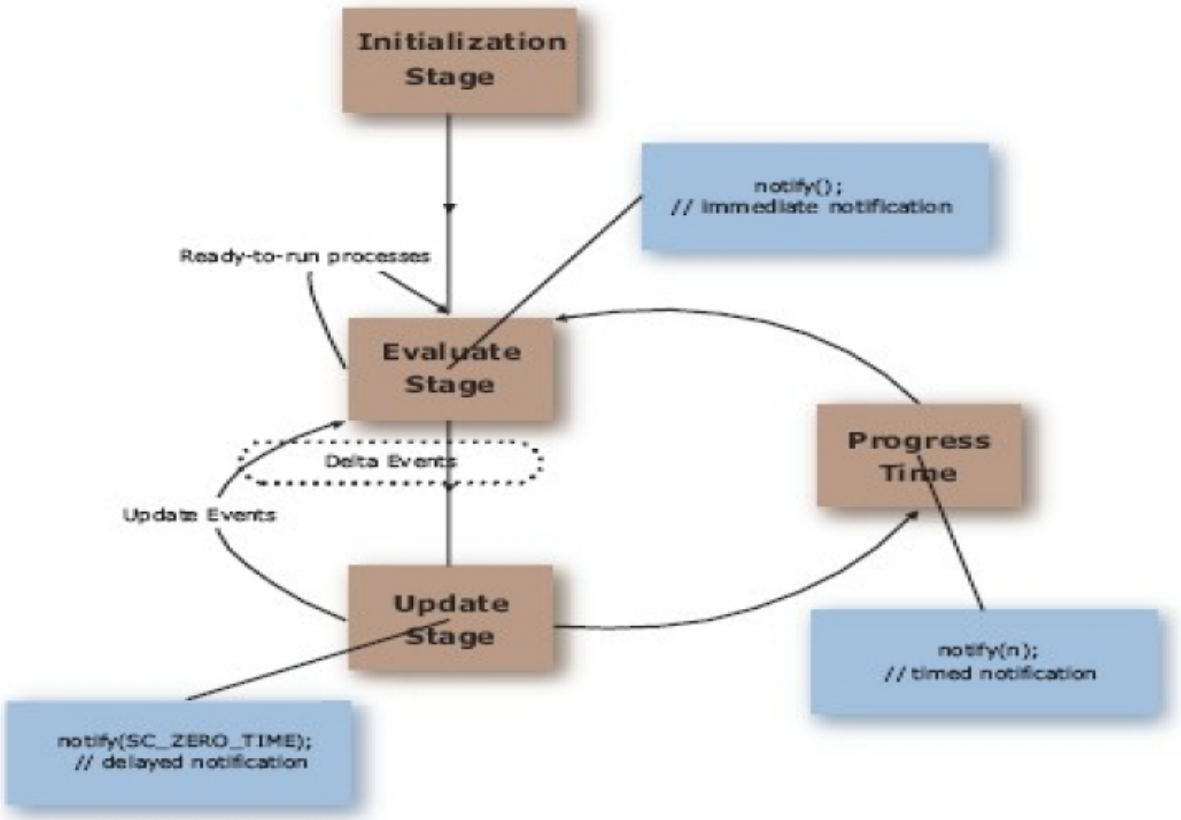

Figure: Kernel de Simulación de SystemC

Pasos realizados por el Kernel de Simulación

1. Inicializar

- Se preparan todos los procesos para ser ejecutados.
- Se ejecuta cada uno de los procesos que no poseen la sentencia "*dont_initialize()*"

Pasos realizados por el Kernel de Simulación

1. Inicializar
2. Evaluación

- Se selecciona un proceso elegible y se ejecuta.
- Si en el proceso se usan notificaciones inmediatas, otro proceso puede volverse elegible también.
- Se continua seleccionando procesos elegibles hasta que no quede ninguno por evaluar.

Pasos realizados por el Kernel de Simulación

1. Inicializar
2. Evaluación
3. Actualización

- Se actualizan las entradas y salidas de las señales.
- Esto podría generar que otros procesos se vuelvan elegibles.

Pasos realizados por el Kernel de Simulación

1. Inicializar
2. Evaluación
3. Actualización
4. Si hay procesos ejecutables, se vuelve a la etapa de evaluación.

Pasos realizados por el Kernel de Simulación

1. Inicializar
2. Evaluación
3. Actualización
4. Si hay procesos ejecutables, se vuelve a la etapa de evaluación.
5. Avance en el tiempo.

- En este pundo no hay procesos ejecutables.
- Cambio de estado del reloj de simulación.

Pasos realizados por el Kernel de Simulación

1. Inicializar
2. Evaluación
3. Actualización
4. Si hay procesos ejecutables, se vuelve a la etapa de evaluación.
5. Avance en el tiempo.
6. Determina procesos ejecutables. Vuelta a la etapa de evaluación.

GRACIAS