Universidad Técnica Federico Santa María
Departamento de Electrónica
Programación de Sistemas (ELO330)

Introducción a SystemC

Integrantes: Waldo López
9804733-6
Javier Santa Ana
9821049-0
Profesor: Agustín González
Fecha: 31/10/2003

RESUMEN

Este informe tiene como idea entregar al usuario la información necesaria para que entienda conceptos básicos sobre SystemC. SystemC está pensado para hacer frente a hardware y software, y permitir modelar y/o simular grandes sistemas.

Aquí se definirán conceptos como: estructura de modelo, planificadores, interfaces, canales, módulos, como definir módulos, los eventos, métodos, jerarquía, modelamiento, etc., conceptos muy útiles para la programación.

Solo se definen algunos conceptos, para mas detalles sobre programación o sintaxis de las funciones véase el anexo, o a las referencias sobre SystemC.

INTRODUCCION

C++ aun cuando cumple un desempeño bastante satisfactorio en el modelado de software, resulta ser inadecuado para el modelado de HW. Por tal razón ha surgido la idea de realizar una librería que permita el mejor modelado de HW. A continuación se detallaran algunas situaciones en las cuales C++ no cumple satisfactoriamente:

Noción de Tiempo:

- C++ no apoya la noción de acontecimientos ordenados tiempo. No hay noción del tiempo.

Concurrencia:

- C++ no apoya la idea de la concurrencia. Los sistemas del hardware son intrínsecamente concurrentes. Funcionan en paralelo

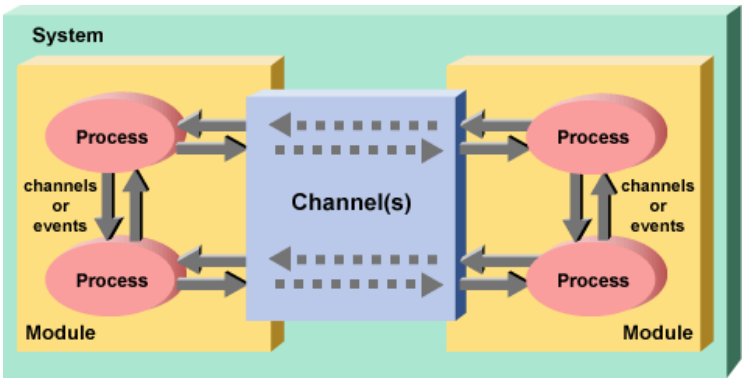
Tipos de datos de Hardware:

- Los tipos de datos nativos de C++ son inadecuados para modelar el hardware. Por ejemplo no hay tipo de datos que podría ser usados para describir un valor de triple estado de la lógica (z) (tri –state logic value)

Todos estos inconvenientes son suplidos gracias a la aparición de SystemC, el cual es concebido para la solución de estos problemas específicos y muchos más. A continuación se presentara una explicación de los conceptos involucrados en esta Librería, con el fin de servir como punto de partida para aprender el funcionamiento de SystemC que puede ser de gran utilidad en algunos modelados

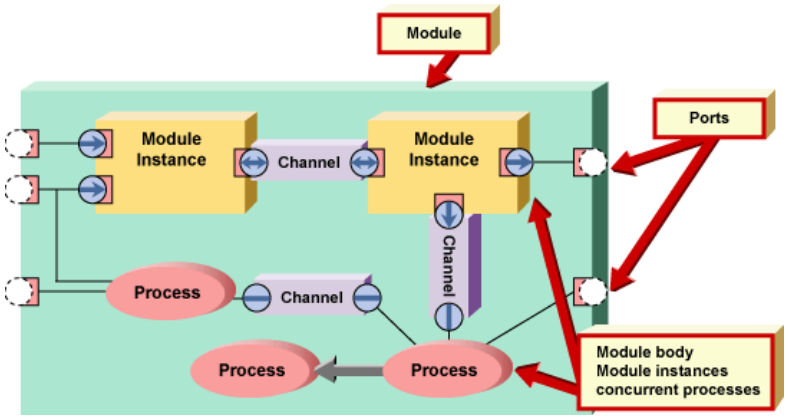
Sistema de SystemC

Un sistema de SystemC consiste en un conjunto de módulos. Los módulos contienen procesos concurrentes. Los procesos describen funcionalidad y se comunican con otros procesos a través de canales o eventos. La comunicación entre módulos es a través de canales. Los módulos se utilizan para crear jerarquía.



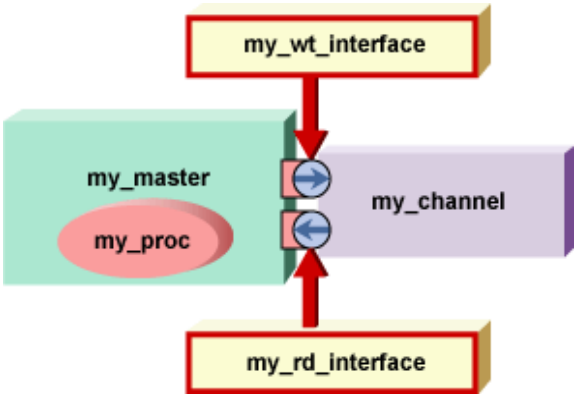
Estructura básica de modelo.

Un módulo puede representar un sistema, un bloque, un tablero, un chip y así sucesivamente. Los puertos representan una interfaz, pins, etc., dependiendo de lo que representa el módulo.



Los casos del módulo en un nivel par están conectados a través de sus puertos con un canal. Conexiones padre hijo de módulos son realizados puerto a puerto. Los procesos se comunican a través de los canales o eventos. La estructura básica de la comunicación incluye interfaces, los canales y los puertos.

Un ejemplo de esta estructura básica es el siguiente:



Como se puede apreciar en la figura superior *my_proc* es un proceso dentro del modulo *my_master*. *my_wt_interface* es una interfaz que define un método (función) llamada *wt()* para el acceso de algún canal a través del puerto al cual esta ligado. *my_rd_interface* es una interfaz que define un método (función) llamada *rd( )* para el acceso de algún canal a través del puerto que esta ligado a el. *my_channel* implementa las funciones *rd()* y *wt()*. Cuando *my_proc* llama rd() o wt() el codigo es ejecutado desde *my_channel.*

Todos estos conceptos serán explicado mas en detalle mas adelante

Modelo del tiempo.

El tipo de datos utilizado para tiempo es un número entero 64-bit sin signo. La unidades de tiempo se designan de la siguiente manera: *SC_FS*, *SC_PS*, *SC_NS*, *SC_US*, *SC_MS*, *SC_SEC.* El tipo sc_time se proporciona para especificar valores del tiempo.

Syntax:

```
// time_value of type unit64
sc_time var_name(time_value, time_unit);
```

Varias funciones se proporcionan para las simulaciones que controlan y reportan la información del tiempo. Entre ellas se encuentran sc_stop; sc_time_stamp; sc_simulation_time(). Para mas información sobre estas funciones refiérase al punto A1 del anexo

Planificador De SystemC (SystemC Scheduler)

La simulación es basada en eventos. Se ejecutan los procesos y sus salidas son actualizadas (updated) basados en la ocurrencia de dichos eventos. La biblioteca de SystemC incluye un planificador (Scheduler) que maneja todos los eventos y

actualizaciones del canal. Se planifica la ejecución de los procesos basados en su lista de sensibilidad. Para conocer los pasos del planificador refiérase al punto A2 del anexo

Tipos de Datos en SystemC

En SystemC existen distintos tipos de datos los cuales son mas adecuados para el modelamiento de hardware, por ejemplo es necesario que existan tipos que cumplen con Fixed precision integres, Arbitrary precision integres, 4-valued logic type, entre otros.

A continuación se explicaran brevemente algunos de los distintos tipos existentes (mas información se puede encontrar en el punto A3 del anexo).

sc_uint<>, sc_int<> : se utilizan para definir enteros de precisión fija con y sin signo. Son representados como un arreglo de bits.

Sintaxis: sc_int<length> variable_name, variable_name, ...;

sc_uint<length> variable_name, variable_name, ...;

en donde lenght depende del tamaño del arreglo

sc_logic : representa la lógica de 4 valores. El valor se interpreta como variable de un solo bit, no como número. Es útil para modelar lógica de triple estado.

Sintaxis: sc_logic variable_name, variable_name, . . . ;

sc_fixed<>, sc_ufixed<>: Los modelos de un nivel más elevado pueden utilizar números de punto flotante para modelar operaciones aritméticas. Estos números se convierten a menudo a los tipos de datos de punto fijo para la puesta en práctica de hardware. Estos tipos son precisión arbitraria y tienen argumentos estáticos.

Sintaxis:

sc_fixed<wl, iwl, q_mode, o_mode, n_bits> object_name, object_name, ... ;

sc_ufixed<wl, iwl, q_mode, o_mode, n_bits> object_name, object_name, ... ;

Interfaces

Un interfaz define un conjunto de métodos de acceso. Es puramente funcional y no proporciona la implementación de los métodos. Los interfaces están ligadas a los puertos (definen qué se puede hacer a través de un puerto particular). La puesta en práctica de interfaces se hace dentro de un canal. Un canal que implanta un interfaz debe implementar todo los métodos definidos.

En SystemC existen las interfaces siguientes: *sc_signal_in_if, sc_signal_inout_if, sc_fifo_in_if, sc_mutex_if*, entre otras. Cada una de estas interfaces implementa distintos métodos

Canales

Los canales se utilizan para la comunicación entre los procesos dentro de los módulos y entre los módulos. Dentro de un módulo un proceso puede tener acceso directamente a un canal si este esta conectado con un puerto de un modulo.

Hay dos clasificaciones de canales, primitivo y jerárquico. Los canales primitivos no tienen ninguna estructura visible y no pueden tener acceso directamente a otros canales primitivos. Hay varios tipos de canal primitivos proporcionados en SystemC: *sc_signal, sc_signal_rv , sc_fifo , sc_mutex, sc_semaphore, sc_buffer* . Los canales jerárquicos son módulos, este tipo de canales pueden contener los procesos, otros módulos etc. Ademas una característica importante de este tipo de canales es que pueden tener acceso directamente a otros canales jerárquicos. A continuación, por la extensión del tema, se detallaran brevemente solo alguno de estos (Para mas información refiérase al punto A4 del anexo):

sc_signal :implementa la interfaz *sc_signal_inout_if.* Se utiliza tanto para comunicaciones punto a punto o multipunto

Syntax: sc_signal<T> signal_name, signal_name, ... ;

Donde T representa el tipo de dato, por ejemplo int, char, sc_int<10>

sc_fifo: implementa las interfaces sc_fifo_in_if y sc_fifo_out_if. Un canal sc_fifo implementa una FIFO. Es una conexión punto a punto y puede ser conectado solamente con un entrada y un puerto de salida al conectar entre los módulos.

Syntax: sc_fifo<T> channel_name, channel_name, ... ;

sc_mutex: implementa la interfaz sc_mutex_if . Se utiliza para el acceso seguro a un recurso compartid**o**

Syntax: sc_mutex mutex_name, mutex_name, . . . ;

Módulos

Un módulo es un container. Cada modulo se describe usando un header file (nombre_modulo.h) y un archivo de implementacion (nombre_modulo.cpp). Un modulo puede contener procesos e instancias de otros módulos

Syntax:
```
SC_MODULE ( module_name) {
  // body of module
} ;
```

Puertos

Puertos son creados desde la clase sc_port y son ligadoa a un tipo de interfaz.

Syntax: sc_port<interface_type, N> port_name, port_name,... ;

N es el numero de canales que pueden ser conectados al puerto

Constructor

Ya que un módulo es una clase de C++ tiene un constructor. El constructor es una función del mismo nombre que la clase, sin el tipo de retorno. Se utiliza para crear y para inicializar un caso de un módulo y para crear las estructuras de datos internas que se utilizan en el módulo y para inicializar estas estructuras de datos a los valores conocidos. Se registran los procesos y los módulos instanciados dentro del constructor.

SystemC proporciona una macro especial (SC_CTOR) para el constructor.

Puertos especializados

El sc_signal (y por lo tanto el sc_buffer), el sc_signal_rv, y los canales del sc_fifo tienen puertos especializados asociados a ellos que se proporcionan para la conveniencia. Pueden ser utilizados en vez de usar el sintaxis del sc_port para estos canales.

Eventos

Sintaxis: sc_event event_name1,event_name2,.....;

Los eventos son el objeto básico de sincronización, se utilizan para la sincronización entre procesos, son solo de control y se declaran dentro de un modulo.

Para gatillar un evento se llama al método *notify ( ).* La notificación de un evento puede ser inmediata (Immediate), Retardada (delayed), o temporizada (timed). Para cancelar las notificaciones pendientes de un evento se utiliza el método *cancel ( )* (Solo funciona para notificaciones del tipo delayed y timed).

Procesos

En los procesos se describe la funcionalidad. No es posible llamar a procesos directamente en un código, de hecho un proceso es invocado por el kernel basado en las lista de sensibilidad (estáticas o dinámicas). Los procesos son muy similar a los métodos de C++ y las funciones, salvo algunas excepciones. Algunos procesos se ejecutan cuando son llamados y retornan al mecanismo que los llamo (se comportan como una función), en cambios otros procesos son llamados una vez y luego se pueden suspender ellos mismo y reanudar su ejecución luego (como los hilos).

Los procesos no son jerárquicos, es decir, no es posible tener procesos dentro de otros procesos. Para comunicarse con otros procesos se utilizan, como ya se menciono los canales y/o eventos. SystemC soporta 3 distintos tipos de procesos:

- Procesos Hilos (SC_THREAD)
- Metodos (SC_METHOD)
- Clocked Threads (SC_CTHREADS) (no muy usado, por tal motivo no se detallara)

Procesos Hilos (Thread Processes)

Un proceso hilo puede ser activado o reactivado a través de una lista de sensibilidad estática o dinámica y deben ser inicializado durante el comienzo de la simulación. Una vez que un proceso hilo es (re)activado, las declaraciones se ejecutan hasta que se encuentra una declaración del *wait().* En la declaración del *wait(),* se suspende la ejecución de proceso. En la reactivación siguiente, la ejecución de proceso empieza con la declaración que sigue el *wait().*

Los procesos hilos se caracterizan, como su nombre lo indica, en que poseen su propio hilo de ejecución, corren solamente cuando son empezado por el planificador de SystemC, son implementados con loop infinito (con el fin de asegurar que los procesos puedan ser repetidamente reactivados y permitir su suspensión y reactivación en cualquier parte del código) y poseen su propio stack de ejecución en donde las variables locales son guardadas. Para la creación de un proceso se deben seguir los siguientes pasos:

1.- Declaración: Se declaran dentro de un modulo como una función que no tomo ningún argumento y retorna vacío.

2.- Definición: Es recomendable colocar la definición en un archivo de implementación separado llamado *nombre_modulo.cpp*

3.- Registro: La función se registra dentro del constructor usando la macro *SC_THREAD( ).* Esta macro toma un argumento el cual es el nombre de la función a ser registrada en el Kernel

4.- Declaración de la Sensibilidad Estática (opcional): La lista de sensibilidad no cambia durante la ejecución. SystemC provee la función *sensitive( )* para agregar eventos a la lista de sensibilidad estática de un proceso hilo

Metodos (Method Processes)

Son similares a un proceso hilo excepto que los métodos no pueden ser suspendido, es decir, se invocan cuando su lista de sensibilidad cambia, pero todo el cuerpo debe ser ejecutado y además, a diferencia de los hilos, una vez ya ejecutados retorna al simulador. Al igual que los hilos para crear un método es necesario realizar los siguientes 4 pasos:

1.- Declaración: Se declaran como una función dentro de un modulo como una función que no tomo ningún argumento y retorna vacío.

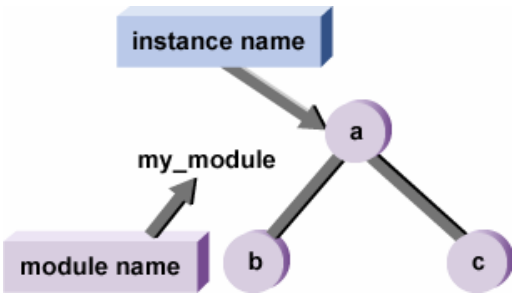
2.- Definición: Al igual que los hilos es recomendable colocar la definición en un archivo de implementación separado llamado *nombre_modulo.cpp*

3.- Registro: La función se registra dentro del constructor usando la macro *SC_METHOD( ).* Esta macro toma un argumento el cual es el nombre de la función a ser registrada en el Kernel

4.- Declaración de la Sensibilidad Estática (opcional): La declaración se realiza exactamente igual a los hilos.

Jerarquía de los Módulos.

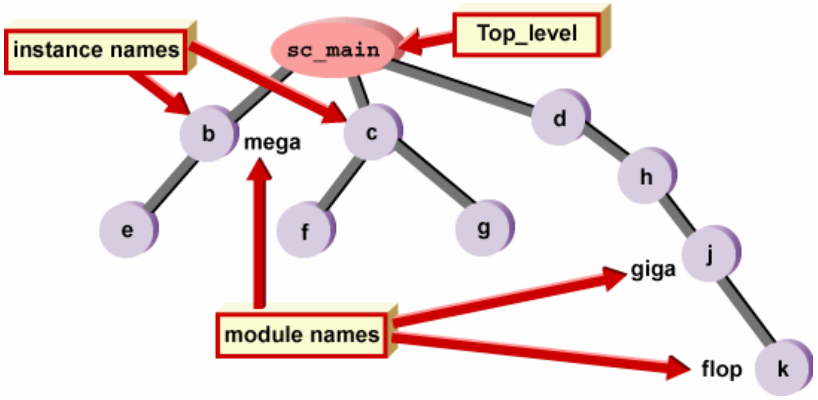
Como SystemC trabaja con módulos esto facilita el trabajo jerárquico. Se puede definir módulos dentro de otros módulos.



Para mas detalle sobre como definir módulos jerárquicos léase el punto A5 el anexo al final. A continuación se describen algunas funciones útiles para los módulos jerárquicos.

La función sc_main( )

SystemC no requiere un módulo de nivel superior. En cambio utiliza la función especial sc_main. Esta función se encuentra en un archivo nombrado main.cpp o main.cc, la función es llamada por SystemC y es por aquí por donde entra el código



Sintaxis:

```
int sc_main (int argc, char *argv [ ] ) {
   // body of function
   return 0 ;
}
```

Esta función retorna un valor entero, 0 en el caso que no hubo error. Los argumentos se envian al igual que en C++. Para los paso para asignación de módulos a la funcion vea el punto A6.

Canales.

Lo primero dentro del sc_main( ) es típicamente la creación de los canales para conectar los módulos iniciados dentro de sc_main(). Esto es solamente necesario si hay más de un módulo iniciado. Vea el ejemplo en el punto A7 del anexo

Escala de tiempo.

La escala de tiempo por defecto para SystemC es 1 picosegundo. Se puede cambiar la escala de tiempo usando la siguiente función. Usted puede especificar una diversa escala de tiempo solamente una vez. Se especifica durante el tiempo de la elaboración en sc_main(). Debe ser especificado antes de cualquier declaración del sc_time.

Sintaxis:

```
     sc_set_time_resolution   (value);   //value   of   type
sc_time
```

Relojes.

Un evento en SystemC, que significa que la ejecución de declaraciones puede ocurrir en cualquier tiempo de la simulación. El tipo sc_clock se puede utilizar para generar un reloj. Hay también otras maneras de generar una señal reloj, por ejemplo accionando la palanca de un sc_signal<bool > o periódicamente notificando un acontecimiento. Un reloj tiene: período, duración del ciclo; ciclo de trabajo, fracción del periodo que la señal es alta; canto de subida, cuando el valor del reloj pasa de 0 a 1; canto de bajada, cuando el valor del reloj pasa de 1 a 0.

Sintaxis:

```
sc_clock clock_name ("name", period, duty_cycle,
      start_time, positive_first ) ;
```

Para un ejemplo de sintaxis vea el ejemplo en el punto A8, del anexo.

Defenicion de módulos.

Unos o más módulos se definen dentro de sc_main(). La sintaxis es similar a definir un módulo dentro de un módulo excepto que puesto que el sc_main() es una función, la definición y la inicialización se puede hacer en un paso. Con mas detalle sobre la definición de módulos se explican en el anexo al final.

sc_start( )

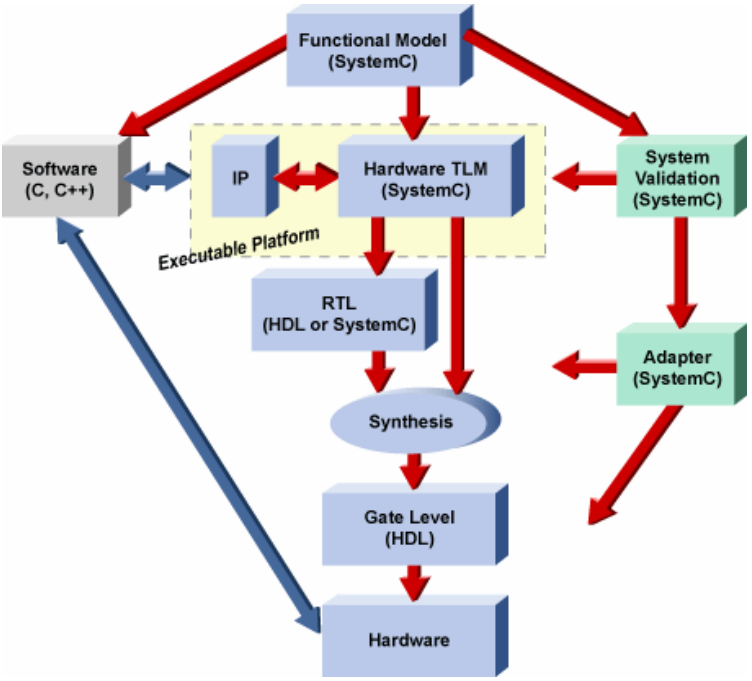
Al final de la función sc_main( ) y antes de la declaración de vuelta, está la función del sc_start(). La ejecución de esta declaración marca que la etapa de la elaboración término y da comienzo a la simulación. El sc_start(arg) tiene un argumento opcional, el argumento especifica el número de las unidades del tiempo para simular. Si es el argumento es nulo la simulación funcionará por siempre

Modelamiento.

Ahora en SystemC es posible modelar los niveles funcionales, los de transacción, los de transferencia d archivos y la transición entre niveles.

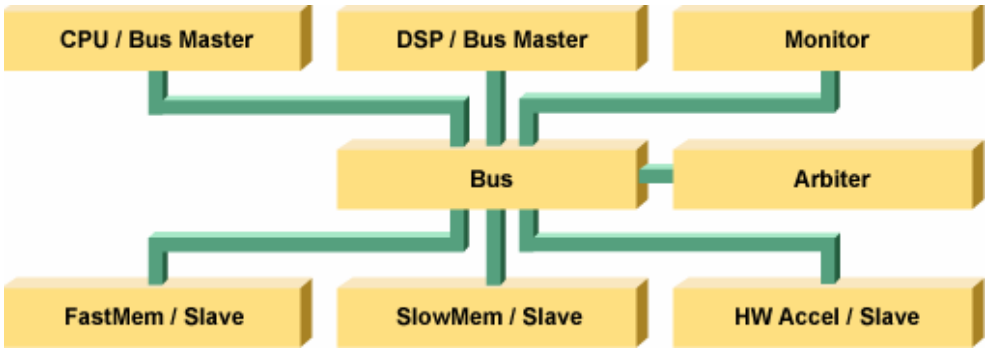
El Modelo funcional.

El modelo a nivel funcional describe, el modelamiento hecho a niveles sobre la transferencia (TLM), abarca los modelos arquitectónicos del sistema (SAM) y los modelos de funcionamiento del sistema (SPM). El modelar a este nivel, es algorítmico en naturaleza y se puede medir en forma sincronizada o no sincronizada. Los modelos pueden representar software, el hardware o ambos. El comportamiento se describe típicamente como los generadores y consumidores de datos. A los procesos se le pueden asignar tiempo, para propósitos del análisis de funcionamiento. Esta sincronización no es de ciclo exacto sino que describe el tiempo generado, consume datos, modelar el buffering o acceso a datos. El comportamiento de las interfaces entre los módulos se describe usando protocolos de comunicación. El tiempo se puede asignar al transporte de los datos para el análisis de funcionamiento. Estos tipos de modelos se utilizan para explorar arquitecturas, para la prueba de algoritmos y para modelar y el análisis de funcionamiento.



Modelado a nivel de transferencia

Los modelos de nivel de transacción describen típicamente el hardware. Muchos diseños a este nivel, por ejemplo los diseños típicos de System on Chip (SoC), tienen unos o más buses modelados como canales. En el bus central están los dispositivos del master -- tales como procesadores -- y los dispositivos del esclavo -- tales como memorias, aceleradores y así sucesivamente.



Las transferencias de datos se modelan como operaciones, por ejemplo lectura y escritura. El comportamiento modelo son de descripciones algorítmicas típicamente medidas en el tiempo. Las interfaces no tienen detallado el nivel pin. Dependiendo del nivel de modelamiento deseado es posible tener o no precisión cíclica Las velocidades resultantes permiten uso de un diseño de TLM como plataforma de la ejecución para aplicaciones múltiples: como desarrollo de software, verificación de plataforma y modelos para IP.

Modelando el nivel de operación del registro.

Los modelos de RTL describen el hardware y contienen una descripción funcional detallada completa. Cada registro, cada bus, cada bit se describe para cada ciclo de reloj.

En general, escribir un código RTL en SystemC es similar al código de la escritura RTL en los idiomas descriptivos del hardware como Verilog o VHDL. Si el código está para la síntesis, entonces las mismas reglas generales se aplican.

Para referenciar un modelo TLM a un modelo de RTL hay varias áreas a considerar:

1.- Sincronizado y no sincronizando descripciones algorítmicas, necesitan ser substituidas por el ciclo del pin exacto o colocan descripciones exactas de la transferencia.
2.- Los canales abstractos tales como sc_fifo necesitan ser substituidos por los canales del hardware tales como sc_signal.
3.- Los tipos de datos de C++ pueden necesitar ser substituido por los tipos de datos de SystemC. Por ejemplo si se modela un autobús de triple estado 32-bit, un tipo del sc_lv necesitaría ser utilizado en vez de un intero sin signo.
4.- Si se definió tipos, tales como el tipo del cmd, necesitaría ser substituido por los tipos de C++ o de SystemC.

Aunque los procesos del hilo se pueden utilizar en el nivel del RT, los procesos de método se utilizan a menudo por razones del funcionamiento. Los procesos de método tienen menos gastos indirectos de la conmutación de contexto que procesos.

CONCLUSIÓN

En el presente informe se mostró una pequeña introducción acerca de los conceptos y principios de funcionamiento de SystemC. Por tal motivo no se profundizo mucho en diversos temas, si no mas bien se mostró a manera de ejemplo solo una pequeña parte de estos. Pero además se deja a disposición un anexo en el cual se especifica de manera mas detallada a cada uno de los puntos a tratar y además se deja como ejemplo el modelado de un exor el cual se explicara mas en detalle en la presentación a realizar

REFERENCIAS

Links:

http://www.systemc.org
http://www.systemc.org/docman2/ViewCategory.php?group_id=4&category_id=2
http://www.doulos.com/knowhow/systemc/tutorial/
http://www.doulos.com/knowhow/systemc/tutorial/module_and_processe/