Utilización de un reloj global para el modelado de un ambiente simulado distribuido

Diego Encinas[1], Emmanuel Frati[1,2], Germán Caseres[1], Hugo Ramón[1], Marcelo Naiouf[1]

[1]Instituto de Investigación en Informática (III-LIDI). Facultad de Informática. UNLP.
Calle 50 y 120 – 2do piso – (1900) La Plata, Argentina.
[2]Consejo Nacional de Investigaciones Científicas y Técnicas.
{dencinas, fefrati, gcaseres, hramon, mnaiouf}@lidi.info.unlp.edu.ar

Resumen. Los Sistemas Distribuidos de Tiempo Real Críticos deben ejecutar algoritmos en plazos de tiempo adecuados a los requerimientos de la implementación. Durante la etapa de verificación y validación del hardware, pueden encontrarse medidas que impliquen la realización de cambios o modificaciones en los mismos. Una manera de disminuir la complejidad en la generación de hardware es desarrollar una simulación específica de éstos. Se propone el diseño e implementación de un modelo utilizando el framework SystemC. Debido a las características del sistema a modelar se utiliza un reloj global que representa de manera abstracta los métodos de sincronización de la capa física del protocolo CAN (Controller Area Network).

Palabras claves: Sistemas Distribuidos de Tiempo Real Críticos, SystemC, Simulación, Reloj global, Controller Area Network, Red de Sensores, Sincronismo de Procesos, Modelos.

1 Introducción

Los Sistemas Distribuidos de Tiempo Real Críticos deben ejecutar algoritmos en plazos de tiempo adecuados a los requerimientos de la implementación. Durante la etapa de verificación y validación del hardware de estos sistemas pueden encontrarse medidas que impliquen la realización de cambios o modificaciones en los mismos. Una manera de disminuir la complejidad en la generación de hardware es desarrollar una simulación específica de los mismos. Para realizar esto, se propone el diseño e implementación de un modelo utilizando el framework SystemC [1].

Este artículo constituye un aporte sobre la utilización de herramientas sincrónicas para el modelado de sistemas que requieren de un protocolo de comunicaciones de sistemas distribuidos de tiempo real. En particular, se realiza una evaluación de un modelo desarrollado con SystemC para analizar y predecir el comportamiento de una red de sensores aplicado a sistemas de vuelo.

Aunque se encuentran pocos trabajos referidos a este tema en el campo de los sistemas distribuidos de tiempo real siempre ha existido una gran tendencia al uso de simuladores. Específicamente en el desarrollo de modelos con este framework, para efectuar la verificación del bus de datos en estos sistemas, se han realizado experiencias que abarcan el área automotriz [2] como el de aviónica [3].

El trabajo está organizado de la siguiente forma: en la segunda sección se describe el contexto del modelo a construir y la herramienta elegida para tal fin. En la tercera sección los problemas considerados al momento de construir el modelo. En la cuarta sección se detallan los modelos desarrollados y el adoptado para realizar los experimentos. En la quinta sección se muestran los resultados obtenidos luego de realizar los experimentos. Finalmente en la sexta sección se presentan las conclusiones del trabajo.

2 Contexto

Un sistema aeroespacial cuenta con una serie de sensores que miden los valores de las diversas magnitudes involucradas en los algoritmos de navegación a ser resueltos. Debido a la complejidad de estos sistemas, se han desarrollado numerosos protocolos de comunicaciones serie no-estandarizados para cada proyecto. Como esto implica un costo muy alto, tanto de dinero como de tiempo, surge la necesidad de utilizar protocolos estandarizados como el que provee el bus CAN (Controller Area Network). Este estándar es conocido por ser un protocolo de comunicaciones serie de alta confiabilidad, robustez y performance; además es apropiado, mediante un desarrollo de capa de aplicación, para el control de sistemas distribuidos de tiempo real. En principio este protocolo y su bus asociado fue desarrollado por Bosch para la industria automotriz pero actualmente es aceptado por distintas compañías de aviación e instituciones como NASA, ESA, Airbus, Boeing, Eurocopter, entre otros.

2.1 Protocolo CAN

En el protocolo CAN el acceso al medio se realiza de la siguiente manera:
Cada nodo conectado posee un ID que lo hace único en la red. Estos nodos se encuentran "escuchando" el bus a la espera de un SOF (Start Of Frame). Una vez que detecta la ocurrencia de este evento, si el nodo desea transmitir, intenta depositar en el bus el ID del nodo destino al que se quiere enviar la trama. Si dos o más nodos intentan transmitir al mismo tiempo, el conflicto de acceso al bus es resuelto mediante arbitraje (bite-wise) usando para ello el ID que cada nodo intenta transmitir. Este método de control de acceso al medio es conocido como "Carrier Sense Multiple Access with Collision Detection and Arbitration on Message Priority" (CSMA/CD + AMP) [4]. El ID de mayor prioridad determinará qué nodo continúa con la operación de transmisión y cual deberá esperar al siguiente ciclo. Cuando el ID ha sido depositado completamente en el bus, debido a que se ha resuelto el arbitraje, se continúa enviando el resto de la trama. Al finalizar enviará un EOF (End Of Frame) para avisar al nodo destino que la trama está completa. Por otro lado una vez terminada la etapa de arbitraje los nodos verificarán si el ID transmitido coincide con su propio ID. Sólo aquel para el que se cumpla esta condición continuará recibiendo la trama hasta que ocurra un EOF.
Por las características del protocolo se debe asegurar que el tiempo entre SOF/EOF no supera K unidades de tiempo. Por lo tanto, si la trama comienza a enviarse en el instante J, entonces se puede estar seguro que el mensaje será recibido a lo sumo en el instante (J+K). Esta característica se comprobó experimentalmente en [5].

En este protocolo se utiliza un método de transmisión de datos sincrónico: esto significa que cada nodo transmite y recibe usando el mismo período de reloj, y todos los períodos de reloj en la red se basan en un único punto de referencia. Esto hace las transmisiones de datos más eficientes, pero dificulta mantener sincronizadas señales de reloj sin ninguna clase de señal de referencia central. Comúnmente, las señales de reloj pierden su sincronización debido al *drift* del oscilador de cada nodo, retardos de propagación y errores de fase. Los nodos CAN utilizan dos métodos diferentes para sincronizar sus señales de reloj: Hard Synchronization y Resynchronization [4].
Los sistemas que utilizan este tipo de protocolo deben ejecutar algoritmos en plazos de tiempo adecuados a los requerimientos de la implementación y generalmente se encuentran expuestos a condiciones de funcionamiento severas, que representan una fuente potencial de fallas. Una manera de disminuir la complejidad en la generación de hardware para incorporar tolerancia a fallas a estos sistemas, es desarrollar una simulación específica de los mismos.
Modelar a nivel de sistemas tiene como ventaja reducir la complejidad e incrementar la comprensión del sistema a un nivel de abstracción alto. Con la simulación resultante se podrán detectar cuellos de botella más allá de la implementación del hardware, permitiendo explorar y validar distintas posibilidades en la primera fase de diseño, lo cual mejora significativamente la etapa de desarrollo. Para efectuar el modelado se utiliza una metodología que contempla el nivel de abstracción de sistemas llamado TLM (Transaction Level Modeling). De esta manera se consigue una mayor eficiencia y rapidez a costa de no modelar a un nivel de abstracción más bajo como el denominado RTL (Register Transfer Level) [6]. Actualmente SystemC es el lenguaje más apropiado para utilizar la metodología TLM en el diseño y verificación de sistemas.

2.2 SystemC

SystemC es un lenguaje de modelado y diseño de sistemas electrónicos cuyo objetivo es proveer a los desarrolladores y arquitectos de sistemas (software y hardware) un estándar basado en C++ [7]. Consiste en un framework que provee los constructores necesarios para modelado de hardware y que permite representar conceptos de tiempo, tipos de datos de hardware, comunicación y concurrencia. Esto permite al usuario escribir un conjunto de funciones C++ que son ejecutadas bajo el control de un scheduler en un orden que imita el paso de tiempo simulado y que se sincronizan y comunican emulando sistemas electrónicos que contienen hardware y software embebido. Es decir, los objetos instanciados basados en SystemC pueden comunicarse en un entorno simulado de tiempo real y ejecutar operaciones secuenciales sincronizadas. Particularmente las estructuras del tipo interfaz, puerto y canal, proveen una gran flexibilidad para el modelado de las comunicaciones y el refinamiento de todo el modelo.
La utilización de SystemC permite afrontar la complejidad de los diseños de sistemas modernos mediante la utilización de técnicas como abstracción, reutilización de diseño, reutilización de proyecto, entre otros.

3 Problema

3.1 Conceptos de diseño

Generalmente antes de iniciar la construcción de un modelo de simulación se realiza un análisis del problema, recolección de información estructural que afecte al mismo y recolección de datos [8]. Como en todo modelo pueden aplicarse restricciones para reducir la complejidad del sistema y mejorar su rendimiento. En este trabajo se considerarán dos problemas particulares: el arbitraje del bus y la sincronización del sistema.
El modelo que debe desarrollarse para esta aplicación es el llamado modelo de simulación de eventos discretos porque puede considerarse discreto, dinámico y estocástico.

3.2 Problema de arbitraje

El mecanismo de transacción usado por el protocolo CAN para transferir datos es similar al modelo productor consumidor en donde cada nodo envía mensajes sin requerir confirmación y de la forma multicast o broadcast. El tipo de acceso al bus CAN es multi-master; por este motivo, el derecho a acceder al bus no es asignado por un nodo central pero de todas formas todos los nodos pueden intentar acceder al mismo tiempo. Puede decirse que el arbitraje se resuelve por un procedimiento que hace uso de los identificadores que se encuentran al mismo tiempo en el bus.
Como puede observarse para diseñar el modelo del sistema se debe tener en cuenta la dependencia de los procesos que se ejecutan en los nodos y en el bus en el momento de la simulación. En este caso, existirán procesos de escritura o transmisión de trama y procesos de lectura o recepción. Al existir un recurso compartido entre los dos procesos mencionados en el bus deben tomarse medidas ya conocidas en programación concurrente como sincronización basada en variables compartidas [9]. Dada la naturaleza inherentemente concurrente/paralela de los sistemas de tiempo real, se aplican técnicas muy utilizadas en programación concurrente para solucionar los problemas de comunicación.

3.3 Problema de sincronización

La evolución en el tiempo de un sistema implica la evolución de sus procesos y variables. Para que el avance de éstos se efectúe correctamente es necesario un reloj global, cuya función es la de marcar un único tiempo de referencia para el sistema completo.
Existen dos posibilidades de mecanismo de avance de tiempo para el reloj de simulación: uno es el llamado avance de tiempo dirigido por eventos, donde el tiempo de simulación avanza cuando ocurre un evento y sólo en ese momento el reloj junto con el sistema es actualizado. El otro es el avance de tiempo discreto, en donde el reloj de simulación avanza con pasos de tiempo fijos. Para éste último los cambios que pueden producir los eventos de cada periodo se actualizan al final del mismo. Básicamente el segundo mecanismo es un caso especial del primero [8].

4 Modelos y trabajo experimental

Con el fin de modelar comportamiento de una red de sensores aplicado a sistemas de vuelo se implementó una red de nodos encargados de enviar y recibir mensajes de forma sincronizada.
Para la construcción del modelo de simulación se reprodujo el funcionamiento del protocolo CAN, y en este caso se diseñaron métodos que reflejen los procesos de escritura y lectura de tramas en la red. Debido al comportamiento productor-consumidor de los nodos y la presencia del recurso compartido que representa el bus surgió la necesidad de aplicar técnicas de programación concurrente como la exclusión mutua para proteger el acceso al medio mediante regiones críticas.
En cuanto a la selección del mecanismo de avance de tiempo se desarrollaron modelos para los dos casos mencionados.
En ambos se modeló el escenario en el que un nodo emisor envía un mensaje a través del bus a un nodo receptor. Los nodos fueron creados por medio de módulos que contienen procesos encargados de armar los objetos tipo tramas (TramaCan) y enviarlos por medio de procesos de escritura a través de puertos conectados al bus. Estas conexiones se realizaron utilizando interfaces que se definieron para tal fin.
Los módulos creados se integran a un módulo superior que actúa como testbench (CAN) para la validación del modelo a través de simulación.

4.1 Modelo dirigido por eventos

En la Figura 1 se puede observar el escenario que se modeló con los objetos antes mencionados para simular la transmisión de un mensaje de un nodo a otro. En este caso el nodo receptor (Nodo2_i) cumple las funciones de monitor de tráfico en la red y sólo lee del bus (Bus_can_i) cuando se dispara un evento indicando que una nueva trama ha sido escrita por el nodo emisor (Nodo1_i).

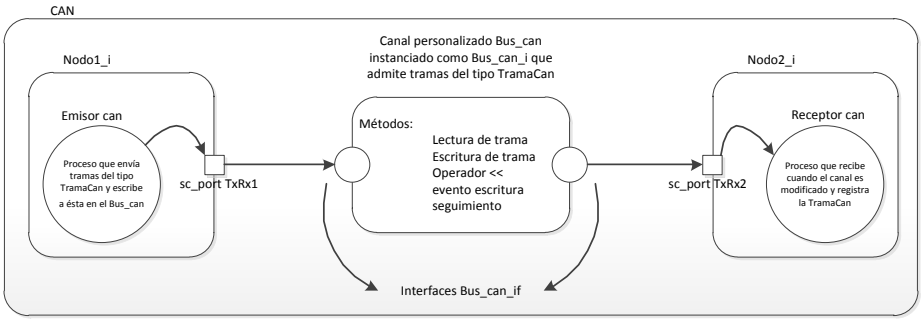

Figura 1. Modelo dirigido por eventos.

Para modelar el acceso concurrente al bus se redefinió la semántica de escritura de manera tal de hacer efectiva la misma sólo si supera el proceso de arbitraje.
Dada la naturaleza sincrónica del problema la competencia por el bus se produce en periodos conocidos. Consecuentemente la ejecución de los objetos que conforman al

sistema estará supeditada a éstos periodos [10].
Por este motivo la solución por eventos no es la más natural teniendo en cuenta que las herramientas de síntesis (VHDL, Verilog) generan código completamente sincrónico. Esto debe considerarse ya que luego de la etapa de verificación del modelo generalmente se prosigue con la de implementación física del mismo.

4.2 Modelo dirigido por tiempo

En la Figura 2 puede verse que se modeló el caso de uso anterior pero utilizando otros objetos y procesos que ofrece SystemC. El proceso o hilo sincronizado se ejecuta con una periodicidad dada por un objeto tipo reloj conectado a dicho proceso. Dada la forma de onda cuadrada del reloj, SystemC provee la posibilidad de configurar el flanco que provocará la ejecución de los procesos sincronizados. Se optó por el flanco ascendente ya que es el que implementa el protocolo CAN.
En este modelo no sólo se ejecutan procesos sincrónicos en los nodos, sino que existe un proceso (hiloBus) en el bus que se ejecuta periódicamente de acuerdo al reloj del sistema. La función de este proceso es la de dividir la transmisión en dos etapas: escritura o arbitraje y lectura o establecimiento de trama.

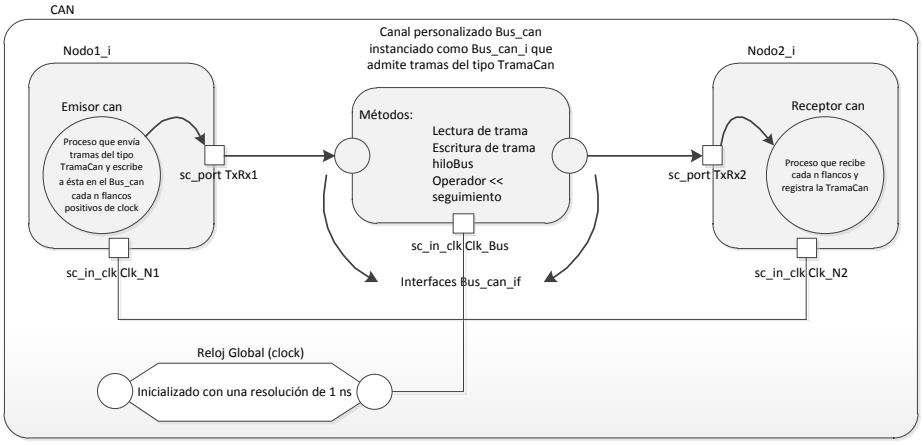

Figura 2. Modelo dirigido por tiempos.

De esta manera se sincronizaron todos los procesos y se facilitó la implementación de los métodos necesarios para solucionar el problema de concurrencia en el arbitraje del bus.

4.3 Modelo desarrollado

Finalmente el sistema modelado para probar la funcionalidad del bus se puede ver en la Figura 3. En la misma, dos nodos (Nodo1_i y Nodo3_i) sólo transmiten mensajes con distintos identificadores (111 y 333 respectivamente), un tercer nodo (Nodo2_i) cumple las funciones de analizador de tráfico y un canal (Bus_can) por el que sólo

circulan tramas (TramaCan) que contienen los campos propios del protocolo CAN. Todos los objetos están conectados a un reloj global que representa de manera abstracta los métodos de sincronización de la capa física del protocolo CAN.

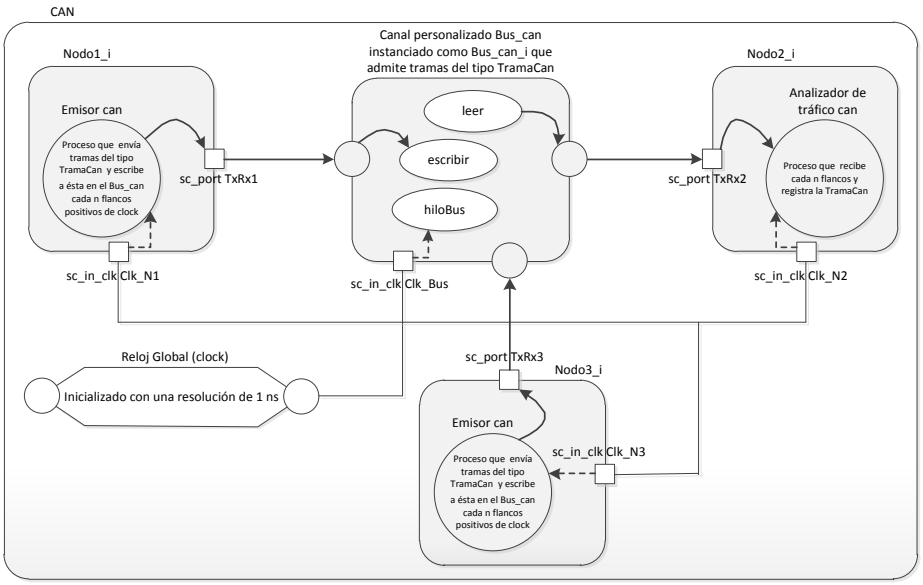

Figura 3. Modelo desarrollado.

Este es el típico escenario de una red de sensores para un sistema distribuido de tiempo real en donde los nodos 1 y 3 representarían a los dispositivos que envían información periódicamente a un nodo central. Un ejemplo práctico de esta situación es caracterizar el comportamiento dinámico de un sistema aeroespacial utilizando una red de sensores inerciales del tipo IFOG (Interferometric Fiber-Optic Gyroscope) que se comunican utilizando el protocolo CAN a una Computadora de Navegación encargada de procesar las mediciones inerciales [11].

La sincronización de los procesos del modelo desarrollado se implementó a través de la definición de ventanas de escritura y lectura que se corresponden con las etapas de escritura y lectura definidas por el proceso del bus.

Las ventanas fueron diseñadas para que tengan un intervalo de tiempo fijo y representan las restricciones que poseen los mensajes cuando llegan en un momento o periodo en el que no pueden transmitir a pesar de tener una alta prioridad, este periodo es también conocido como busy period [12]. Es decir, cuando un mensaje llega durante el intervalo de tiempo ventana de lectura debe esperar hasta el inicio del intervalo ventana de escritura para competir con otros posibles mensajes en el momento de arbitraje.

En la Tabla 1 se muestran los cuatro escenarios posibles que pueden encontrarse con el modelo desarrollado.

Tabla 1. Resultados de pruebas.

Escenarios	Ventana Escritura			Ventana Lectura		
	Nodo1_i	Nodo3_i	Nodo2_i	Nodo1_i	Nodo3_i	Nodo2_i
0	W	W	-	-	-	R
1	W	-	-	-	-	R
2	-	W	-	-	-	R
3	-	-	-	-	-	R

En el escenario cero (0) cuando los dos nodos transmisores (1 y 3) intentan transmitir (W) al mismo tiempo durante el intervalo de la ventana adecuada (Ventana Escritura) el nodo monitor (2) registra (R) la trama con el identificador que ha ganado el bus en el arbitraje. En este caso el mensaje con identificador 111 tiene mayor prioridad por ser el de menor valor como indica el protocolo.
En los escenarios uno y dos, sólo uno de los nodos transmisores intenta transmitir a la vez y por lo tanto pueden ganar el bus al no competir con ningún nodo en la etapa de arbitraje. El nodo monitor registra las tramas con identificador 111 y 333 cada vez.
Finalmente en el último escenario (3) ninguno de los nodos transmite y el nodo monitor registra el estado desocupado del bus.
En todos los escenarios descriptos se muestra que si los dos nodos intentan transmitir durante el intervalo de Ventana Lectura no podrán hacerlo (-) y deberán esperar hasta el próximo intervalo de Ventana Escritura.

5 Resultados

Para verificar el funcionamiento del modelo desarrollado se realizó un seguimiento (tracing) de los datos en función del tiempo dentro del bus (Bus_can) y del reloj global dentro del sistema (CAN). Este seguimiento se consiguió por medio de métodos de registro de señales que provee SytemC. Los datos se depositaron en un archivo de formato VCD (Value Change Dump).
En la Figura 4 se puede ver el resultado de la simulación al representar los cuatro escenarios descriptos para el modelo desarrollado.

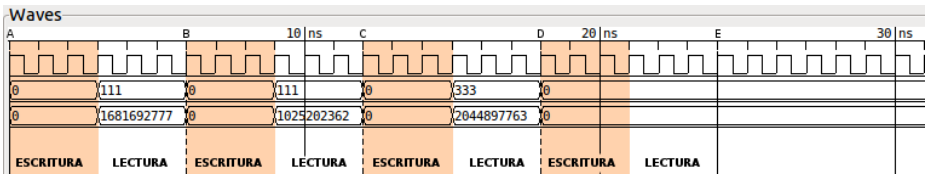

Figura 4. Resultados de la simulación

En la figura se puede observar la señal del reloj global seguida por las señales ID y datos. La resolución del reloj es de un nanosegundo (nseg) y el tiempo de simulación es de 31 nseg.

Cada intervalo de comunicación está formado por una ventana de escritura y una de lectura diferenciadas por su color. Cada ventana tiene una duración de 3 nseg, completando el intervalo en 6 nseg. Dado que no se ha completado el arbitraje hasta que se finaliza la ventana de escritura, el valor de las señales ID y datos es cero.
Se puede ver que entre los puntos A y B la trama con identificador (ID) 111 ha ganado el bus y depositado su campo de datos. En el intervalo que marca B y C sólo fue transmitida la trama con ID 111. En el siguiente intervalo, entre C y D ocurre lo mismo que en el caso anterior pero con ID 333. Finalmente en el último intervalo, entre D y E, ningún nodo transmite y continua la red de esa manera.
Como en el primer escenario el nodo con menor prioridad (Nodo3_i) pierde en la etapa de arbitraje frente a Nodo1_i se comprueba que su mensaje no es insertado en el bus.

6 Conclusiones

En este trabajo se ha realizado una evaluación de un modelo desarrollado con el framework llamado SystemC para analizar y predecir el comportamiento de un bus en una red de sensores, en particular aplicado a sistemas de vuelo.
La utilización de un reloj global ha permitido recrear un comportamiento más real del protocolo de comunicaciones.
Los experimentos realizados por medio de la simulación cumplieron con los escenarios explicados anteriormente y el modelo demostró reproducir la funcionalidad del bus que describe el protocolo CAN.
Esta validación resulta fundamental para continuar con la siguiente etapa en el desarrollo del modelo, especialmente en el diseño y refinamiento de los nodos del sistema, lo que permitirá generar un modelo que simule el comportamiento de los nodos frente a ocurrencias de fallas en los dispositivos. Estos escenarios pueden presentarse en condiciones operativas del contexto aeroespacial.

Referencias

1. IEEE Computer Society. IEEE Standard for Standard SystemC Language Reference Manual. IEEE Std 1666-2011 (2011).
2. Woo Sik Kim; Hyun Ah Kim; Jin-Ho Ahn; Byungin Moon. System-Level Development and Verification of the FlexRay Communication Controller Model Based on SystemC. Second International Conference on Future Generation Communication and Networking, (2008)
3. Pichappan, R.C.; Aziz, S.M. "A Bus Level SystemC Model for Evaluation of Avionics Mission System Data Bus" IEEE Region 10TENCON (2005)
4. Bosch R. GmbH. CAN Specification 2.0 (1991).
5. Encinas D., Frati E., Ramón H., Naiouf M. Caracterización del comportamiento de una red de sensores basado en COTS bajo condiciones de temperatura hostil para sistemas aeroespaciales. XVII Congreso Argentino de Ciencias de la Computación (CACIC). La Plata, Buenos Aires, Argentina. (2011)
6. Ghenassia F. Transaction Level Modeling with SystemC. TLM Concepts and Applications for Embedded Systems. Springer(2005).
7. Black D., Donovan J., Bunton B., Keist A. SystemC: From the Ground Up. Springer (2010).
8. Law, A. M. Simulation Modeling and Analysis, Fourth Edition. s.l. : McGraw-Hill (2006).

9. Burns A., Wellings A. Real-Time Systems and Programming Languages. Fourth Edition. Addison Wesley (2009).
10. Defo G., Kuznik C., Müller W. Verification of a CAN bus model in SystemC with functional coverage. IEEE. International Symposium on Industrial Embedded Systems (SIES). (2010)
11. Encinas D., Alustiza, D., Frati E., Ramón H., Naiouf M. Análisis de tráfico de una red de sensores girométricos del tipo IFOG aplicado a sistemas aeroespaciales. VI Congreso Argentino de Tecnología Espacial, Universidad de La Punta, San Luis, Argentina (2011).
12. Davis R. I., Burns A. Controller Area Network (CAN) schedulability analysis: Refuted, revisited and revised. Real-Time Systems, Springer 35(3):239–272. (2007).