Aplicación práctica de co-diseño de HW y SW para la apertura de un sistema de tiempo real

Christian L. Galasso[1,2,3] Adrián H. Laiuppa[1,3] Joel Ermantraut[2,3] Sergio D. Leoni[3] Diego M. Martínez[2] Martín E. Paz[2]

[1] Universidad de la Defensa Nacional – FADARA – ESOA, Punta Alta, Argentina
[2] Servicio de Análisis Operativos, Armas y Guerra Electrónica de la Armada, Punta Alta, Argentina
[3] Universidad Tecnológica Nacional – FRBB, Bahía Blanca, Argentina
clgalasso@frbb.utn.edu.ar; alaiuppa@frbb.utn.edu.ar; joelermantraut@gmail.com; sergioleoni@frbb.utn.edu.ar; dmmartinez7@gmail.com; pazmartin35@gmail.com

Abstract. En la Armada Argentina, como en otras armadas e industrias conviven sistemas de larga data con sistemas modernos. En el presente escrito se expone el trabajo de co-diseño de un sistema embebido basado en una FPGA y un software de alto nivel que permiten la apertura de una red de tiempo real, para poder llevar a delante el reemplazo de uno de sus componentes. En algunas embarcaciones se pueden encontrar consolas de ayuda a la navegación vinculadas a sistemas de cómputo de la década del 70'. La conexión de estos y otros sistemas es mediante una red con restricciones de tiempo real estricto. Se describe a continuación una posible solución y las pruebas realizadas para validarla.

Keywords: Sistema Embebido, FPGA, Python, Tiempo Real.

1 Introducción

1.1 La vinculación entre sistemas nuevos y viejos

Desde hace más de tres décadas existen en servicio un sinnúmero de líneas de control y accionamientos remotos en una variedad de campos de interés tales como la generación eléctrica, la navegación, y la aeronáutica. La revolución tecnológica ha transformado profundamente la interrelación entre los sistemas digitales, logrando relevamientos más sencillos de las condiciones de funcionamiento, garantizando operaciones cada vez más seguras y amenizando las interfaces humanas. A la luz de estos hechos, es evidente que equipamiento específico, que está operativo y cuyo reemplazo completo es en exceso oneroso, podría encontrar una alternativa de modernización gradual (económicamente más viable) si pudiera desarrollarse una suerte de Gateway que implemente un protocolo que permita vincular tecnologías obsoletas con actuales.

Considerando entonces los casos de interés mencionados, existen numerosos ejemplos de aplicación de sistemas informáticos antiguos, diseñados ad hoc, en que la vida útil de la planta, por su diseño, queda fuertemente vinculada a la vida útil del disposi-

tivo que la controla [1]. La posibilidad, a futuro, de operar estos sistemas depende entonces de la capacidad de adaptarlos, como parte de una estrategia de reingeniería [2], a los nuevos conceptos del estado del arte, los cuales exigen interfaces que puedan interaccionar a distancia, y con compatibilidad multiplataforma [3].

Para el caso específico de este trabajo, se tiene un sistema antiguo de tiempo real duro [4], de arquitectura cerrada, cuyas unidades funcionales están distribuidas en distintos ordenadores en red, interconectados entre sí por varias placas de comunicaciones de tecnología propietaria, formando una topología tipo malla, aunque con un nodo central bien diferenciado. Interesa entonces, generar un enlace full dúplex entre este sistema cerrado y un dispositivo externo, como puede ser una PC comercial, con los objetivos de monitorear de forma externa las variables de estado que se controlan, y también poder realizar pruebas y desarrollar simulaciones, al contarse con la posibilidad, tanto de interpretar como de generar los mensajes de varios periféricos, que se reciben y envían, desde y hacia el ordenador principal.

En el presente trabajo se describe el desarrollo conjunto de un hardware (FPGA) que oficia de Gateway entre sistemas, y el correspondiente desarrollo de un software en alto nivel (Python) que cumple las funciones del subsistema a ser reemplazado/modernizado.

1.2 Sistemas de Tiempo Real

Un STR no siempre se refiere a un sistema de rápida respuesta, si bien una respuesta rápida puede satisfacer las demandas temporales del mismo. Tiempo real se refiere a que las tareas a desarrollar por el sistema tienen un tiempo de inicio y un vencimiento predeterminados, y que incumplir con los mismos tiene alguna consecuencia más o menos catastrófica. Los STR pueden tener restricciones de tiempo estrictas y no estrictas. Cuando el sistema posee el primer tipo, todo incumplimiento de la misma es considerado una falla catastrófica (por ejemplo: la aplicación tardía de refrigerante en un reactor nuclear, o la actuación tardía del sistema de frenado de emergencia automático en vehículos), y se lo denomina “de tiempo real duro”. Por otra parte si el incumplimiento de los límites en los tiempos de inicio o de vencimientos de las tareas no influye significativamente en el proceso se denomina al sistema “de tiempo real blando” (por ejemplo: reproducción de contenido multimedia, o voz sobre IP).

El sistema operativo de las computadoras de a bordo de los barcos donde se pretende aplicar el reemplazo es de tiempo real duro. El propio sistema verifica que se cumplan los temporizados de los mensajes y respuestas, y si uno de los subsistemas que conforman la red distribuida no lo cumple es dado de baja para garantizar la seguridad funcional del conjunto.

1.3 Sistemas Distribuidos

Los Sistemas Distribuidos pueden clasificarse por la forma en que se acoplan los elementos de procesamiento del mismo. Si lo que se tiene para intercambiar información entre las unidades de procesamiento es una memoria compartida (como es el caso de los procesadores multi-núcleo) el sistema distribuido es fuertemente acoplado. Si en

cambio hablamos de un grupo de computadoras personales completas que comparten la información mediante una red (como es el caso de los clúster de computadoras conectadas mediante Ethernet) se dice que el sistema es débilmente acoplado. En todos los casos lo que se pretende es que el conjunto de computadoras o procesadores, actúe como un servicio integrado destinado a un fin específico. Dichos sistemas están equipados con un software de sistemas distribuidos, el cual permite que las computadoras coordinen sus actividades y compartan recursos.

El sistema que se aborda en el presente trabajo se define, más apropiadamente, como un conjunto de computadoras autónomas conectadas por una red, y con el software distribuido adecuado para que el mismo sea visto por los usuarios como una única entidad capaz de proporcionar facilidades de procesamiento.

1.4 Un antecedente

Habiendo estudiado la red de computadoras utilizadas para operaciones navales y navegación de un barco, se procedió a realizar un análisis de la factibilidad de reemplazar uno de sus nodos, por una aplicación emulador del mismo que se ejecutara sobre un sistema operativo de tiempo diferido, sobre una notebook. Se observó que, por su antigüedad y su empleo actual, sería imposible modificar el software o hardware de las computadoras originales que componen el sistema para integrar de manera nativa un nuevo enlace de comunicaciones externo. Sin embargo, se dispone del puerto de comunicación nativo con un protocolo propietario, por donde el ordenador central intercambiaba información con el nodo. En un trabajo anterior [5], miembros del equipo de investigación realizaron con éxito la re-ingeniería de la placa de comunicaciones original del sistema mediante dispositivos lógicos programables (También conocidos como FPGA) y a posteriori desarrollaron un sistema embebido [6] que emula el handshake del mismo. De esta forma, se decidió desconectar el dispositivo que interactuaba con el ordenador central, para reemplazarlo por un dispositivo compuesto por un sistema embebido programable, micro controlado con un Córtex M4. Este dispositivo, llamado banco de pruebas, posee una interfaz compatible con las placas de comunicación, pudiendo enviar y recibir datos por medio de éstas. La conexión realizada se esquematiza en la figura 1.

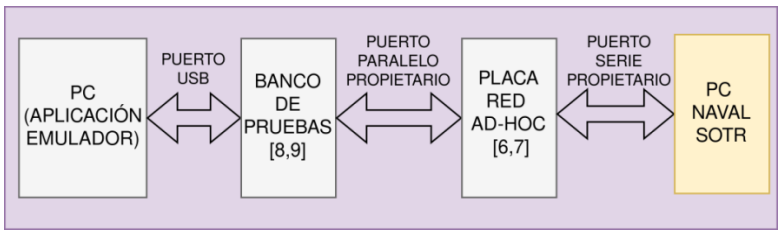

Fig. 1. Esquema de interconexión realizado para el reemplazo de uno de los nodos de un sistema distribuido de tiempo real. SOTR: Sistema Operativo de Tiempo Real.

La comunicación al ordenador central posee una estructura y semántica que debe respetar el periférico que se encuentre conectado al mismo, a fin de establecer una

comunicación confiable a lo largo del tiempo. Esta comunicación es por medio de tramas de entrada y salida de longitud fija. Respecto al temporizado, el sistema de tiempo real exige una transacción entrada/salida cada 500 ms.

Por otro lado, el banco de pruebas posee una interfaz USB, que permite la conexión con el dispositivo externo a un ordenador. En este caso, se seleccionó como dispositivo final una PC comercial, que ejecuta un software de generación y monitoreo de paquetes de la red cerrada.

Con el objetivo de que el intercambio de información sobre ambos extremos sea transparente, el protocolo junto con el algoritmo de enrutamiento de los datos, se diseñó en torno al firmware del sistema embebido, confiriéndole al mismo la funcionalidad del Gateway [7], debido a su capacidad de operar interconectando dos canales de comunicación distintos. De esta forma, por un lado, la red de ordenadores interactúa naturalmente con nuestro dispositivo, debido a que se respeta la estructura de la comunicación que se tenía con el periférico anterior. En el extremo opuesto, se establece entre la PC y nuestro dispositivo, una comunicación asincrónica, libre de vencimientos, y con una trama de datos de entrada y una de salida definidas.

Este sistema fue probado en laboratorio y a bordo, verificándose el correcto funcionamiento y dejando una base de conocimiento para ampliar la solución alcanzada a otros nodos.

2 Proyección conceptual de la solución alcanzada

2.1 Elección de un nuevo nodo que pueda reemplazarse

Sobre la base de conocimiento y la solución lograda para el sistema particular presentado, se intentó extender el concepto a un subsistema diferente.

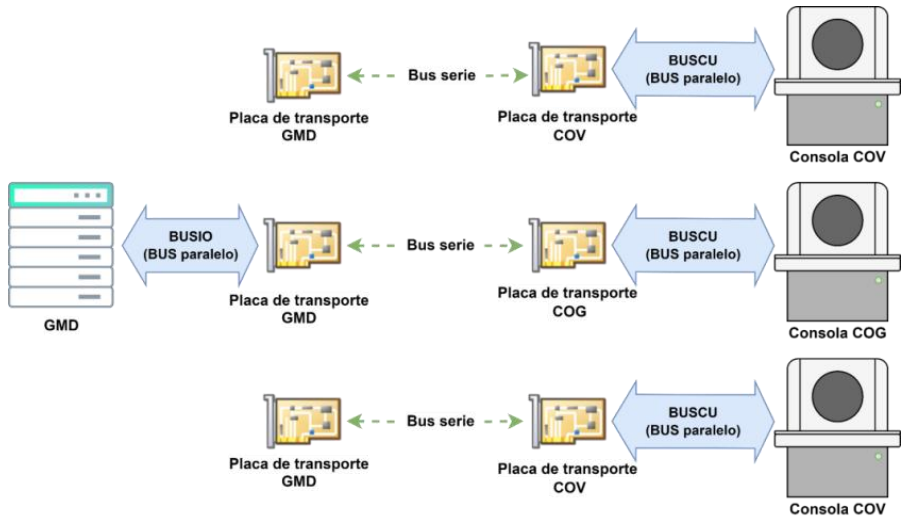

Fig. 2. Esquema de conexión de la computadora naval central y sus consolas.

En varios barcos se puede encontrar una computadora central (En adelante GMD) vinculada a tres consolas (COV: Consola de Operaciones Vertical y COG: Consola de Operaciones General) como puede verse en la figura 2. Ambas consolas cuentan con teclados y dispositivos de apuntamiento para enviar comandos al GMD y poder así realizar un gran número de operaciones. Están provistas, además, de dos pantallas que permiten visualizar información gráfica y alfanumérica. La comunicación entre GMD y consola es bidireccional y se caracteriza por tener cinco mensajes predefinidos. Sobre el GMD se ejecutan aplicaciones que corren sobre un sistema operativo de tiempo real. En la consola no se ejecuta ningún software, pero tiene suficiente electrónica para conformar varias decenas de máquinas de estado. Esta forma constructiva hace a la consola bastante determinística en los tiempos que requiere para realizar cada operación. Por ende, y según la documentación a la que se pudo acceder, no solo están definidas a nivel de bit las comunicaciones, sino que también están establecidos tanto los tiempos de vida máximos de los mensajes, como los de espera para recibir la respuesta a determinados mensajes

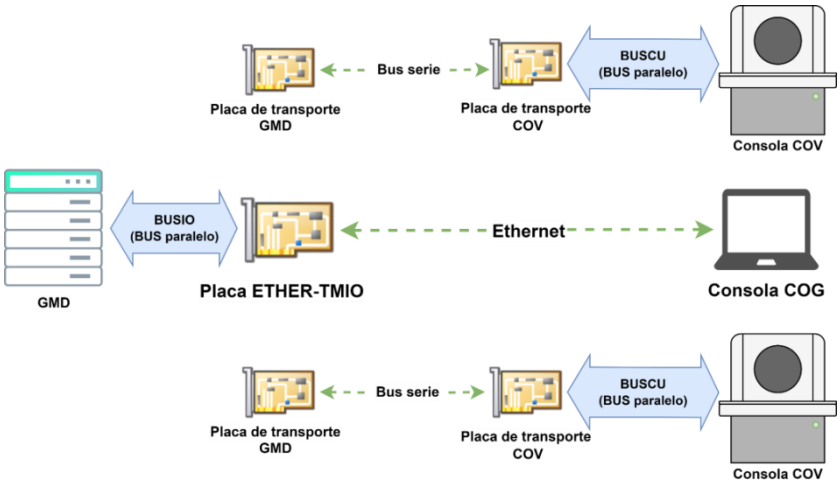

Fig. 3. Nuevo sistema propuesto para el reemplazo de uno de los nodos de la red de tiempo real.

Se propuso el cambio de hardware que puede verse en la figura 2, con el correspondiente desarrollo de software de alto nivel (Python). Se eligió la consola COG por encima de la COV por ser esta última una versión reducida de la primera, por ende al desarrollarse la COG, el desarrollo de la COV sería una reducción o simplificación del desarrollo previo.

2.2 Planteo de un nuevo Gateway entre dominios temporales

Partiendo de la premisa que en lugar de la consola original, se pretende utilizar una PC convencional, se estableció que la comunicación de la PC con el dispositivo que resuelve el tiempo real va a ser Ethernet. De esta forma la “placa de transporte COV o

COG" es absorbida por la placa de red de la propia computadora y no necesitan desarrollarse dos placas, sino solo una, que pueda dialogar con el GMD por un lado y con un puerto Ethernet por el otro. Cabe destacar que el cableado original del barco era apto para montar un puerto Ethernet y fue ensayado arrojando resultados de 88 Mbits/s.

Se optó por proyectar una solución de Gateway enteramente en hardware, por lo que se utilizó una FPGA para la placa que se denominó ETHER-TMIO, en referencia a que realizaba una comunicación Ethernet con el módulo de transporte del bus denominado I/O de la computadora naval.

El diseño digital de la placa ETHER-TMIO se dividió en diferentes partes. Cada una cumple una función esencial dentro del sistema. Los dos grandes grupos que pueden distinguirse son: la comunicación entre la placa y la PC, y la comunicación entre la placa y el servidor central del buque GMD. La primera de ellas se encarga de transmitir y recibir mensajes UDP. La segunda cumple la función de comunicarse por puerto paralelo con el buque. Ambas comunicaciones poseen restricciones temporales diferentes, ya que el buque posee requerimientos de tiempo real mientras que la PC utiliza un sistema operativo de tiempo diferido. Para sincronizar ambos sistemas se describió un componente general que en el Figura 4 se denomina "Entidad superior main".

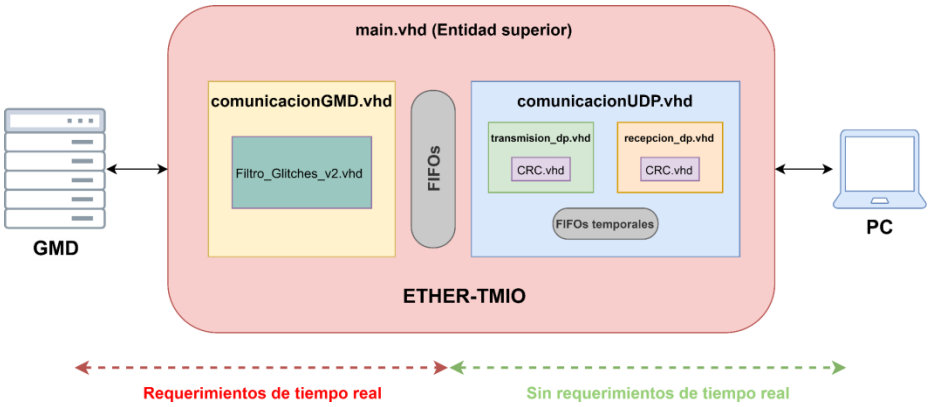

Fig. 4. Diagrama general del diseño digital dentro de la FPGA

3 Consideraciones del diseño digital

3.1 Entidad superior main

La entidad superior "main" es la encargada de sincronizar todos los componentes del código. En ella se describe un modelo estructural en el que se interconectan diferentes componentes a través de señales que actúan como cables de interconexión. Se instancian los componentes "comunicacionUDP", "comunicacionGMD" y FIFOs correspondientes a los mensajes predefinidos entre el GMD y la consola.

A continuación se describe el funcionamiento del sistema. En primer lugar, se reciben mensajes del GMD a través del componente comunicacionGMD, los mensajes destinados a las pantallas (Que se referencian como PANTALLA15P y PANTALLA07P) se almacenan en memorias FIFOs y los mensajes que no requieren ser transmitidos a la PC se responden de forma automática con una respuesta predefinida (Por ejemplo el mensaje de keep alive [8] del sistema). Luego, los mensajes almacenados son transmitidos hacia la PC por puerto Ethernet, esta acción es realizada a través del componente comunicacionUDP. La entidad superior también lee los mensajes recibidos de la PC a través de comunicacionUDP y los almacena en la memoria "fifo_CONCENTRATOR" que luego será leída por el componente comunicacionGMD para transmitir el mensaje hacia el GMD cuando este lo solicite. Por lo tanto, las memorias FIFOs instanciadas en main conforman una memoria compartida entre ambas comunicaciones. Cuando se recibe el mensaje CONCENTRATOR de la PC se escribe su FIFO y cuando se requiere enviar mensajes al GMD se lee. De igual forma, cuando se reciben mensajes PANTALLA15P y PANTALLA07P del GMD se escriben las FIFOs correspondientes y se leen desde la PC.

Los mensajes recibidos por parte de la PC o GMD deben guardarse en la placa hasta que sean enviados y se haya recibido el acuse de recibo o ACK ("Acknowledgement" o confirmación de lectura). La forma elegida para guardar la información dentro de la FPGA es por medio de "FIFOs", estructuras de datos que ya están implementadas.

3.2 Elección de protocolo de comunicación: UDP o TCP s

El protocolo TCP ("Transmission Control Protocol" o protocolo de control de transmisión) es un protocolo de transporte orientado a la conexión, posee detección y corrección de errores, control de gestión y confirmación de recepción. Requiere de un establecimiento de conexión previo al inicio de la comunicación. Esta conexión previa garantiza la conexión efectiva de ambos lados. Incluye dentro de los campos de la trama el número de secuencia y la confirmación de recibo. Sin embargo, los mecanismos mencionados tienen como penalización mayor carga de protocolo en los mensajes, mayores latencias y mayor complejidad de implementación.

El protocolo UDP carece de muchas de las características del protocolo TCP. A diferencia del protocolo TCP, en donde los paquetes se envían en forma de flujo, en UDP cada paquete es independiente. La fiabilidad del protocolo es menor porque no se incluye confirmación de lectura y no permite control de gestión. No obstante la velocidad de transferencia es mayor y la complejidad en su implementación es menor.

Para la elección del protocolo de comunicación que se implementa en la FPGA se tuvieron en cuenta las características descritas anteriormente. Muchas de las funcionalidades de TCP no son de utilidad en el sistema e incrementan considerablemente la complejidad del código HDL. Sin embargo, el sistema debe, por requerimiento, contar con una confirmación de lectura de mensajes, ya que forma parte de las características esenciales de la comunicación en los sistemas de cómputo navales. Por esta razón, se optó por implementar una comunicación UDP con ACK. Es decir, se transmiten y reciben mensajes UDP, pero dentro del campo DATA o carga útil se colocan campos

que contienen el número de secuencia y la confirmación de lectura. De esta manera se agregan características al protocolo de comunicación UDP para incrementar su fiabilidad sin complejizar en exceso su implementación.

3.3 Protocolo de control de flujo RQ inactiva

El método utilizado para el envío de MSJ/ACK entre la PC y la placa ETHER-TMIO se denomina “RQ inactiva”. Se eligió debido a su simplicidad, ya que los sistemas más complejos requerirían complejizar el código VHDL, utilizar un “Softcore" o componentes de librería en la FPGA. El siguiente diagrama muestra el funcionamiento del protocolo de control de flujo RQ (“Request”) inactiva y sus tiempos relacionados.

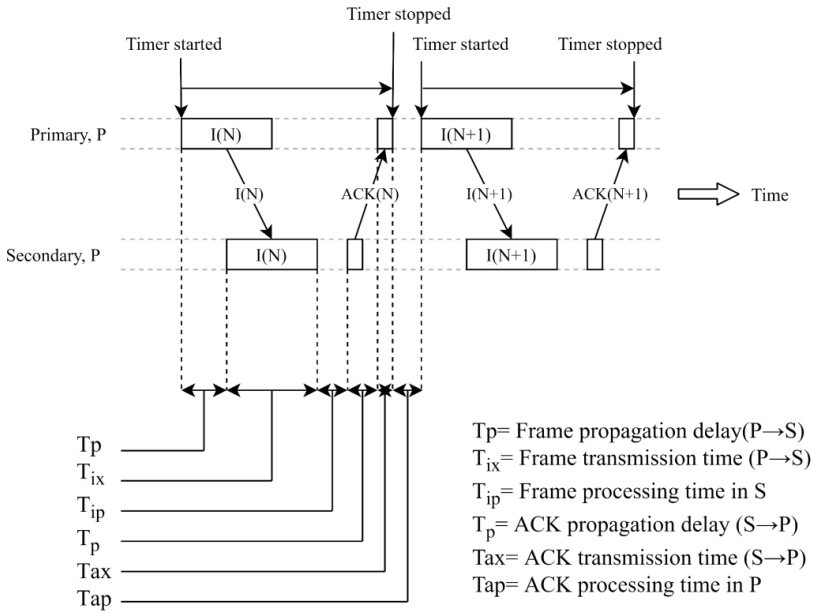

Fig. 5. Comunicación UDP mediante RQ inactiva

Una característica de este tipo de comunicación es que el emisor del mensaje debe esperar la recepción de un ACK por parte del receptor para poder enviar un mensaje nuevo. Esta simplicidad en la implementación conlleva pérdidas temporales. Existen métodos más complejos para solucionar este problema. Sin embargo, RQ inactiva permite cumplir los requerimientos temporales para este caso.

El sistema en cuestión no requiere la transmisión de múltiples mensajes de la misma clase, sino que se espera la confirmación de recibo para que otro mensaje del mismo tipo sea emitido. De esta manera se evita la creación de distintos temporizadores de vencimiento, buffers elásticos con los mensajes enviados, entre otros.

El rendimiento del canal utilizado en este protocolo solo es una fracción del ideal. En el caso que no sea suficiente para el volumen de datos manejado se debe evolucionar a protocolos de ventana deslizante como lo pueden ser de rechazo simple (Go-Back-N) o de repetición selectiva (Selective Repeat).

4 Descripción de la aplicación de software

4.1 Organización del proyecto y clases

Como introducción a la organización del software y qué función cumple cada clase, primero se debe comprender cómo se divide una COG, y cómo se implementa en la aplicación. Como se puede ver en la figura 6, cada consola está compuesta por un PANTALLA15P, como se ve en la parte central de la figura, una o dos PANTALLAS07P (que se muestra en la figura 7) y varios botones que permiten al usuario comunicarse con el GMD.

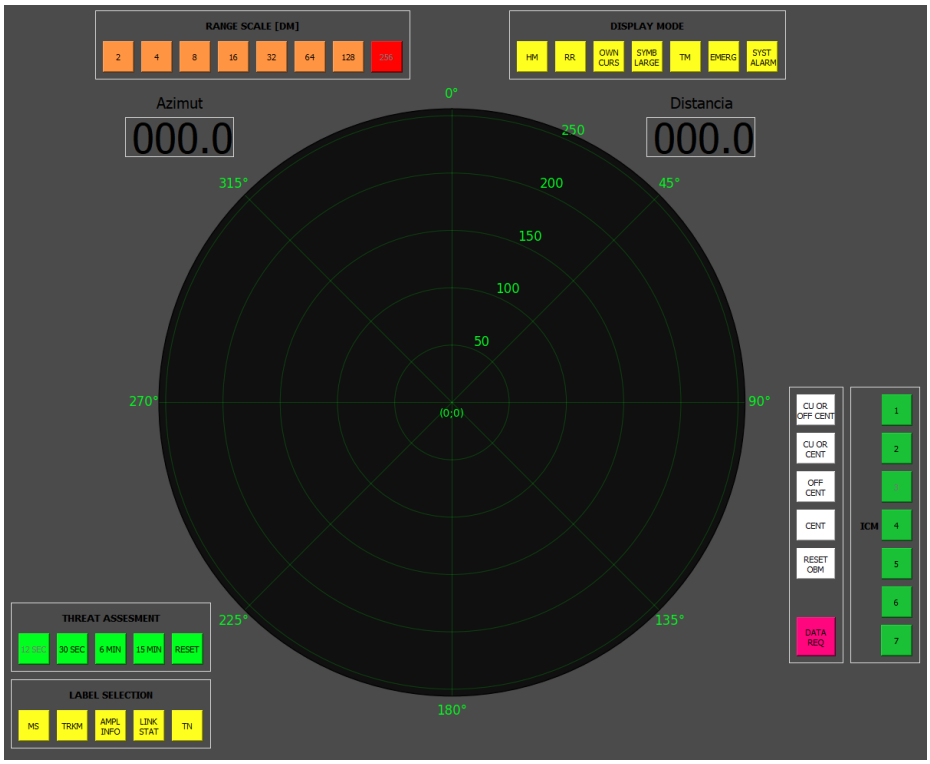

Fig. 6. Representación de la PANTALLA15P y los botones de la COG.

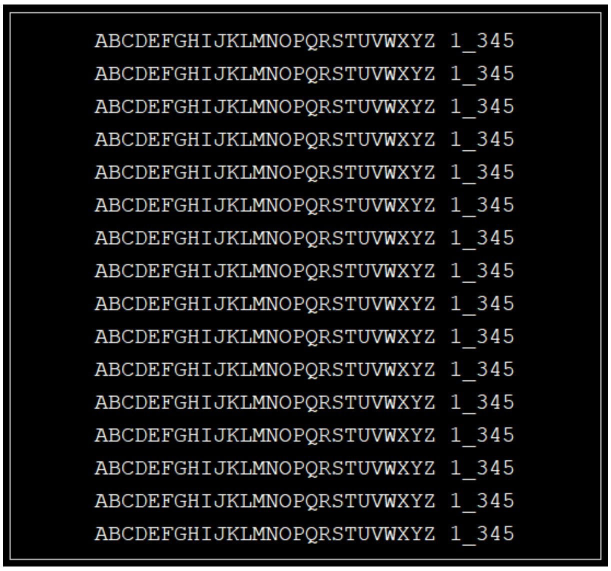

Fig. 7. Representación de la PANTALLA07P

En la figura 8 se muestran las distintas clases que conforman el programa y cómo se relacionan entre ellas. Se ha asignado un color específico a cada una.

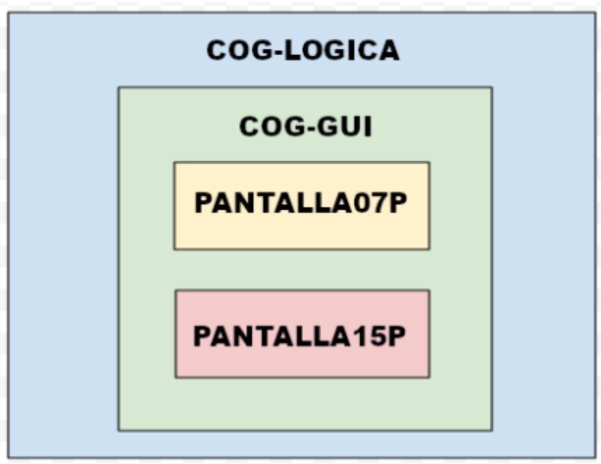

Fig. 8. Diagrama general de interrelación de las distintas clases que conforman el programa que reemplaza la COG.

clase COG-LOGICA:

Es la clase principal, encargada de la comunicación con la FPGA. Contiene una instancia de la clase COG-GUI y es donde se configuran los hilos que se utilizan en el

programa. Es la encargada de armar los mensajes que se deben mandar al GMD y de clasificar los que recibe.

clase COG-GUI:

Es la clase encargada de la parte gráfica, aquí se encuentran todos los métodos necesarios para la comunicación entre la PC y el usuario. Contiene el archivo ".ui" donde se encuentra toda la interfaz: los botones, las dos PANTALLAS07P y la PANTALLA15P.

clase PANTALLA07P:

Es la clase que permite instanciar una o dos PANTALLAS07P. Su función es simular las pantallas de 7 pulgadas de los barcos que sirven para mostrar información alfanumérica de las páginas del sistema operativo del GMD (similar al CMD o al TERMINAL).

clase PANTALLA15P:

En esta clase se encuentra el desarrollo de la PANTALLA15P de la aplicación. Esta es la pantalla principal de la consola y es la que muestra la mayor cantidad de información gráfica.

5 Ensayos realizados

Se realizaron varios ensayos y pruebas, tanto por partes como de forma integral. A los fines del presente escrito se presentan dos que los autores creemos representativas del funcionamiento y las limitaciones encontradas.

5.1 Capturas mediante analizador lógico

Para verificar el correcto funcionamiento del sistema desarrollado, se comprobó que las señales enviadas hacia la computadora naval correspondiesen en tiempo y forma con las de la placa original. Con este fin, a través de un analizador lógico, se realizaron capturas de múltiples señales en rangos temporales configurables. Su posterior análisis permitió entender el funcionamiento del sistema original, detectar fallas y depurar el diseño digital desarrollado.

Los tiempos de respuesta que debía cumplir la FPGA en la comunicación con el buque fueron determinados, en un principio, analizando capturas del sistema original. Se respetaron los tiempos de respuesta promedio. Durante pruebas a bordo se verificó la ausencia de errores y se comprobó el cumplimiento de los tiempos definidos.

A continuación se muestran capturas lógicas con los tiempos de respuesta de la ETHER-TMIO a los mensajes recibidos por el GMD, figura 9, y se comparan con los de la placa original, figura 10.

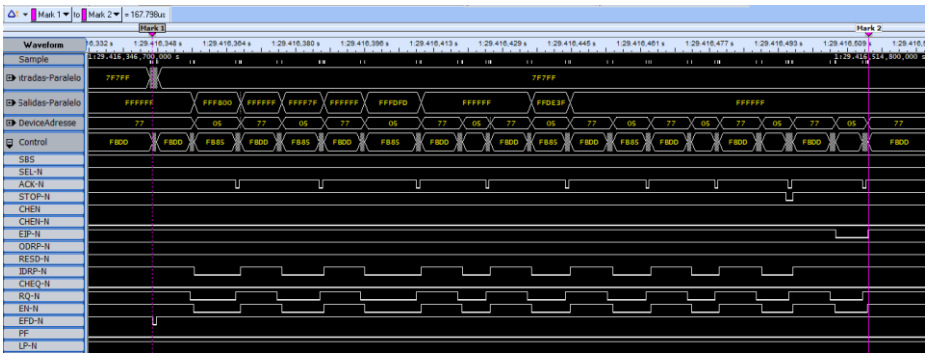

Fig. 9. Captura de analizador lógico de placa ETHER-TMIO. Se registra el tiempo entre cursores: 167,586 us.

Se puede observar una gran similitud entre las capturas, lo que demuestra que la descripción de la FSM concuerda con el comportamiento original de las terminales navales. Además el hecho de que el sistema estuviera en operación durante horas a bordo y que el GMD en ningún momento indicara falla o baja del sistema reemplazado es también una confirmación del correcto temporizado y handshake.

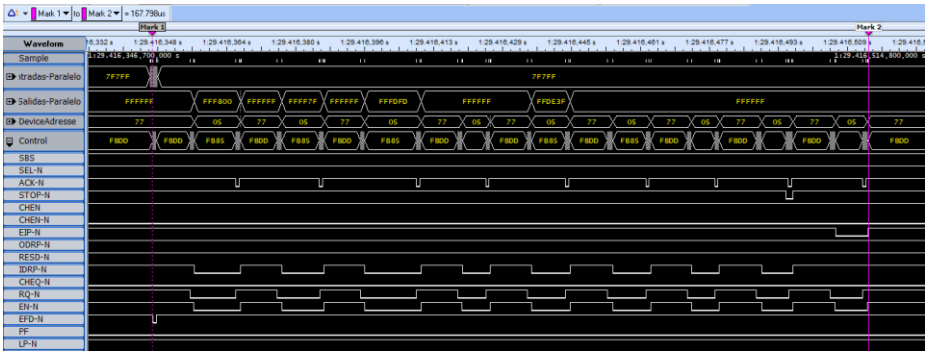

Fig. 10. Captura de analizador lógico de placa de transporte GMD original. Se registra el tiempo entre cursores: 167,798 us.

5.2 Capturas mediante Wireshark

Una de las herramientas principales para la depuración del proyecto consistió en realizar capturas del tráfico de red utilizando el software Wireshark, para posteriormente analizarlas y detectar posibles fallas. Se prestó especial atención al tiempo de respuesta de los ACK enviados tanto por la FPGA como por la PC. Debido a la utilización de máquinas de estado para la recepción y transmisión de mensajes, el tiempo de respuesta por parte de la FPGA es extremadamente breve.

En primer lugar, se analiza el tiempo de respuesta de la FPGA ante los mensajes recibidos desde la PC. Pueden observarse tiempos de aproximadamente 600 µs entre la recepción de un mensaje y la transmisión de una confirmación ACK por parte de la

FPGA. Este tiempo engloba la recepción de un mensaje desde la FPGA, la transmisión de una respuesta hacia la PC, y la recepción junto al procesamiento en la misma para mostrar su contenido en Wireshark.

El tiempo de respuesta de la PC ante mensajes recibidos desde la FPGA varió a medida que la aplicación en la PC se fue optimizando. Las mejoras se realizaron ya que el método utilizado (RQ inactiva) necesita de la recepción del ACK para enviar el posterior mensaje, por lo que el tiempo de respuesta se debe mantener lo suficientemente bajo para no producir un cuello de botella y evitar que las FIFOs de la FPGA desborden. En un comienzo se obtuvieron tiempos mayores a 100 ms que producían el desborde de la aplicación a los pocos segundos. Actualmente el tiempo de respuesta se mantiene inferior a los 10 ms. Cabe destacar que los tiempos son muy variables ya que dependen de muchos factores como el uso de la PC al momento de recibir el mensaje, el sistema operativo, optimización de la aplicación, cantidad de información que la computadora naval envíe hacia la PC para su representación, entre otros.

6 Conclusiones

En base a los resultados obtenidos mediante el analizador lógico y la evidencia de que la computadora del GMD no dé de baja el prototipo en períodos extensos de uso (mayores a las 24Hs), se puede afirmar que el diseño digital está maduro.

Por el lado del software, se evalúan distintas versiones de despachadores y/o asignación de prioridades que pudieran usarse para aumentar la prioridad de la aplicación.

Para reducir aún más los tiempos, en futuras versiones del software se realizará el procesamiento gráfico de la PANTALLA15P a través de la placa de video, ya que de lo contrario se consumen muchos recursos del microprocesador.

Se verificó que ante la recarga de figuras dentro de la PANTALLA15P, se aumenta la acumulación de mensajes que no reciben su ACK. Verificado el administrador de tareas o el monitor de recursos, se encuentra que el causante de la saturación del procesador es la máquina virtual de Phyton, por ende consideramos propicio continuar el desarrollo cambiando de lenguaje.

Mejorando el rendimiento de la aplicación puede llegar a ser una interesante solución para el reemplazo gradual de los componentes del sistema.

Agradecimientos. Los autores agradecen el invaluable aporte que realizaron para este desarrollo los alumnos Marcos Loidi, Joaquin Lutri, y Manuel Fernández en el marco de su proyecto final de Carrera. Así mismo las contribuciones de los alumnos: María Baza, Leandro Ferrari y Emiliano Martínez, quienes participaron desde su práctica profesional supervisada. También el financiamiento recibido desde la UNDEF por medio de su programa UNDEFI, y desde el Ministerio de Defensa por medio del programa PIDDEF.

Divulgación de intereses. Este trabajo se realizó con aportes del Ministerio de Defensa y de la Universidad de la Defensa Nacional (UNDEF). Las unidades y laboratorios, así como el instrumental utilizado son propiedad de la Armada Argentina.

Referencias

1. Ejemplo de un sistema informático de tiempo real que opera sin modificaciones durante todo el ciclo de vida de una planta. Nuclear plant powers up on real-time OS. Disponible en: http://www.itbusiness.ca/news/nuclear-plant-powers-up-on-real-time-os/9084.
2. Manejo de la obsolescencia tecnológica. Suresh K. Nair. A model for equipment replacement due to technological obsolescence. European Journal of Operational Research 63 (1992) 207-221 Disponible en: https://www.researchgate.net/profile/Wallace_Hopp/publication/4941721_A_Model_for_Equipment_Replacement_Due_to_Technological_Obsolescence/links/57f3e72e08ae886b897dccad.pdf
3. Ejemplo de pérdida de funcionalidad y adaptabilidad de una planta debido a la antigüedad de los sistemas informáticos vinculados. Ver: Re-programming "Little Boy". Adelanto sobre la reestructuración de Ferrania. Disponible en: http://www.filmferrania.it/news-articles/2017/welcome-to-2017
4. Categories of real time systems. Giorgio C. Buttazzo. Hard Real-Time Computing Systems. Disponible para vista previa en:https://books.google.com.ar/books?id=h6q-e4Q_rzgC&printsec=frontcover&hl=es
5. Desarrollo de un prototipo basado en FPGA. Galasso Ch. L.; Friedrich G. R.; Antonini A. A.; Díaz G. J. IV Congreso Microelectrónica Aplicada, UEA 2013, Memorias del Congreso. Friedrich G., Reggiani G., Coppo R., Baldini P., Iparraguirre J., Pellegrino S., Cayssials R (Editores); pp. 59 - 64. ISBN 978-987-1896-18-9.
6. Plataforma de pruebas para interfaces de red en tiempo real basado en un sistema embebido. Franco S. Caspe, Emmanuel Pita, Miguel A. Banchieri, Christian L. Galasso. Congreso Argentino de Sistemas Embebidos, CASE 2016, Libro de Trabajos, Modalidades Foro Tecnológico y Póster. Brengi D., De Micco L., Lipovetzky J., Lutenberg A., García Inza M., Antonelli M. (Editores); pp. 65. ISBN 978-987-45523-8-9.
7. Definición de gateway. Telecommunications: Glossary of Telecommunications Terms. Editado por la National Telecommunication Information. 1997
8. Keepalive Overview. Funciones de los paquetes de tipo "keepalive". The Linux Documentation Project. Disponible en: http://tldp.org/HOWTO/TCP-Keepalive-HOWTO/overview