An Approach for Side Scan Sonar Acoustic Images Segmentation Using Programmable Logic

S. A. Villar, *Member, IEEE,* S. R. Rossi, *Member, IEEE* and G. G. Acosta, *Senior Member, IEEE*

***Abstract*— In applications related to maintenance of underwater infrastructure in harbour installations, cables and pipelines, it is necessary to separate the relevant objects of study from the rest of the acoustic image obtained by sonar. Current acoustic image processing techniques require a high computational effort and time consumption due to the manipulation of large volumes of data generated by these devices. Motivated by this problem, in this paper we present an alternative for segmenting acoustic images, proposing a framework for use with programmable logic technology. We propose to perform data segmentation into three types of regions, acoustic highlight, seafloor reverberation and acoustic shadow areas, by using programmable logic and VHDL hardware description language. Finally, results obtained over seafloor images with pipeline presence and acquired with a side scan sonar from a vessel in Salvador de Bahía, Brazil are presented.**

***Keywords*— Side Scan Sonar, Acoustic Image, Programmable Logic, Hardware Description.**

I. Introducción

LAS TAREAS en acústica submarina enfocadas en las estructuras planificadas [1] a profundidades cada vez mayores, representan grandes desafíos desde el punto de vista tecnológico [2]. Se consideran en esta categoría el mantenimiento de infraestructuras sumergidas como tuberías y cables, la búsqueda de recursos, la extracciones y perforación de petróleo, la detección de minas, el estudios de sitios de descarga, de puertos y accesos a embarcaciones, de arqueología de naufragios, entre otros [1], [3]–[6].

En este sentido, los sistemas de sonar proporcionan casi una fotografía de las áreas subacuáticas, en imágenes de alta resolución, incluso en aguas con escasa o nula visibilidad para la visión humana [7].

Las ecosondas multihaz (MBE – *Multibeam Echosounders*) y los sonares son dispositivos ampliamente empleados para la exploración del lecho marino [8]. El sonar de barrido lateral (SSS – *Side Scan Sonar*) constituye uno de los tipos de sonares de escaneo lateral más difundidos, constituyendo la principal herramienta de visualización para la obtención de imágenes acústicas del suelo marino [6]. Dichos dispositivos han sido probados en aguas profundas con resultados muy satisfactorios [9]–[11].

S. A. Villar, INTELYMEC, CIFICEN, CONICET, Fac. de Ingeniería. Universidad Nacional del Centro Prov. de Buenos Aires (UNCPBA), Argentina, svillar@fio.unicen.edu.ar.

S. R. Rossi, INTELYMEC, CIFICEN, CONICET, Fac. de Ingeniería. Universidad Nacional del Centro Prov. de Buenos Aires (UNCPBA), Argentina, srossi@fio.unicen.edu.ar.

G. G. Acosta, INTELYMEC, CIFICEN, CONICET, Fac. de Ingeniería. Universidad Nacional del Centro Prov. de Buenos Aires (UNCPBA), Argentina, gerardo.acosta@ieee.org.

En relación a la segmentación de imágenes de sonar de alta resolución se deben identificar tres tipos de regiones [12], [13]: resalte acústico, sombra y reverberación del suelo marino. Las áreas de resalte acústico se originan a partir de la reflexión de onda acústica del objeto, mientras que las zonas de sombra se forman por una falta de reverberación acústica detrás del objeto. Las áreas restantes consisten en la reverberación del fondo marino.

Diversos enfoques de técnicas de segmentación para el procesamiento de imágenes acústicas en entornos de software están actualmente disponibles y varían en su velocidad, eficacia, necesidades de recursos, precisión y robustez. Muchas de las técnicas de segmentación se migraron de la teoría de procesamiento digital de imágenes [14], [15]. También existen desarrollos relacionados con la segmentación de imágenes, mediante el empleo de VHDL y su implementación en dispositivos lógicos programables, para diferentes áreas de aplicación [16]–[18], no obstante, el presente trabajo pretende reflejar su viabilidad de implementación, tomando como caso la segmentación de imágenes acústicas provenientes de SSS, para su potencial aplicación en vehículos submarinos autónomos para el seguimiento de cables y tuberías submarinas.

Se presenta el diseño de un sistema basado en la combinación de software desarrollado en C++, con un conjunto de bloques sintetizables para su implementación en hardware y bloques no sintetizables para finalidades de prueba y simulación, ambos descritos en lenguaje VHDL (*Very High Speed Integrated Circuit – Hardware Description Language*). Se realiza la segmentación en tres tipos de regiones: zonas de resalte acústico, reverberación del fondo marino y zonas de carencia acústica.

El artículo está organizado de la siguiente manera: en la Sección II se realiza una breve descripción del principio de funcionamiento del SSS; en la Sección III se presenta la metodología completa de descripción del sistema y en la Sección IV se muestran los resultados obtenidos. Finalmente se exponen las conclusiones del trabajo.

II. Sonar de Barrido Lateral

El SSS está formado por un conjunto de transductores que, en cada ciclo de adquisición de datos, escanean hacia los lados y hacia abajo, constituyendo un plano vertical que avanza en el sentido de la orientación del vehículo. La Fig. 1 muestra una representación idealizada de la operación de un SSS montado en un vehículo en movimiento.

Los transductores de ambos lados del sonar envían señales acústicas oblicuas, en forma de abanico, con pulsos acústicos estrechos hacia los planos perpendiculares de la dirección del

movimiento. El rango de operación de estos pulsos acústicos normalmente oscila entre 100 y 500 kHz. El lado izquierdo babor y derecho estribor de las imágenes se escanean por separado.

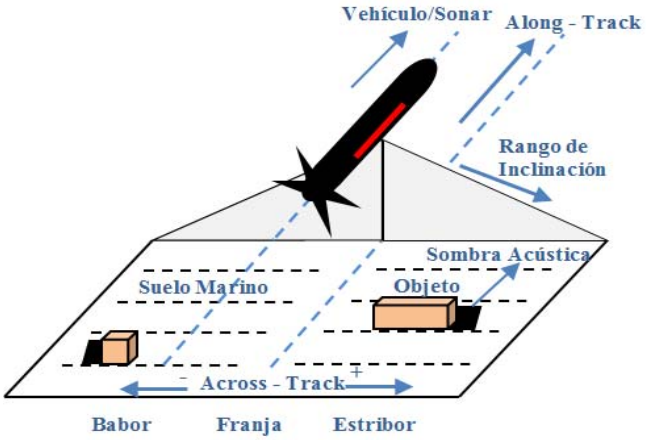

Figura 1. Operación del sonar de barrido lateral [19].

Los pulsos acústicos viajan a lo largo de la columna de agua, golpean el fondo del mar y el eco retorna al sensor de recepción donde se cuantifica su amplitud. Esta amplitud depende del rango de inclinación, del ángulo de incidencia y del material constituyente del fondo marino. Los objetos que se encuentran sobre el suelo marino generan sombras o carencia acústica útil para obtener su detección.

Los ecos procedentes directamente del suelo marino constituyen la señal. También hay múltiples rebotes del fondo del mar o de la superficie del mar que constituyen la reverberación o ecos no deseados. Las regiones debajo del sonar y sobre el sonar corresponden a puntos de baja y alta reflexión de la superficie del fondo marino y de la superficie del mar, respectivamente.

Los datos adquiridos son proyectados sobre una línea trazada a lo largo del lecho marino que representa una fila de una imagen acústica. Si el vehículo se está moviendo en una línea recta y a una velocidad uniforme, la imagen acústica de SSS se construye colocando una secuencia de líneas consecutivas a lo largo de la trayectoria de navegación para crear una representación del lecho marino en 2-D [20].

Los SSS convencionales proporcionan líneas de pulsos acústicos que varían desde 200 a 8200 muestras. Vale decir que a mayor cantidad de muestras implica un mayor desempeño computacional. La capacidad de segmentar imágenes acústicas de alta resolución de manera eficiente es fundamental para cualquier aplicación práctica de las ya mencionadas.

III. Materiales y Métodos

A. *Lineamientos generales*

La segmentación de imágenes acústicas provistas por SSS se fundamenta en que los objetos que se encuentran sobre el fondo marino suelen ser más reflexivos que el sedimento circundante. Por este motivo, una de las alternativas de detección se centra en encontrar las intensidades máximas o resalte acústico, que varía considerablemente según la orientación relativa del sonar hacia el objeto, pudiendo caer por debajo del umbral de detección originando que el objeto sea invisible [21]. Además, una característica adicional relevante de este tipo de imágenes acústicas es que los objetos que sobresalen por encima del suelo marino generan sombras, es decir, áreas donde la intensidad del eco es frecuentemente menor que el nivel procedente del fondo marino. La longitud de la sombra depende de la altura vertical del objeto. Así, otra alternativa de detección consiste en la utilización de las sombras [22]. En virtud de que la adquisición se realiza desde un vehículo en movimiento, la geometría del sonar con respecto al objetivo es variable, pudiendo una sombra estar presente incluso cuando el resalte acústico no lo está. Por esta razón, resulta deseable combinar ambas aproximaciones.

En general los algoritmos para segmentación se clasifican como supervisados y no supervisados. Los algoritmos supervisados utilizan un clasificador entrenado para segmentar en regiones [9], [23]–[25]. Los no supervisados realizan la segmentación del fondo marino, haciendo un análisis directo de la imagen de entrada sin ningún tipo de información a priori, como es el caso analizado en este trabajo.

B. *Propuesta para la segmentación de imágenes acústicas de SSS*

Se diseñó una metodología basada en la combinación de software desarrollado en C++, con un conjunto de bloques sintetizables para su implementación en hardware y bloques no sintetizables para finalidades de prueba y simulación, ambos descritos en lenguaje VHDL.

La elección de VHDL radica en su amplia aceptación a nivel mundial, por ser estandarizado y permitir la síntesis de hardware en dispositivos lógicos programables de diferentes fabricantes [26]–[28].

En la Fig. 2 se ilustra la metodología propuesta, la cual consiste de tres fases definidas: *sistema de conversión*, *arquitectura de pruebas* y *síntesis*, las cuales se detallarán en las secciones siguientes. En dicha figura se aprecian las etapas de procesamiento dentro de cada fase. Estas etapas son las siguientes:

- *Etapa 1*: adquisición/lectura de la imagen acústica.
- *Etapa 2*: conversión de datos acústicos a un lenguaje común para la arquitectura de pruebas.
- *Etapa 3 y 4*: lectura de datos desde un archivo de entrada y escritura a memoria RAM del sistema (*Read File*).
- *Etapa 5*: procesado de datos de memoria RAM y lógica de segmentación de datos acústicos (*Process Memory*).
- *Etapa 6 y 7*: lectura de datos de memoria RAM y escritura a un archivo de salida (*Write File*).
- *Etapa 8*: conversión de datos de un archivo de texto a una imagen acústica.
- *Etapa 9*: salida y verificación del resultado obtenido con el implementado totalmente en software.
- *Etapa 10*: la técnica de segmentación verificada se sintetiza en un dispositivo lógico programable empleando lenguaje VHDL.

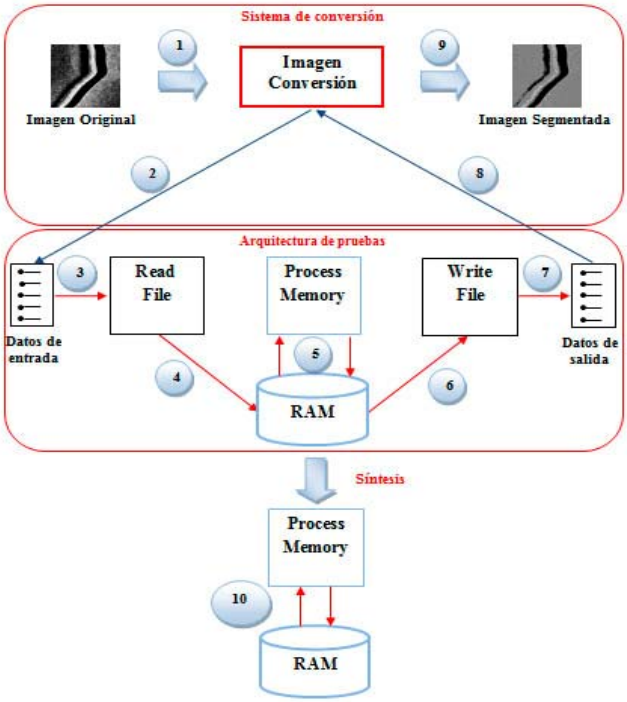

Figura 2. Metodología propuesta para la segmentación de imágenes acústicas.

Sistema de conversión

El *sistema de conversión* de la Fig. 2 tiene el objetivo de realizar el proceso de transformación de datos acústicos de entrada a un lenguaje común para la *arquitectura de pruebas*. Asimismo, tomar los resultados de salida proporcionados por la *arquitectura de pruebas* y verificarlos con el mismo proceso implementado totalmente en software. El *sistema de conversión* representa una aplicación rápida de software que toma como entrada archivos de tipo imagen con extensiones *bmp*, *tiff*, *jpg*, *png*, entre otros, y realiza la conversión a un archivo de texto simple y viceversa. Este sistema externo fue implementado en el entorno de programación (IDE) QtCreator de Nokia para sistemas GNU/Linux con código de implementación C++. Además, el agregado de librerías OpenCV 2.3 [29] para utilizar sus estructuras de datos eficientes.

Arquitectura de pruebas

La *arquitectura de pruebas* constituye una estructura general que contiene código VHDL sintetizable (*Process Memory* y *RAM*), para su posterior síntesis en dispositivo lógico programable y código VHDL no sintetizable (*Read File* y *Write File*) para realizar la verificación funcional del sistema mediante simulación. Esta simulación funcional tiene el objetivo de implementar una técnica de segmentación de imagen acústica y que funcione de igual manera con respecto a la técnica implementada totalmente en software. Se empleó el entorno ModelSim de la empresa Mentor Graphics en su versión 6.3, para realizar la verificación funcional del diseño y Quartus II, versión 8.5 de la empresa Altera para realizar la síntesis de hardware.

En la Fig. 3 se muestra el diagrama general en bloques e interconexiones de la *arquitectura de pruebas*. Entre los componentes se tiene, lectura de un archivo de texto de entrada (*Read File*), procesado de memoria RAM (*Process Memory*), escritura a un archivo de texto de salida (*Write File*) y memoria RAM para el almacenamiento de datos. Asimismo, contiene un bloque general de proceso denominado *GP_State_Machine (*El acrónimo de *"GP"* hace referencia a *General Process* (señales de proceso general interno a la *arquitectura de pruebas*).) que coordina el flujo de trabajo entre los componentes mencionados. Además, como se puede apreciar, se tienen ilustrados los componentes mencionas más cuatros multiplexores, de tres entradas y una salida, y un demultiplexor, de una entrada y dos salidas, que conforman la organización completa de bloques de la *arquitectura de pruebas*.

El componente *RAM* (ver Fig. 3) representa una memoria de acceso aleatorio para almacenar los datos utilizados en el proceso. La memoria RAM se implementó como un vector de elementos que contiene un número de palabras y donde cada palabra tiene un ancho de *n* bits, parametrizable. Enfocándose en el caso de estudio, los datos acústicos tienen un rango en escala de grises que varían entre 0 y 255.

De esta manera, el ancho de la palabra es de 8 bits (*n* igual a 8 bits, es decir, 256 posibles valores). Por otro lado, la cantidad de palabras se configura teniendo en cuenta el tamaño de la línea o franja que depende del número de muestras digitalizadas proporcionadas por el dispositivo sonar, entre 200 a 8200 muestras.

En la Fig. 3 se muestran las distintas interconexiones de bloques de código descritos en VHDL, incluyendo:

a) Señales que representan las interconexiones externas, es decir, los parámetros de ejecución de la *arquitectura de pruebas*, por ejemplo, *t_clk (*El acrónimo de *"t_"* representa una señal.*)* (señal de reloj), *t_Threshold_1* (señal de umbral de detección 1), entre otros.
b) Señales de interconexiones internas, es decir, el flujo de datos interno entre los distintos componentes, por ejemplo, *t_SMGP_start_read_file (*El acrónimo de *"SMGP"* hace referencia a *State Machine General Process* (señales de proceso general provisto por la máquina de estados).*)* (señal de comienzo de lectura de un archivo), *t_ SMGP_end_read_file* (señal de finalización de lectura de un archivo), entre otros.
c) Buses de datos, entre ellos se tienen el dato con ancho de palabra de 8 bits y la dirección de memoria (cantidad de palabras configurable al tamaño de la memoria RAM).

El bloque denominado *GP_State_Machine* es una máquina de estados, que coordina los procesos de la *arquitectura de pruebas*. Dicha máquina conduce a los procesos de *Read File, ProcessMemory* y *Write File*.

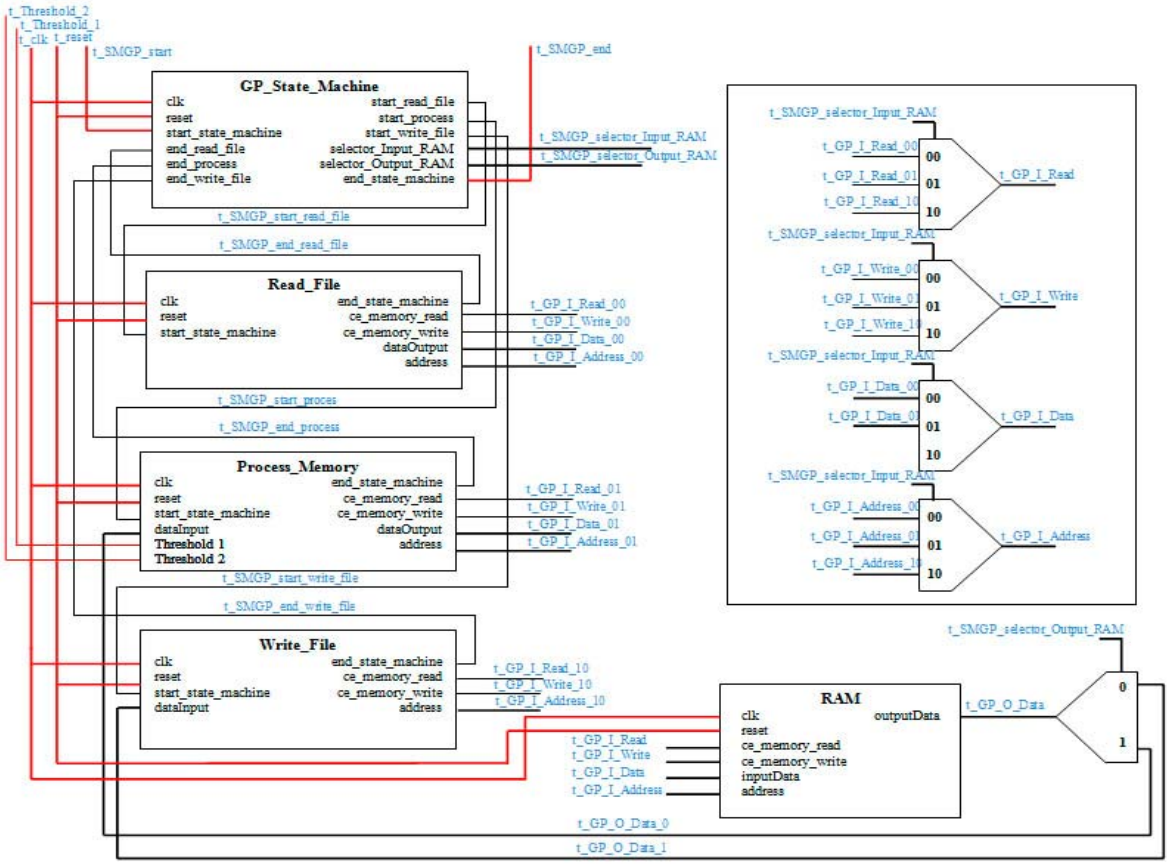

Figura 3. Diagrama general de bloques de la *arquitectura de pruebas*.

En la Fig. 4 se muestra el diagrama de estados y sus transiciones correspondientes, conteniendo los estados e_0 a e_4. Las salidas de la máquina de estado en función del estado actual se presentan en la Tabla I. Cada uno de los estados representa:

- e_0: estado inicial donde se reinician todos los componentes involucrados.
- e_1: se habilita la lectura de datos desde un archivo de entrada (*start_read_file*) y se establece el almacenamiento en memoria RAM (*selector_Input_RAM*).

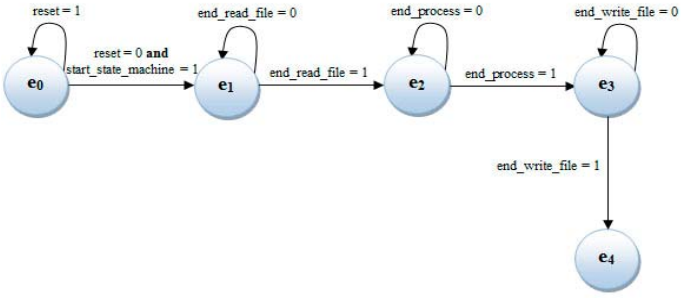

Figura 4. Máquina de estados de la arquitectura de pruebas.

- e_2: se habilita la lectura de datos de memoria RAM (*selector_Output_RAM*), se procesan los datos (*start_process*) y se almacena nuevamente en memoria RAM (*selector_Input_RAM*).
- e_3: se habilita la lectura de datos de memoria RAM (*selector_Output_RAM*) y se escribe en un archivo de texto de salida (*start_write_file*).
- e_4: estado final.

La Fig. 5 muestra la simulación funcional de la máquina de estados cuando se producen las transiciones. Como se puede apreciar para los tres casos, Fig. 5 (a), (b) y (c), se tienen las señales de reloj (*t_clk*), reseteo (*t_reset*), comienzo y finalización de la máquina de estados general (*t_smgp_start* y *t_smgp_end*), comienzo y finalización del proceso de lectura de un archivo (*t_smgp_start_read_file* y *t_smgp_end_read_file*), comienzo y finalización del procesado de memoria RAM (*t_smgp_start_process* y *t_smgp_end_process*).

TABLA I. VALORES DE SALIDAS DE LA MÁQUINA DE ESTADOS *GP_STATE_MACHINE*.

	e_0	e_1	e_2	e_3	e_4
start_read_file	0	1	0	0	0
start_process	0	0	1	0	0
start_write_file	0	0	0	1	0
selector_Input_RAM	ZZ	00	01	10	ZZ
selector_Output_RAM	Z	Z	1	0	Z
end_state_machine	0	0	0	0	1

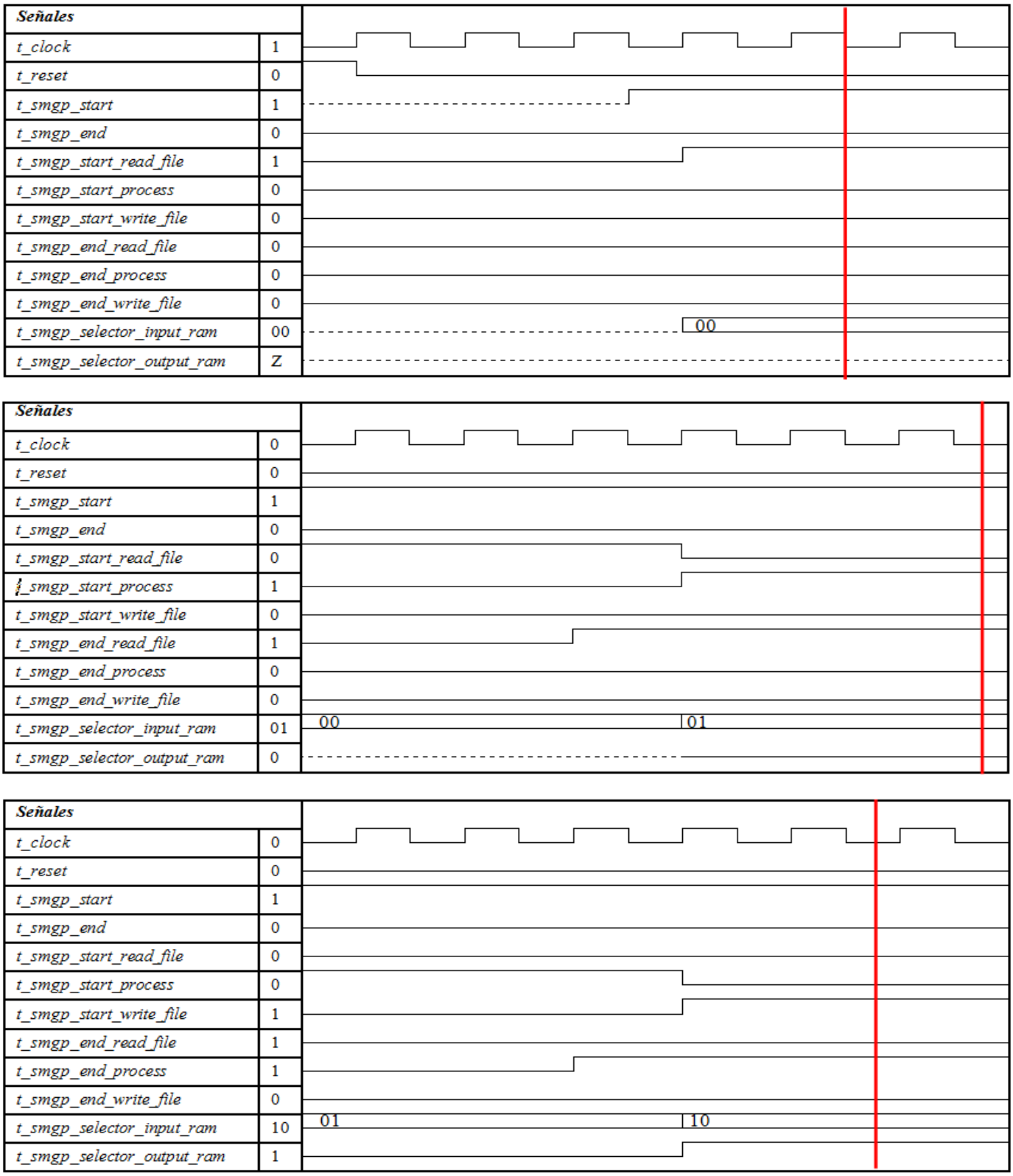

Figura 5. Simulación de la máquina de estados general (*GP_State_Machine*) para: (a) lectura de un archivo (*Read File*); (b) procesado de memoria RAM (*Process Memory*) y (c) escritura a un archivo (*Write File*).

Asimismo, se tienen las señales de comienzo y finalización de la escritura a un archivo de salida (*t_smgp_start_write_file* y *t_smgp_end_write_file*), y selección de entrada y salida de los multiplexores y demultiplexor para la lectura/escritura de la memoria RAM (*t_smgp_selector_input_ram* y *t_smgp_selector_output_ram*).

En la Fig. 5 (a) muestra las señales para la lectura de un archivo de texto de entrada (*Read File*). Como se puede apreciar cuando la señal *t_reset* cambia a cero comienza el proceso general con el inicio de la máquina de estados (*t_smgp_start = 1*). En el próximo flanco ascendente de reloj, se produce la transición al estado e_1 iniciándose la primera tarea de lectura de archivo, habilitándose la señal correspondiente (*t_smgp_start_read_file = 1*) y la selección de lo multiplexores para la escritura de los datos en la memoria RAM (*t_smgp_selector_input_ram = 00*).

La Fig. 5 (b) muestra las señales para la máquina de estados cuando se produce el intercambio de lectura de un archivo (*Read_File*) a procesado de los datos que se encuentran en memoria RAM (*Process_Memory*). De esta manera, tanto la señal de fin de lectura de archivo (*t_smgp_end_read_file*) como la señal de comienzo de procesado de memoria RAM (*t_smgp_start_process*) asumen valor lógico uno. Por consiguiente, cambian los valores de señal para la selección de los multiplexores de entrada y salida tanto para la lectura/escritura de la memoria RAM (*t_smgp_selector_input_ram = 01; t_smgp_selector_ouput_ram = 0*).

La Fig. 5 (c) muestran las señales para la máquina de estados cuando se produce el intercambio de procesado de memoria RAM (*Process Memory*) a escritura a un archivo de salida (*Write_File*). Como se puede apreciar, se produce un cambio en los valores de la señal de fin de procesado de memoria RAM (*t_smgp_end_process*) y comienzo de escritura a un archivo de salida (*t_smgp_start_write_file*) a valor lógico uno. Asimismo, cambian los valores de la señal para la selección de los multiplexores de entrada y salida a la memoria RAM (*t_smgp_selector_input_ram = 10; t_smgp_selector_ouput_ram = 1*).

El componente *Read File* realiza tres etapas para realizar a cabo su tarea: (1) leer un dato que reside dentro de un archivo de texto, (2) generar una dirección de memoria secuencial y (3) escribir dentro en memoria RAM. Asimismo, se requieren registros para almacenar parcialmente los valores del dato y de la dirección de memoria en donde residirá finalmente el dato.
Para implementar este componente se desarrolló una máquina de estados que se observa en la Fig. 6. Esta máquina contiene 7 estados (e_0 - e_6). Cada uno de los estados representa:

- e_0: estado inicial donde se reinician todos los componentes involucrados.
- e_1: se habilita la lectura de un dato desde el archivo de texto y se reinicia el temporizador. En este estado el dato leído se almacena en un su correspondiente registro.
- e_2: se habilita la memoria RAM para escritura. En este estado simplemente se establece que se escribirá la memoria RAM.

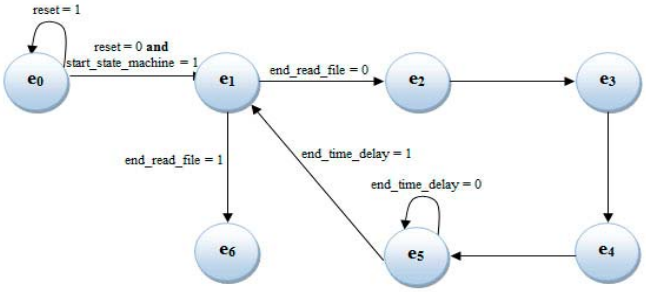

Figura 6. Máquina de estados del componente *Read File*.

- e_3: se escribe el dato en memoria RAM y se genera una nueva dirección de memoria secuencial. El dato y la dirección de memoria almacenada en los registros se utilizan para escribir en memoria RAM. Luego, se genera una nueva dirección de memoria secuencial.
- e_4: se inicializa el temporizador.
- e_5: se espera la finalización del temporizador.
- e_6: estado final de lectura de un archivo de texto.

El temporizador representa simplemente un contador de flancos de reloj con el objetivo de mantener el tiempo suficiente una señal determinada. En este caso mantener la señal de fin de lectura de archivo (ver Fig. 5 (b) *t_smgp_end_read_file* = 1) que hace referencia al estímulo de salida de la máquina de estado en su estado e_6 (*end_state_machine*).

El componente *ProcessMemory* representa el proceso de segmentación sobre los datos acústicos. Este bloque de código VHDL sintetizable es el que finalmente se sintetiza en un dispositivo lógico programable. Para desarrollar su tarea necesita realizar tres etapas generales: (1) leer un dato de memoria RAM, (2) procesar el dato leído, y (3) escribir en la memoria RAM el dato modificado. Para implementar este componente se desarrolló una máquina de estados que se observa en la Fig. 7. Esta estructura contiene 8 estados (e_0 -e_7). Cada uno de los estados representa:

- e_0: estado inicial donde se reinician todos los componentes involucrados.
- e_1: se habilita la memoria RAM para lectura.
- e_2: lectura de un elemento de la memoria RAM y almacenamiento en un registro específico.

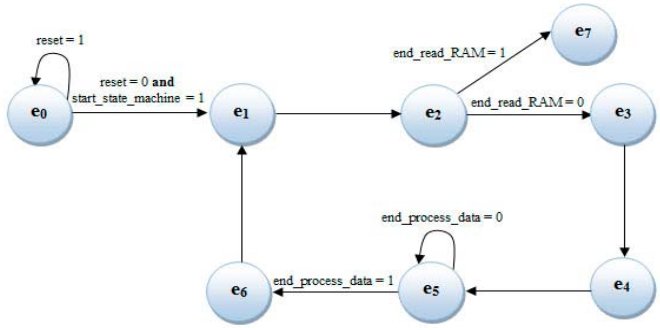

Figura 7. Máquina de estados del componente *Process Memory*.

- e_3: estado de transición.
- e_4: se habilita el registro que contiene el dato actual para aplicar la técnica de segmentación.
- e_5: habilita el comienzo del procesado del dato.
- e_6: se escribe el dato en memoria RAM y se comienza la lectura de un nuevo dato de memoria RAM.
- e_7: estado final del procesado de memoria RAM.

El proceso de segmentación implementado representa la aplicación de una función de umbralización. Para ello, se considera a la imagen acústica de SSS como una función *f(x,y)* que puede tomar valores entre 0 y 255 (8 bits). De esta manera, se definen dos umbrales T_1 y T_2 para una umbralización de 3 niveles (zonas de resalte acústico, reverberación del suelo marino y sombra acústica). De esta manera, el pixel *(x,y)* de la imagen *f(x,y)* debería ser clasificado como una región de sombra acústica si $f(x,y) \leq T_1$, asimismo se clasifica como región de reverberación del suelo marino cuando $T_1 < f(x,y) \leq T_2$, y por último, se clasificará como zona de resalte acústico cuando $f(x,y) > T_2$.

El componente *Write File* representa el proceso mediante el cual se leen los todos los datos almacenados en la memoria RAM y se graba en un archivo de texto de salida. Para desarrollar su tarea necesita realizar tres etapas generales: (1) leer un dato de la memoria RAM, (2) generar una dirección de memoria secuencial y (3) escribir dentro de un archivo de texto de salida. Para implementar este componente se desarrolló una máquina de estados, cuyo diagrama se observa en la Fig. 8. Esta estructura contiene 7 estados (e_0 - e_6). Cada uno de los estados representa:

- e_0: estado inicial donde se reinician todos los componentes involucrados.
- e_1: se habilita la memoria RAM para lectura y se reinicia el temporizador.
- e_2: se escribe el dato en un archivo de texto de salida.
- e_3: se genera una nueva dirección de memoria.
- e_4: se inicializa el temporizador.
- e_5: se espera la finalización del temporizador.
- e_6: estado final de escritura.

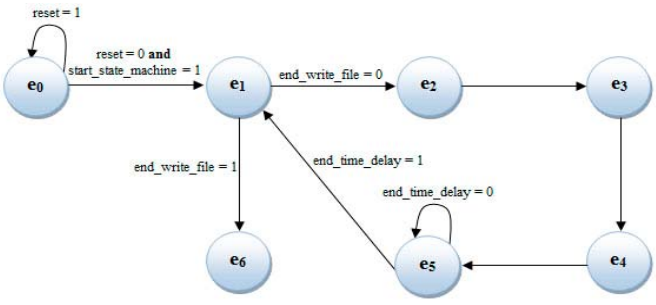

Figura 8. Máquina de estados del componente *Write File*.

Síntesis de hardware

La síntesis de hardware corresponde a la tercera fase de la metodología propuesta, donde se traslada la descripción que representa a la segmentación de datos acústicos a un hardware en concreto. En este caso el bloque *ProcessMemory* fue sintetizado en un dispositivo lógico programable tipo FPGA (*Field Programmable GateArray*) Cyclone II, familia EP2C20F484C7, empleando la herramienta Quartus II de la empresa Altera.

La Fig. 9 muestra el diagrama de bloques del componente *ProcessMemory*. Se tienen cinco bloques involucrados: *PM_State_Machine (*El acrónimo de *"PM"* hace referencia a *Process Memory.)*, *PM_Read_RAM*, *PM_Process_Data*, *GP_Register_N_Bits* y *RAM*.

El componente *PM_State_Machine,* ya detallado en la Sección de Arquitectura de Pruebas, representa el coordinador general del procesado de datos de la memoria RAM implementado por medio de una máquina de estados.

El componente *PM_Read_RAM* tiene por objetivo realizar la lectura de datos necesarios para procesarlos desde la memoria RAM. Para ello genera direcciones de memoria secuencial para recuperar el dato de la memoria RAM.

El componente *PM_Process_Data* realiza el proceso de segmentación sobre los datos. Para ello, para cada dato de entrada se le realiza una clasificación en base a los dos umbrales de entrada.

El componente *GP_Register_N_Bits* representa un registro de *n* bits. El valor de *n* coincide con el ancho de palabra de la memoria RAM (8 bits). El componente *RAM* representa una memoria de acceso aleatorio que contendrá el almacenamiento de los datos.

La Fig. 10 muestra una simulación funcional del componente *Process_Memory*. Como se puede apreciar, se tienen las señales de reloj (*t_clk*), reseteo (*t_reset*), comienzo y finalización del procesado de memoria RAM (*t_smgp_start_process* y *t_smgp_end_process*), selección de entrada y salida de los multiplexores y demultiplexores para la lectura/escritura de la memoria RAM (*t_smgp_selector_input_ram* y *t_smgp_selector_output_ram*), umbrales de detección (*t_smgp_threshold_1* y *t_smgp_threshold_2*), señal de entrada de lectura/escritura de la memoria RAM (*t_gp_i_read_01* y *t_gp_i_write_01*), señal de entrada de la dirección de memoria (*t_gp_i_address_01*), señal de entrada del dato (*t_gp_i_data_01*) y señal de salida del dato desde memoria RAM (*t_gp_o_data_0*).

Al observar la simulación se tienen los umbrales de detección *t_smgp_threshold_1* y *t_smgp_threshold_2*, parametrizados con los valores de 01100100_2 (100_{10}) y 11001000_2 (200_{10}), respectivamente. Si el valor a segmentar es inferior al primer umbral (*t_smgp_threshold_1*) la salida del sistema debería dar el valor de 00000000_2 (0_{10}) haciendo referencia a la sombra acústica, caso contrario encontrándose entre ambos umbrales (*t_smgp_threshold_1* y *t_smgp_threshold_2*) debería brindar el valor de 10000000_2 (128_{10}) siendo la reverberación del fondo marino, y por último, si supera el valor del segundo umbral (*t_smgp_threshold_2*) otorgada el valor de 11111111_2 (255_{10}) representando el resalte acústico.

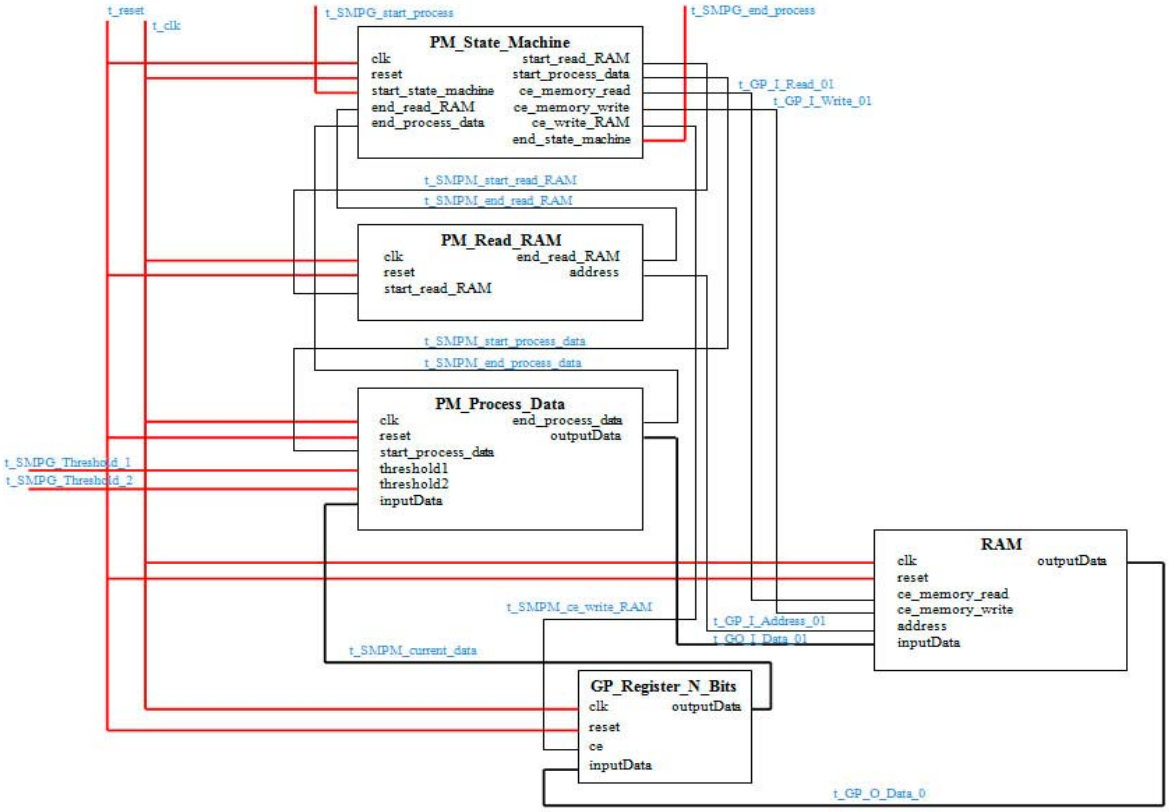

Figura 9. Diagrama general de bloques del componente general *Process Memory*.

Asimismo, observando la posición del cursor en diagrama se muestra como el valor de la señal de salida de la memoria RAM (*t_gp_o_data_0*) es 10100010_2 (162_{10}) en la dirección de memoria 7. Como este valor se encuentra entre los umbrales de detección (*t_smgp_threshold_2* y *t_smgp_threshold_2*) la señal de entrada a la memoria RAM (*t_gp_i_data_01*) en la dirección de memoria 7 asignará el valor de 1000000_2 (128_{10}).

Por otro lado, se puede observar como las señales de entrada de lectura/escritura de la memoria RAM (*t_gp_i_read_01* y *t_gp_i_write_01*) se alternan para leer el valor y escribir en la memoria RAM.

Señales	
t_clock	0
t_reset	0
t_smgp_start_process	1
t_smgp_threshold_1	01100100
t_smgp_threshold_2	11001000
t_smgp_selector_input_ram	01
t_smgp_selector_output_ram	0
t_gp_i_read_01	0
t_gp_i_write_01	0
t_gp_i_address_01	7
t_gp_i_data_01	10000000
t_gp_o_data_0	10100010
t_smgp_end_process	0

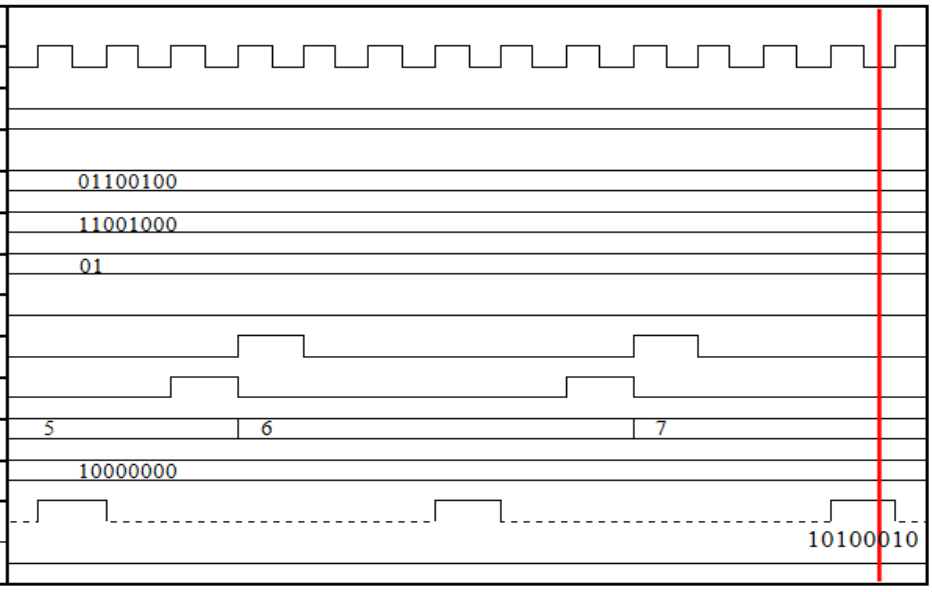

Figura 10. Simulación referente al componente general *Process Memory*.

La Tabla II muestra los recursos utilizados del FPGA elegido disponibles para la segmentación de datos acústicos basándose en el tamaño de la línea o franja. Este valor depende del número de muestras digitalizadas que proporciona el dispositivo sonar. De esta manera, se realizaron distintas ejecuciones con 100, 200, 500, 1000 y 2000 muestras. Los datos fueron obtenidos al sintetizar el bloque *ProcessMemory* con la herramienta Quartus II de Altera.

IV. Resultados

Los datos de prueba para este trabajo son imágenes acústicas de SSS en el formato de archivo *bmp* con 8 bits por píxel, es decir, una imagen en escala de grises de 0 a 255 posibles valores por pixel. Estas imágenes se emplean como entrada al Sistema de Conversión para finalidades de prueba y simulación algorítmica o directamente sobre el hardware especifico (dispositivo lógico programable tipo FPGA - Cyclone II) como detallo en Sección III – Materiales y Métodos.

TABLA II. RECURSOS HARDWARES UTILIZADOS PARA LA SEGMENTACIÓN DE DATOS ACÚSTICOS.

	Recursos Usados				
Número de Muestras	**100**	**200**	**500**	**1000**	**2000**
Elementos lógicos Totales	26 (< 1%)	25 (< 1%)	30 (< 1%)	32 (< 1%)	33 (< 1%)
Funciones Combinacionales Totales	26 (< 1%)	25 (< 1%)	30 (< 1%)	32 (< 1%)	33 (< 1%)
Registros Lógicos Dedicados	12 (< 1%)	13 (< 1%)	14 (< 1%)	15 (< 1%)	16 (< 1%)
Recursos Disponibles			**18752**		
Uso de elementos lógicos por número de entradas LUT					
4 funciones de entrada	10	8	13	9	10
3 funciones de entrada	7	9	8	7	11
<=2 funciones de entrada	9	8	9	16	12
Elementos Lógicos por Modo					
Modo Normal	20	18	22	23	23
Modo Aritmetico	6	7	8	9	10
Total de Registos	12	13	14	15	16
Registros Lógicos Dedicados	12	13	14	15	16
Registros de I/O	0	0	0	0	0

La Fig. 11 muestra datos propios adquiridos en Salvador de Bahía, Brasil con la presencia de tubería sumergida apoyada sobre la superficie del fondo marino. El sensor SSS utilizado en la campaña es el Starfish 450F (http://www.tritech.co.uk/) de la empresa Tritech©. El sensor comprende un doble canal de eco de sonido (babor y estribor), empleando una señal de chirrido de dos frecuencias 430 kHz y 470 kHz. El rango máximo establecido es de 50 metros. El sonar fue montado debajo del casco en una embarcación mediante un soporte de acero. La velocidad de la embarcación se estableció a menos de 4.5 m/s para profundidades entre 5 a 20 metros, y se redujo entre 1.5 a 2 m/s en aguas más profundas (> 20 metros). Estas velocidades son definidas por el fabricante del sensor para su correcta adquisición. Esto permite tener la cobertura de las líneas contiguas de sonar para la formación de imágenes acústicas.

El seguimiento de la tubería se inició en una latitud y longitud de -12º 50’ 49,5’’, -38º 31’ 23,03’’ respectivamente, finalizando en -12º 51’ 33,28’’,- 38º 32’ 48,48’’. Se recolectaron 50500 líneas de datos acústicos válidos que proporcionaron 101 imágenes de 1000 x 500 pixeles de resolución

La Fig. 11 muestra cuatro ejemplos de imágenes de SSS originales y segmentadas utilizando el sistema desarrollado. Estas imágenes han sido recortadas para su mejor presentación. En todos los casos la tubería se encuentra en el lado derecho (estribor). En la Fig. 11 (a) se observa una tubería exhibiendo una ligera curva. Por otro lado, en la Fig. 11 (b) la tubería se encuentra semi-expuesta con sectores enterrados debajo de la superficie marina. En la Fig. 11 (c) la tubería se encuentra en una línea recta, pero su sombra no está totalmente definida (ver Fig. 11 (c) parte inferior derecha). Este detalle observado se debe a que la tubería no está totalmente apoyada sobre el suelo marino, más conocido por su término en inglés como *free-span*, y representa una característica relevante al proceso de inspección. Por último, en la Fig. 11 (d) la tubería se encuentra en línea recta, pero en la parte superior se observa que está apoyada sobre un conjunto de sedimentos, más conocido por su término en ingles *rock-dump*. Por otro lado, se emplearon imágenes acústicas de SSS descargadas de sitios web (http://www.jwfishers.com/) (http://www.innerspaceexploration.com/). Estos sitios contienen galerías de imágenes para la prueba de algoritmos y técnicas de procesamiento.

En la Fig. 12 se observan dos ejemplos que comprenden un barco hundido y un emisario. Los datos de ambos ejemplos fueron adquiridos por un SSS de la empresa Marine Sonic Technology© a una frecuencia de 600 kHz con un rango máximo de 50 metros. El primer ejemplo (Fig. 12 (a)) muestra claramente un barco hundido que se encuentra en el Lago de Washington. Estos datos se emplean en la problemática de arqueología de naufragios. El segundo ejemplo (Fig. 12 (b)) representa un emisario y se deben realizar las mismas tareas de mantenimiento que los mencionados en la Fig. 11.

Los umbrales de detección para la segmentación fueron para la Fig. 11 (a) $T_1 = 75$, $T_2 = 115$; (b) $T_1 = 60$, $T_2 = 115$; (c) $T_1 = 6$, $T_2 = 251$ y (d) $T_1 = 6$, $T_2 = 249$, respectivamente. Por otro lado, los umbrales de detección para la Fig. 12 (a) $T_1 = 5$, $T_2 = 130$ y (b) $T_1 = 5$, $T_2 = 135$. Como se explicó anteriormente estos umbrales pueden ser establecidos durante la simulación o directamente sobre dispositivo lógico programable.

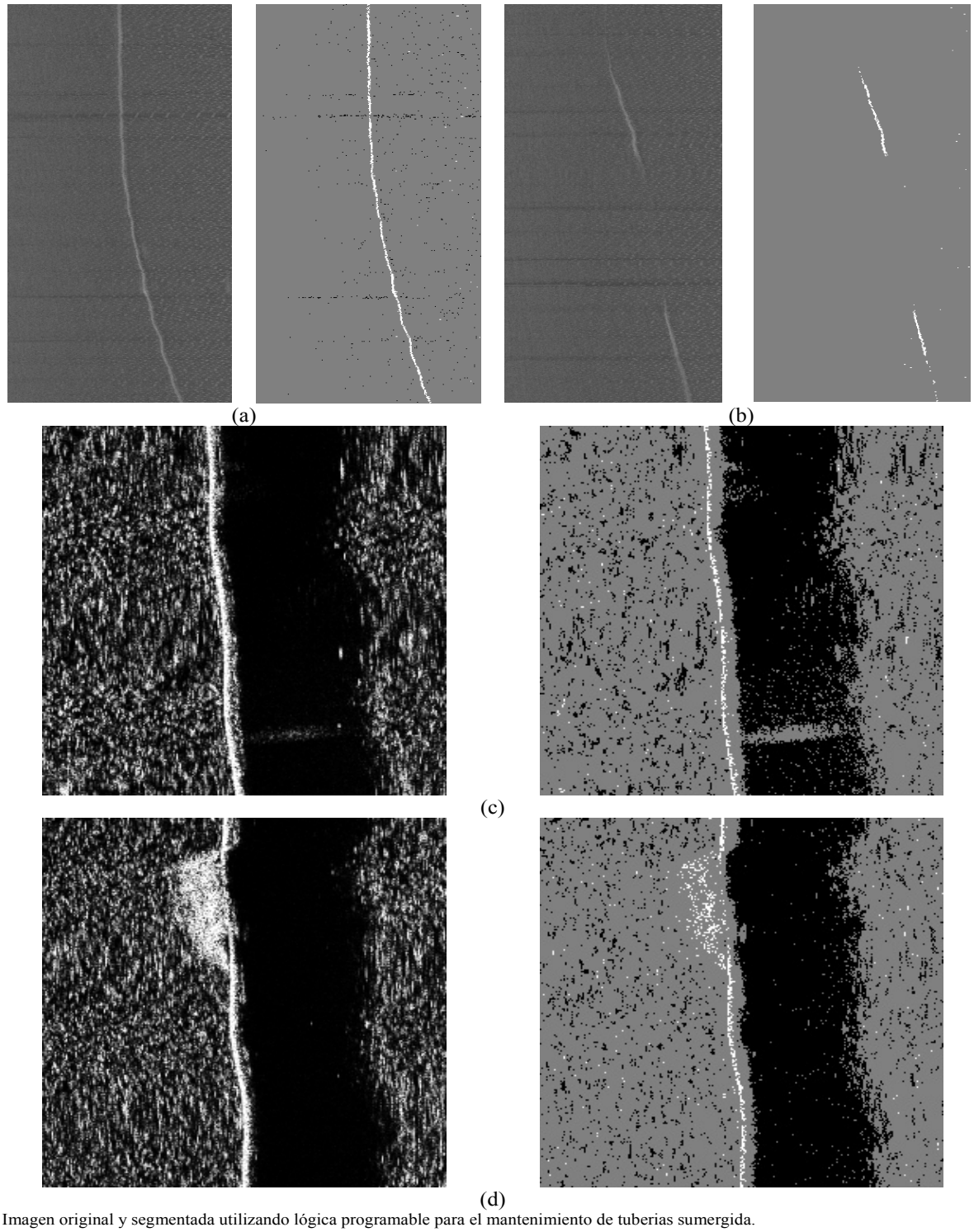

Figura 11. Imagen original y segmentada utilizando lógica programable para el mantenimiento de tuberias sumergida.

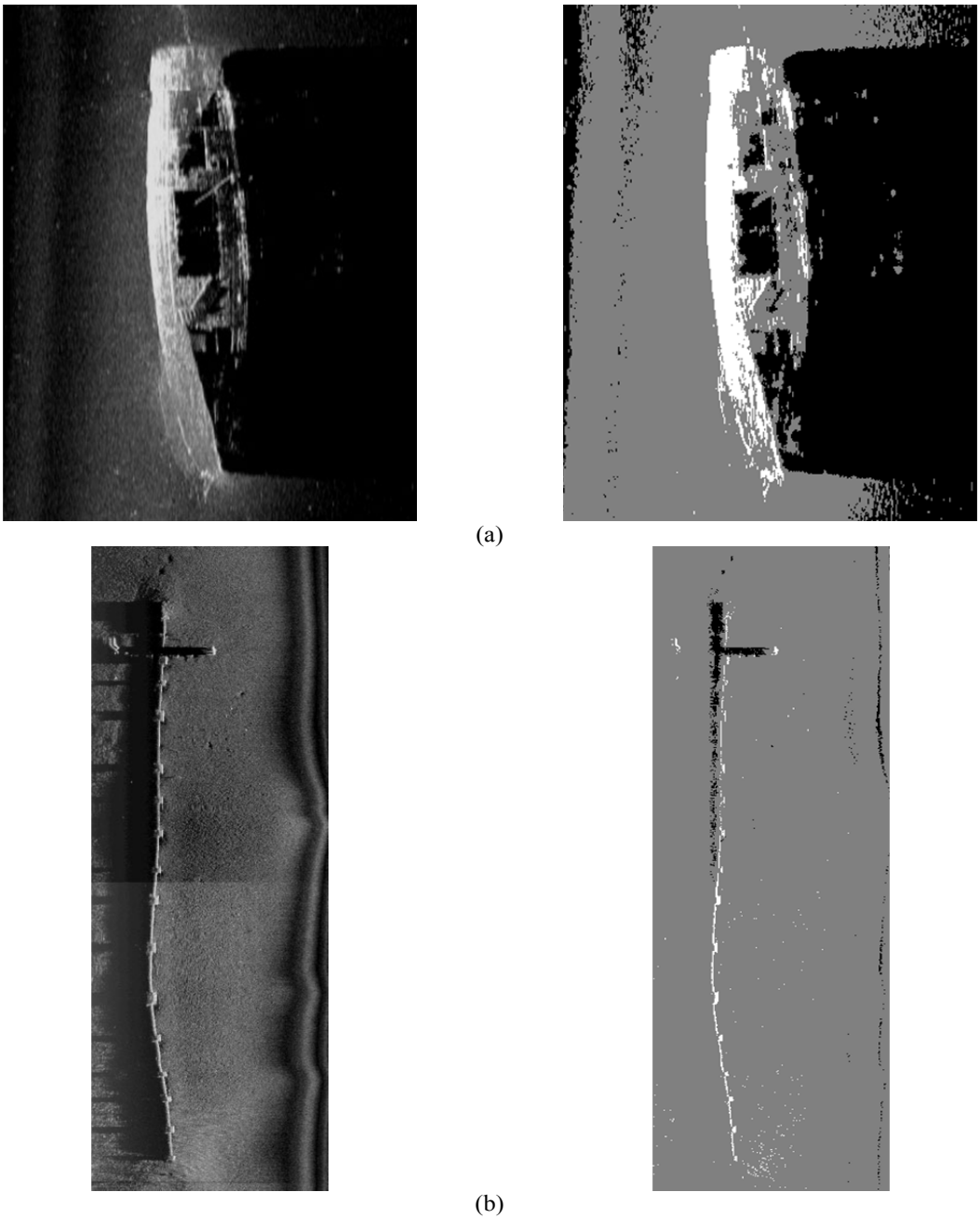

Figura 12. Imagen original y segmentada utilizando lógica programable para (a) arqueleogia de un barco hundido y (b) mantenimiento de emisario.

Tener en cuenta, que el proceso de segmentación utilizado representa una segmentación simple o umbralización en tres regiones: resalte acústico, sombra y reverberación del suelo marino. En general para poder llevar a cabo esta tarea se utilizan métodos en entornos de software con modelos matemáticos complejos que varían en su velocidad, eficacia, necesidades de recursos, precisión y robustez [9], [23]–[25]. Estos métodos son aplicados una vez que la muestra de datos son adquiridos y no directamente en el procesamiento en línea de una campaña debido al costo computacional que estos requieren [16]. No obstante, el presente trabajo refleja su viabilidad en la implementación de un método de segmentación de imágenes acústicas de SSS utilizando lógica programable, y a su vez la posibilidad de escalar e implementar a futuro cualquiera de los métodos de segmentación mencionados.

V. Conclusiones

La principal contribución de este trabajo consiste en la propuesta de una metodología de segmentación de imágenes acústicas para su implementación en hardware, mediante un dispositivo lógico programable. Esta metodología fue probada

con un algoritmo de segmentación, para demostrar su aplicabilidad. Se empleó VHDL para describir el conjunto sintetizable de código, el cual se sintetiza en el hardware, y para describir el conjunto no sintetizable para realización del *testbench* y simulación funcional del sistema. Como resultado de la aplicación de esta metodología se generó un *framework* reutilizable, incluso con otros HDL. Por otra parte, al poder sintetizar en hardware una fase del procesamiento (fase denominada *síntesis*), se obtiene una optimización para el empleo del procesamiento en tiempo real, lo cual resulta muy beneficioso para las aplicaciones en robótica subacuática, por ejemplo. Otra característica destacable es que el hecho de describir la fase de *arquitectura de pruebas* en el mismo lenguaje (no sintetizable), que agiliza enormemente la etapa de desarrollo algorítmico.

AGRADECIMIENTOS

Este trabajo se realizó gracias al financiamiento de los proyectos DPI2009-11298, MICINN de España, CONICET – PIP 11420090100238, y ANPCyT – PICT-2009-0142, al programa de Subsidio de la CIC-Res. 179, de Argentina.

Los autores agradecen al Ing. André Sousa Sena por su indispensable participación en la recolección de datos en Brasil.

REFERENCIAS

[1] P. Blondel, *The Handbook of Sidescan Sonar*. Praxis Publishing. UK: Springer Berlin Heidelberg, 2009.

[2] W. E. Landay, M. A. LeFever, R. A. Spicer, R. M. Levitre, S. J. Tomaszeski, A. W. Joseph, and R. M. Smith, "The Navy Unmanned Undersea Vehicle (UUV) Master Plan." 11-Sep-2004.

[3] A. Balasuriya and T. Ura, "Optical and navigational sensor fusion scheme for cable following by AUVs," presented at the OCEANS, 2001. MTS/IEEE Conference and Exhibition, vol. 4, pp. 2383–2388 vol.4.

[4] P. Chapple, "Automated Detection and Classification in High-resolution Sonar Imagery for Autonomous Underwater Vehicle Operations," 2008.

[5] I. Ishøy, A. Bjerrum, O. Calvo, G. G. Acosta, Y. Petillot, J. Evans, K. Kyruakopoulos, G. Lionis, T. Slater, and R. Nunn, "New challenges for AUTOTRACKER," presented at the Unmanned Underwater Vehicle Showcase, Southampton, UK, 2002.

[6] X. Lurton, *An Introduction to Underwater Acoustics. Principles and Applications.* Springer Praxis Books in Geophysical Sciences. Berlin Heidelberg, 2003.

[7] T. Celik and T. Tjahjadi, "A Novel Method for Sidescan Sonar Image Segmentation," IEEE J. Ocean. Eng., vol. 36, no. 2, pp. 186–194, Apr. 2011.

[8] Y. R. Petillot, S. R. Reed, and J. M. Bell, "Real time AUV pipeline detection and tracking using side scan sonar and multi-beam echo-sounder," presented at the OCEANS'02 MTS/IEEE, vol. 1, pp. 217–222.

[9] S. Reed, I. Tena Ruiz, C. Capus, and Y. Petillot, "The fusion of large scale classified side-scan sonar image mosaics," IEEE Trans. Image Process., vol. 15, no. 7, pp. 2049–2060, Jul. 2006.

[10] S. Reed, Y. Petillot, and J. Bell, "Automated approach to classification of mine-like objects in sidescan sonar using highlight and shadow information," IEE Proc. Radar Sonar Navig., vol. 151, no. 1, pp. 48–56, Feb. 2004.

[11] S. Reed, Y. Petillot, and J. Bell, "An automatic approach to the detection and extraction of mine features in sidescan sonar," IEEE J. Ocean. Eng., vol. 28, no. 1, pp. 90–105, Jan. 2003.

[12] M. Mignotte, C. Collet, P. Perez, and P. Bouthemy, "Sonar image segmentation using an unsupervised hierarchical MRF model," IEEE Trans. Image Process., vol. 9, no. 7, pp. 1216–1231, Jul. 2000.

[13] M. Mignotte, C. Collet, P. Pérez, and P. Bouthemy, "Three-Class Markovian Segmentation of High-Resolution Sonar Images," Comput. Vis. Image Underst., vol. 76, no. 3, pp. 191–204, 1999.

[14] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. New Jersey 07458: Prentice Hall Upper Saddle River. Second Edition. 2001.

[15] W. K. Pratt, *Digital Image Processing*. JOHN WILEY & SONS, INC. Third Edition, 2001.

[16] S. Kumar, P. Pandey, "FPGA implementation of image segmentation by using edge detection based on Sobel edge operator," Intl. Journal of Research in Engineering and Technology, vol. 2, no. 10, pp. 198–203, Oct. 2013.

[17] R. Sathish Kumar, A. Vimala Juliet, D. Rajapan, "An embedded system for synthetic aperture sonar," Intl. Journal of Advanced Engineering Applications, vol. 4, no. 3, pp. 50–55, 2011.

[18] B. Kinsella, Embedded Image Segmentation on an FPGA, National University of Ireland, 2008.

[19] S. A. Villar, G. G. Acosta, A. L. Sousa, and A. Rozenfeld, "Evaluation of an efficient approach for target tracking from acoustic imagery for the perception system of an autonomous underwater vehicle," J. Adv. Robot. Syst., 2013.

[20] M. Lianantonakis and Y. R. Petillot, "Sidescan Sonar Segmentation Using Texture Descriptors and Active Contours," Ocean. Eng. IEEE J. Of, vol. 32, no. 3, pp. 744–752, Jul. 2007.

[21] F. Schmitt, M. Mignotte, C. Collet, and P. Thourel, "Estimation of noise parameters on sonar images," Signal Image Process. SPIE, vol. 2823, pp. 1–12, 1996.

[22] C. Collet, P. Thourel, P. Perez, and P. Bouthemy, "Hierarchical MRF modeling for sonar picture segmentation," presented at the International Conference on Image Processing, vol. 3, pp. 979–982 vol.3, 1996.

[23] I. Karoui, R. Fablet, J.-M. Boucher, and J.-M. Augustin, "Seabed Segmentation Using Optimized Statistics of Sonar Textures," IEEE Trans. Geosci. Remote Sens., vol. 47, no. 6, pp. 1621–1631, Jun. 2009.

[24] D. P. Williams, "Bayesian Data Fusion of Multiview Synthetic Aperture Sonar Imagery for Seabed Classification," IEEE Trans. Image Process., vol. 18, no. 6, pp. 1239–1254, Jun. 2009.

[25] B. Zerr, E. Maillard, and D. Gueriot, "Sea-floor classification by neural hybrid system," presented at the OCEANS '94, vol. 2, pp. 239–243.

[26] B. Andelkovic, V. Litovski, and V. Zerbe, "New Aspects in HDL's Performance Evaluation," presented at the The International Conference on Computer as a Tool, 2005. EUROCON 2005, vol. 1, pp. 499–502.

[27] P. Ashenden and J. Lewis, The Designer's Guide to VHDL. Third Edition., Morgan Kauffman Publishers. USA: Elsevier, 2008.

[28] T. Riesgo, Y. Torroja, and E. de la Torre, "Design methodologies based on hardware description languages," IEEE Trans. Ind. Electron., vol. 46, no. 1, pp. 3–12, Feb. 1999.

[29] G. Bradski and A. Kaehler, *Learning OpenCV: Computer Vision with the OpenCV Library*. First Edition. 2008.

Sebastian A. Villar graduated as Systems Engineer at the Exact Sciences Faculty of National Buenos Aires Province Centre University (UNCPBA), Argentina (2009), and as Masters in Business Administration (MBA) Economic Science Faculty of UNCPBA (2011), and as Ph.D. in Engineering, at Engineering Faculty of UNCPBA (2014). He is also a researcher of the Argentinean National Research Council (CONICET) with a scholarship, working in Engineering Group INTELYMEC (Av. del Valle 5737-B7400JWI Olavarría; Argentina), UNCPBA. svillar@fio.unicen.edu.ar.

Silvano R. Rossi was born in Tandil, Argentina, in 1970. He received the B.Sc. degree in electromechanical engineering from Universidad Nacional del Centro de la Provincia de Buenos Aires (UNCPBA), Argentina, and the Ph.D. degree in electrical engineering from the São Paulo State University (UNESP), Brazil, in 1999 and 2005, respectively. He is currently involved with instrumentation systems and programmable logic technology. He is a Professor with the Department of Electro-Mechanical Engineering, faculty of Engineering at UNCPBA, and researcher member of the INTELYMEC group at the same University. His research interests include instrumentation systems and measurements, smart transducer networks, autonomous vehicles and digital systems. srossi@fio.unicen.edu.ar.

Gerardo G. Acosta graduated as Engineer in Electronics at the National University of La Plata, Argentina (1988), and as Ph.D. in Computer Science, at the University of Valladolid, Spain (1995). He is currently a Full Professor in Control Systems (Electronic Area)in the Engineering Faculty at the National Buenos Aires Province Centre University (UNCPBA), Argentina. He is also a researcher of the Argentinean National Research Council

(CONICET), since 1997 and Director of the Research & Development Group "INTELYMEC", at the Engineering Faculty -UNCPBA. His working interests comprise the use of computational intelligence in automatic control, particularly intelligent control techniques in terrestrial and underwater robotics. He has more than one hundred and twenty publications and two copyrights in this and related fields. He has been awarded with INNOVAR 2011 second position in Robotics, for the autonomous robot CARPINCHO, and with INNOVAR 2012 first position in Robotics, for the autonomous underwater vehicle ICTIOBOT, both developed at INTELYMEC-UNCPBA. He is *Senior Member* of the IEEE since 2001, Chairman of the IEEE Computational Intelligence Society Argentinean Chapter (2007-2008), receiving the 2010 Outstanding Chapter Award from CIS, and current Chairman of the IEEE Oceanic Engineering Society Argentinean Chapter, being one of its funders. Also he is a member of the Hispanic-American Fuzzy Systems Association-HAFSA (local branch of IFSA) and AADECA (local branch of IFAC). He has been the research leader of more than ten R+D projects, funded by the Argentinean Government, the Spanish Government and the European Union. He has been invited as a professor of Ph.D programs in Argentina and Spain, he is the present Director of the PhD program at the Engineering Faculty-UNCPBA, and serve as reviewer and member of the scientific committee of several national and international journals and conferences. gerardo.acosta@ieee.org.