IMPLEMENTACIÓN DE UNA MAQUINA DE VECTORES SOPORTE EMPLEANDO FPGA

RESUMEN
En este artículo, se presenta el diseño y la implementación de una maquina de vector de soporte (SVM) sobre un dispositivo lógico programable (PLD). La maquina es utilizada para solucionar el problema de la XOR. Además, se presenta la implementación del estándar IEEE 754 para la representación de números en punto flotante. La arquitectura del sistema implementado es descrito en detalle y sintetizada sobre una tarjeta del sistema de desarrollo con un arreglo de compuerta programable por campo - FPGA (ALTERA® FLEX10K).

PALABRAS CLAVES: SVM, FPGA, kernel, números de punto flotante

ABSTRACT
In this paper, we present a support vector machine (SVM) design and implementation on programmable logic device (PLD). The machine is used to solve the XOR problem. In addition, we present the implementation of IEEE 754 standard for representing numbers of floating point. The architecture implementing the system is described in detail and mapped on a developed system whit field programmable gate array - FPGA (ALTERA® FLEX10K).

KEYWORDS: *SVM, FPGA, kernel, numbers of floating point.*

JORGE E. HERNANDEZ L.
Ingeniero Electrónico
Estudiante en Maestría en Ingeniería – Automatización Industrial
Universidad Nacional de Colombia sede Manizales
jehernandezl@unal.edu.co

SANTIAGO SALAZAR G.
Ingeniero Electrónico
Universidad Nacional de Colombia sede Manizales
santysalazar@yahoo.com

Grupo de Trabajo Académico Percepción y Control Inteligente (PCI)

1. INTRODUCCIÓN

Las tareas de clasificación en los procesos productivos, muy a menudo son efectuadas por personas que han sido capacitadas y entrenadas. Por lo general estas tareas se convierten en monótonas y repetitivas, causando fatiga y reflejándose en la perdida del criterio de clasificación.

Matemáticamente las tareas de clasificación pueden ser ejecutadas por métodos de aprendizaje como las máquinas de vector soporte (SVMs) y redes neuronales. Estos métodos han sido implementados inicialmente en software bajo plataformas secuenciales, limitando el desempeño de los mismos. En la actualidad, se busca lograr un mayor rendimiento utilizando sistemas digitales que requieren gran capacidad de procesamiento, como lo son: los procesadores en paralelo y los dispositivos lógicos programables, entre otros. Estos últimos presentan una alternativa viable para la implementación de las SVMs en hardware.

En este documento se presenta el diseño e implementación de un clasificador por el método de la SVM, el cual resuelve el problema de la XOR; utilizando el lenguaje descriptor de hardware VHDL ("*Very High Speed Hardware Description Language*") y sintetizado en un arreglo programable de compuertas (FPGA). La arquitectura está implementada en el sistema de desarrollo UP2 del programa universitario de *ALTERA® Corporation* en el dispositivo de la familia FLEX10K [2, 8], que trabaja con un reloj de 25.175 MHz. El código de la programación fue simulado bajo el entorno MAX+PLUS II versión 10.2.

Fecha de Recepción: 16 Enero de 2006
Fecha de Aceptación: 07 Junio de 2006

2. MAQUINAS DE VECTORES SOPORTE (SVM)

Las SVM son máquinas de aprendizaje que utilizan el método de los vectores soporte (Support Vector o SV), este es un método general para la resolución de problemas de clasificación [6], y regresión [3]. Fue propuesto originalmente por Vladimir Vapnik [13] en el reconocimiento de patrones para la solución de problemas de clasificación binarios en los que las clases son linealmente separables.

2.1 SVM COMO CLASIFICADOR LINEAL

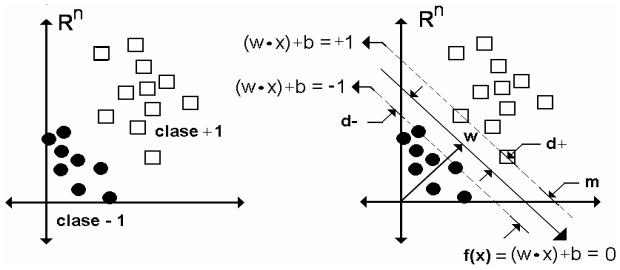

Figura 1. Hiperplano óptimo de separación, para clases linealmente separables.

Si se tiene un conjunto de N muestras con vectores x_i, etiquetadas en dos clases ($y_i = +1$, $y_i = -1$) en un espacio de entrada n dimensional, y se necesita determinar cuando una muestra x pertenece a alguna de las dos clases, el objetivo será encontrar un único hiperplano que proporcione el mayor margen de separación entre las clases con el mínimo error de generalización, como se muestra en la Figura 1.

El conjunto de las N muestras etiquetadas puede ser separado geométricamente por medio de la recta o hiperplano de separación de la forma:

$$f(x) = \left(\vec{w} \bullet \vec{x}\right) + b \quad (1)$$

Así, el margen de separación (*m*), esta comprendido entre las muestras x_i de cada clase más cercanas al hiperplano de separación *f(x)*. Estas muestras x_i son conocidas como los vectores soporte y sobre ellas pasan hiperplanos paralelos a *f(x)*, de tal forma que el margen de separación (*m*) geométricamente es la distancia perpendicular entre los hiperplanos de los vectores soporte.

2.1.1 Hiperplano Óptimo

Se encuentra al maximizar el margen de separación (*m*) que equivale a minimizar la norma euclidiana de *w*, sujeto a la restricción lineal:

$$y_i\left\{\left(w \bullet x\right) + b\right\} \geq 1 \;\forall\; i = 1,...,n \quad (2)$$

Si se soluciona este problema por el método de Lagrange [7], se obtiene una solución de la forma:

$$w = \sum_{i}^{n} a_i y_i x_i \quad (3)$$

donde:

w son los pesos del vector.
a_i son los multiplicadores de Lagrange.

La búsqueda de los valores de w y a_i se convierte en un problema de Programación Cuadrática (QP) [5], el cual puede ser solucionado por medio del algoritmo de Secuencia Mínima de Optimización, (SMO) [10]. La constante b se obtiene de las condiciones de Karush - Kuhn - Tucker (KKT) [4].

Si se reemplaza la Ecuación (3) en (1), se define la función de decisión o el hiperplano óptimo de separación de la SVM, así:

$$f(x) = signo\left[\sum_{\substack{i=1 \\ vectoressoporte}}^{n} a_i y_i \left(x_i \bullet x\right) + b\right] \quad (4)$$

Es de aclarar que la función signo se introduce en la Ecuación (4), debido a que la SVM, es un clasificador binario donde las clases se han designado como *+1* y *- 1*.

2.2 SVM COMO CLASIFICADOR NO LINEAL

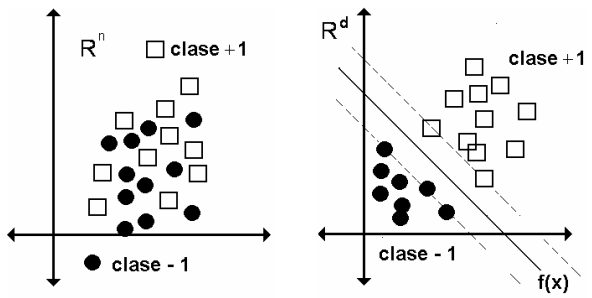

Figura 2. Transformación del espacio de entrada, cuando las muestras no son linealmente separables

Cuando el conjunto de muestras etiquetadas no son linealmente separables, el objetivo será, transformar los vectores de entrada x_i (*n*-dimensionales) en vectores de dimensión más alta $F(x_i)$ (incluso de dimensión infinita) donde las clases puedan ser linealmente separables. Bajo las condiciones de Mercer [4, 12], el producto escalar en la Ecuación (4) se puede escribir por medio de un polinomio de interpolación de espacio, conocido como Kernel ($K(x_i, x)$), como lo muestra la Ecuación (5).

$$f(x) = signo\left[\sum_{\substack{i=1 \\ vectoressoporte}}^{n} a_i y_i K\left(x_i, x\right) + b\right] \quad (5)$$

2.2.1 SVM para el problema de la XOR

Este es un ejemplo de clasificación no lineal, el cual ha sido solucionado utilizando el método de la SVM por el profesor José Luis Alba Castro [1] en su curso de doctorado. El problema consiste en encontrar el hiperplano óptimo de separación con el cual se pueda clasificar sin error el grupo de muestras de la Tabla 1.

Xi 1	*Xi 2*	*Yi*
+1	+1	+1
+1	-1	-1
-1	-1	+1
-1	+1	-1

Tabla 1. Problema de la XOR.

Solución: Con un Kernel Polinomial de segundo orden

$$K\left(x_i, x\right) = \left(\left(x_i \bullet x\right) + 1\right)^2 \quad (6)$$

donde:

$$K = \begin{pmatrix} 9 & 1 & 1 & 1 \\ 1 & 9 & 1 & 1 \\ 1 & 1 & 9 & 1 \\ 1 & 1 & 1 & 9 \end{pmatrix} \quad (7)$$

El problema de optimización para el problema de la XOR consiste en:

$$\max\left(Q(a)\right) = a_1 + a_2 + a_3 + a_4 - \frac{1}{2}\sum_{i,j=1}^{4} a_i a_j y_i y_j k_{ij} \quad (8)$$

sujeto a la restricción:

$$\sum_{i=1}^{4} a_i y_i = a_1 - a_2 + a_3 - a_4 = 0 \quad a_i \geq 0 \quad \forall i = 1,\cdots,4 \quad (9)$$

Como resultado al problema de optimización se obtiene que:

$$a_i^* = \frac{1}{8} = 0.125 \quad (10)$$

indicando que las cuatro muestras de la Tabla 1 son SVs.

Determinados los valores de los multiplicadores de Lagrange a_i y con el Kernel Polinomial de segundo orden la función de decisión para el problema de la XOR será:

$$f(x) = signo\left[\sum_{i=1}^{4} a_i^* y_i \left[\left(x_i \bullet x\right) + 1\right]^2\right] \quad (11)$$

3. IMPLEMENTACIÓN DE UNA SVM SOBRE LA FPGA

Como una primera implementación de las SVM en la FPGA, se propone la función de decisión obtenida para la solución del problema de la XOR (Ecuación (11)). Esta Ecuación se encuentra limitada solo a cuatro vectores soporte, sin embargo, se puede generalizar para problemas de *n*? vectores soporte, siempre y cuando, la solución a dichos problemas se haya obtenido por medio del método de los multiplicadores de Lagrange con Kernel Polinomial de segundo orden.

Antes de realizar la implementación de la función de decisión se necesita implementar las operaciones aritméticas de suma y multiplicación en punto flotante usando el estándar IEEE 754.

3.1 ESTÁNDAR IEEE 754, Adaptado a 16 bits.

Los números en notación científica o representados en punto flotante tienen la forma:

$$(-1)^{SIGNO} \times MANTISA \times \text{BASE}^{\text{EXPONENTE}}$$

El estándar **IEEE 754** [9, 11] representa en binario este tipo de números por una trama de *n* bits distribuidas en tres campos: SIGNO, EXPONENTE y MANTISA.

Inicialmente, las operaciones aritméticas de suma y multiplicación en punto flotante, se implementaron en VHDL utilizando el formato de simple precisión (32 bits) del estándar **IEEE 754,** pero al integrar todas las operaciones aritméticas de la función de decisión de la SVM, los dispositivos lógicos disponibles en el chip de la FPGA no fueron suficientes. Por lo cual, la trama de bits del estándar fue reducido a 16 bits, empleando 1 bit para el signo, 7 bits para el exponente (sesgado en 63) y 8 bits para la mantisa.

3.1.1 SUMA EN PUNTO FLOTANTE

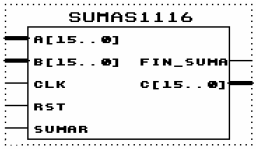

Figura 3. Componente en VHDL para realizar la suma en punto flotante (bajo el estándar IEEE 754 [9, 11]).

La arquitectura digital o componente de la Figura 3 realiza la suma algebraica de los operandos disponibles en los puertos de entrada *A* y *B* cuando el puerto *SUMAR* es colocado en alto; el resultado se obtiene en el puerto de salida *C* cuando el puerto *FIN SUMA* se pone en alto. Al interior del componente, el proceso de esta operación se realiza por medio de la máquina de estados de la Figura 4.

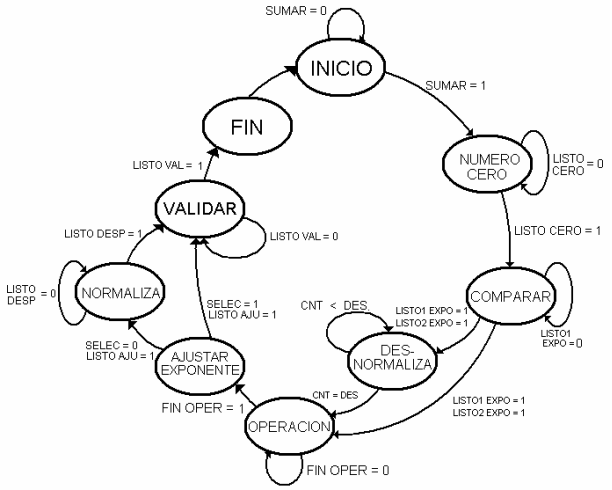

Figura 4 Máquina de Estado para el proceso de la suma en punto flotante.

El proceso de la suma comienza detectando cuando alguno de los operandos *A* o *B* es **CERO**, luego se **COMPARA** los exponentes y se determinar cual es el exponente mayor (este será el exponente parcial del resultado). Cuando los exponentes son diferentes la mantisa menor es **DESNORMALIZADA** desplazando hacia a la derecha |*expo1* - *expo2*| posiciones; con los operandos representados con el "mismo exponente" se procede a realizar la **OPERACIÓN** entre las mantisas, así: si el bit de signo de los dos operandos son iguales se suman las mantisas, si son diferentes se restan las mantisas. Cuando la mantisa resultante se encuentra desnormalizada se determina la cantidad de posiciones en que se debe desplazar para **AJUSTAR** el **EXPONENTE** y poder realizar la **NORMALIZACIÓN** de la mantisa. Por último, en el caso en que los bits del exponente y la mantisa sean todos ceros, el signo del resultado será cero, esto con el fin de **VALIDAR** la única representación del cero (+0). Terminado el proceso, se publica el resultado en el puerto *C* colocando en alto el puerto *FIN_SUMA* (estado **FIN**), y la máquina queda en espera de un nuevo proceso (estado **INICIO**).

3.1.2 MULTIPLICACIÓN EN PUNTO FLOTANTE

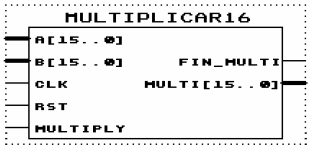

Figura 5. Componente en VHDL para realizar la multiplicación en punto flotante (bajo el estándar IEEE 754 [9, 11]).

El proceso de la multiplicación se puede ilustrar por medio de la máquina de estados de la Figura 6. La máquina pasa de su estado de reposo o de **INICIO** al estado **OPERACIÓN** con la puesta en alto del puerto *MULTIPLY*, en el estado **OPERACIÓN** se obtiene un resultado parcial de la multiplicación así:

- El signo es el resultado de realizar la or-exclusiva entre los bits de signo de los operandos *A* y *B*.
- El exponente parcial, corresponde a la suma de los exponentes; como los exponentes se encuentran sesgados, al realizar esta suma, se está sumando dos veces el sesgo, por lo cual se debe de restar una vez para obtener la representación adecuada del exponente.
- La mantisa parcial del resultado se obtiene por la multiplicación de las mantisas de los operandos.

Si terminadas las actividades en el estado **OPERACIÓN** la mantisa parcial no se encuentra normalizada, se pasa al estado **NORMALIZAR** donde se normaliza y se hace el respectivo ajuste del exponente. Como en la implementación anterior, el estado **VALIDAR** busca una única representación del cero (+0).

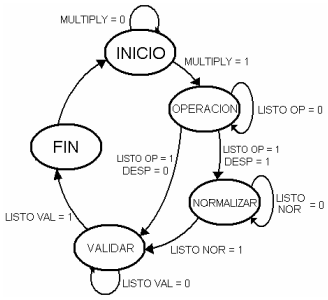

Figura 6. Máquina de Estados para el proceso de la multiplicación en punto flotante.

3.2 IMPLEMENTACIÓN DE LA SVM.

Para implementar en VHDL la máquina de vector soporte se propone el esquema por bloques de la Figura 7. Este esquema por bloques, hace que la implementación sea flexible para realizar trabajos futuros, en los cuales se pueda cambiar el tipo de kernel en la función de decisión.

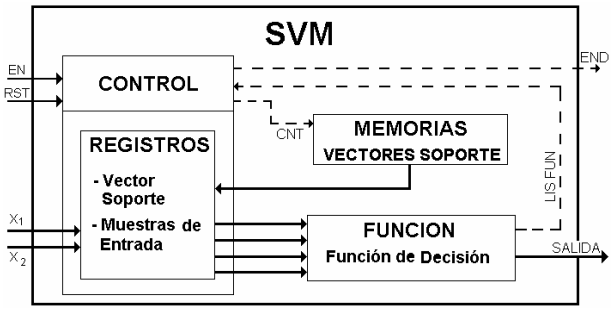

Figura 7. Implementación por bloques de la Máquina de Vector Soporte.

El bloque FUNCIÓN es donde se llevan a cabo todas las operaciones aritméticas de la función de decisión (Ecuación (11)). Este bloque se desarrolla en cinco etapas como lo muestra la Figura 8.

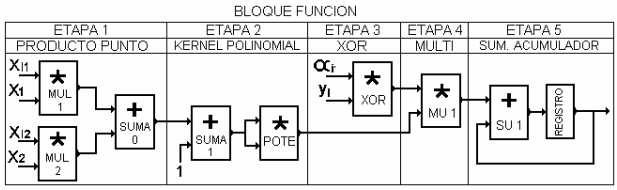

Figura 8 Esquema por Etapas para la implementación de la función de decisión (Ecuación (11)).

Es de aclarar que la Etapa 3, corresponde a la multiplicación entre el multiplicador de lagrange a_i y la etiqueta y_i, como las etiquetas siempre son +1 y -1 se pueden representar solo por el bit de signo, por lo cual esta operación se reduce a la or-exclusiva entre la etiqueta y el bit de signo del multiplicador de lagrange (el exponente y la mantisa del resultado son los del a_i).

Aparentemente, todas las operaciones aritméticas se ejecutan en forma secuencial, pero en la implementación las Etapas 1 y 4 se realizan en forma paralela, así que las multiplicaciones nombradas como *MUL 1*, *MUL2* y "*XOR*" se inician con la misma señal de habilitación; las demás operaciones se desarrollan en forma secuencial debido al flujo normal de los datos en la función de decisión.

El bloque MEMORIAS corresponde a 4 memorias de tipo ROM en donde se almacenan las etiquetas, los multiplicadores de Lagrange y las dos componentes vectoriales de los vectores soporte, en la primera posición se almacena la cantidad de vectores soporte.

El bloque CONTROL utiliza la máquina de estados de la Figura 9 para proporcionarle al bloque función cada uno de los datos almacenados en las memorias. La máquina en su estado de reposo o de INICIO, comienza el contador de direcciones de memoria en ceros, de forma que el primer dato que se lee en memoria es la cantidad de vectores soporte.

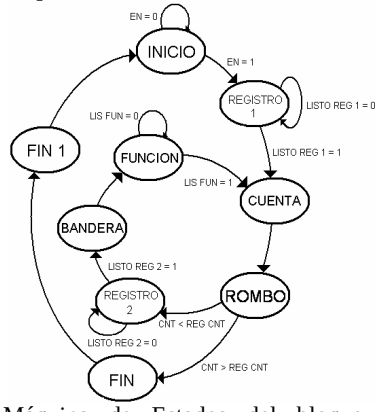

Figura 9. Máquina de Estados del bloque CONTROL, funcionamiento del esquema por bloques de la Figura 7.

Cuando el puerto EN se coloca en alto se habilita un primer bloque de registros (estado **REGISTRO 1**) para almacenar la cantidad de vectores soporte, las componentes vectoriales X1 y X2 de la muestra de

entrada. Luego, se incrementa en uno el contador de direcciones (estado **CUENTA**) para acceder a la siguiente posición de memoria, si el contador de direcciones no es superior a la cantidad de vectores soporte (estado **ROMBO**), se habilita un segundo bloque de registros (estado **REGISTRO 2**) donde se almacena cada uno de los datos de los vectores soporte que se encuentran en memoria. Con todos los datos almacenados en los registros se puede evaluar la función de decisión, por lo cual se genera un pulso de habilitación (estado **BANDERA**) para poner en operación el bloque **FUNCIÓN**. Terminadas todas las operaciones aritméticas del bloque función se incrementa de nuevo el contador de direcciones (estado **CUENTA**) repitiéndose de nuevo el proceso. Cuando el contador de direcciones es mayor a la cantidad de vectores soporte el proceso termina publicando el resultado en el puerto SALIDA (estado **FIN**), por último se pone en alto el puerto END (estado **FIN 1**) indicando que ha terminado todo el proceso.

4. PRUEBAS Y RESULTADOS

Para determinar el desempeño de la arquitectura digital de la SVM, su función de decisión también se implementó en Matlab buscando una referencia con la que se pudiera comparar los resultados entregados por la FPGA. Las pruebas preliminares en la implementación, se realizaron con los datos de la Tabla 1 y multiplicadores de Lagrange iguales a 0.125.

El primer grupo de muestras de entrada empleadas para realizar las pruebas, corresponden al conjunto de muestras cuyas componentes vectoriales se encuentran acotadas entre los rangos -5 y +5. En la Figura 10 se puede apreciar el espacio de clasificación generado para este primer grupo de muestras, en este grupo se evaluaron los cuatro vectores soporte de la Tabla 1, los cuales se clasificaron correctamente. Al evaluar en Matlab este primer conjunto de muestras se obtuvo el mismo espacio de clasificación, además la magnitud de los resultados antes de aplicar la función *signo* fue el mismo.

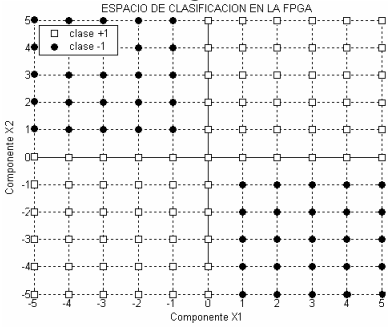

Figura 10. Espacio de clasificación obtenido por la FPGA: Espacio inicial a escala 1:1, y subespacio a escala 1:0.1

El segundo grupo de muestras corresponde al subespacio acotado por los rangos -0.5 y +0.5; al igual que con el primer grupo de muestras, el espacio de clasificación generado por las dos implementaciones tiene la misma distribución de la Figura 10. Al estudiar la magnitud de la respuesta en la clasificación de cada un de las muestras el promedio del porcentaje de Error Absoluto Fraccional (EAF)[1] es igual a 1.16%.

Para evaluar al sistema en una región más detallada entre las clases se utilizó un tercer grupo de muestras, acotado por los rangos -0.1 a 0.0 en la *componente vectorial X1* y +0.1 a 0.0 en la *componente vectorial X2*. En la Figura 11 se aprecia el espacio de clasificación para este último conjunto de muestras. Dicho espacio no presenta el resultado esperado, donde el EAF se incrementa al 22.51%.

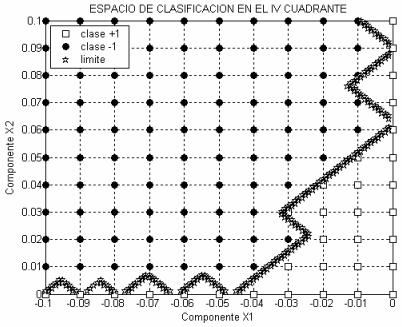

Figura 11. Espacio de clasificación obtenido por la FPGA: subespacio a escala 1:0.01, con limite entre clases a escala 1:0.001

5. TRABAJO FUTURO

Como la arquitectura digital para la SVM hasta el momento se encuentra restringida para clasificadores con Kernel Polinomial de segundo orden, se propone implementar el siguiente Kernel Gaussiano

$$K(x_i,x)=\exp\left[\frac{-\|x_i-x\|^2}{2s^2}\right] \qquad (12)$$

Pero, para poder implementar este Kenel, la función y=*exp(x)* se debe expresar por medio de su aproximación matemática

$$\exp(A)=\sum_{m=0}^{\infty}\frac{A^m}{m!} \qquad (13)$$

Para implementar en la FPGA la función de decisión de la SVM con la aproximación matemática del Kernel Gaussiano, se puede conservar la estructura por bloques de la Figura 7, realizando las siguientes modificaciones:

- Se adiciona un *REGISTRO* en el cual el usuario pueda ingresar el valor de la desviación estándar (s).
- La máquina de estados del bloque *CONTROL*, habilitaría el registro del ítem anterior en su estado *REGISTRO 1*.
- Las etapas 1 y 2 de la Figura 8, que corresponde a la implementación de la función de decisión se modificaría por las etapas 1 y 2 de la Figura 12.

[1] Error Absoluto Fraccional (EAF) para esta aplicación se define como $abs[\,{}^{(resp.\,MATLAB-resp.\,FPGA)}/_{resp.\,MATLAB}\,]\times 100\%$

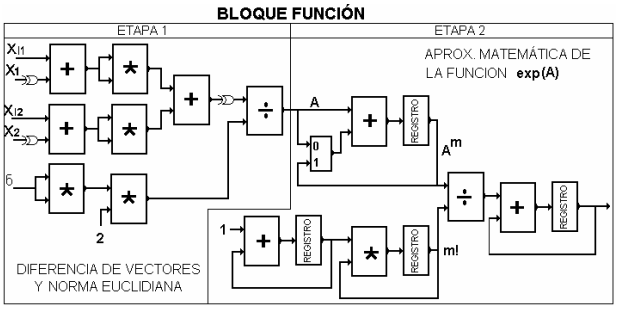

Figura 12 Implementación para el Kernel Gaussiano; modificación de las Etapas 1 y 2 del Bloque *FUNCIÓN* de la Figura 8.

Antes de realizar la implementación del Kernel Gaussiano, primero se debe diseñar la arquitectura digital para la división en punto flotante; como se aprecia en la Figura 12, son necesarias dos divisiones. Por otra parte, para realizar la resta, primero se hace la XOR lógica entre el bit de signo del sustraendo y un nivel alto (1) obteniéndose la negación, con este número resultante se procede a realizar la suma.

6. CONCLUSIONES Y DISCUSIONES

La versatilidad en la reconfiguración de los dispositivos lógicos de la FPGA permite que una arquitectura digital pueda ser reutilizable dentro de otras arquitecturas, esto hace que las implementaciones se puedan realizar en forma modular y flexible.

Con el diseño por bloques de la función de decisión de la SVM se logra reducir el tiempo total de cálculo, debido a que se esta aprovechando al máximo la capacidad de procesamiento paralelo que proporcionan las FPGA.

El estándar IEEE 754 adaptado a 16 bits permite representar cierta cantidad de números decimales, pudiéndose realizar las cuatro operaciones aritméticas entre los mismo. Pero una de las limitaciones es que los números enteros menores a 512 tienen una única representación en punto flotante, pero en los números mayores, no se conserva esta misma unicidad en la representación porque varios números enteros se pueden representar con 1 en punto flotante. Si la magnitud de las componentes vectoriales fuese superior a 512 el porcentaje de error para la implementación de la FPGA podría ser considerablemente elevado.

El aumento en EAF del tercer grupo de muestras no se debe al clasificador, sino a las limitaciones de hardware para poder implementar la representación de 64 bits como lo es un procesador actual. Este problema se observa cuando números muy pequeños son sumados o restados con un número entero; donde el resultado llega a ser el mismo número entero. En la suma cuando se desnormaliza el número menor, se puede correr el riesgo de hacerse cero, por lo tanto el número menor no tiene incidencia sobre el mayor.

7. BIBLIOGRAFÍA

[1] ALBA, José L. Castro. Curso de Doctorado: Decisión, Estimación y Clasificación. Universidad de Vigo. [http://www.gts.tsc.uvigo.es/~jalba/doctorado/]

[2] ALTERA, C. User Guide [http://altera.com]. October 2001.

[3] GUNN, S., Support Vector Machines for Classification and Regression, ISIS Technical Report, 1998.

[4] BARTLETT, Peter L. and Smola Alexander J. Advances in Large Margin Classifiers. MIT Press. 2000.

[5] BOYD, Stephen and Vandenberghe Lieven. Convex Optimization. Cambridge University Press. 2004

[6] BURGES, Christopher J. C. A Tutorial on Support Vector Machines for Pattern Recognition. Data Mining and Knowledge Discovery. vol. 2, no. 2, 1998.

[7] FLORES José D. Método de Multiplicadores de Lagrange: Una Versión Animada. The University of South Dakota. Noviembre 2004.

[8] HANDLEN, J. O. and Furman, M. D. Rapid Prototyping of Digital Systems Tutorial Approach. 2003.

[9] KAHAN, W. Lecture Notes on the Status of IEEE Standard 754 for Binary Floating-Point Arithmetic. University of California. Octubre 1997.

[10] PLATT, John C. Fast training of support vector machines using sequential minimal optimization, In Advances in Kernel Methods. MIT Press. 1998.

[11] SHAABAN, Muhammad E. Representation of Floating Point Numbers in Single Precision Single Precision IEEE 754 Standard. 2000.

[12] SMOLA, Alexander J. Lerning with Kernels. GMD. vol. 1, no. 25, 1998.

[13] VAPNIK, Vladimir. Statistical Learning Theory. Wiley, New York, 1998.