Instituto Tecnológico de Costa Rica

Escuela de Ingeniería Electrónica

Implementación en hardware del Sistema de Reconocimiento de Patrones Acústicos (SiRPA)

Informe de Proyecto de Graduación para optar por el título de

Ingeniero en Electrónica con el grado académico de Licenciatura

Luis Adolfo Alfaro Hidalgo

Cartago, 26 de junio de 2013

Declaro que el presente Proyecto de Graduación ha sido realizado enteramente por mi persona, utilizando y aplicando literatura referente al tema e introduciendo conocimientos propios.

En los casos en que he utilizado bibliografía he procedido a indicar las fuentes mediante las respectivas citas bibliográficas. En consecuencia, asumo la responsabilidad total por el trabajo de graduación realizado y por el contenido del correspondiente informe final.

Luis Adolfo Alfaro H

Luis Adolfo Alfaro Hidalgo

Cartago, 26 de junio de 2013

Céd: 206670982

Instituto Tecnológico de Costa Rica
Escuela de Ingeniería Electrónica
Proyecto de Graduación
Tribunal Evaluador

Proyecto de Graduación defendido ante el presente Tribunal Evaluador como requisito para optar por el título de Ingeniero en Electrónica con el grado académico de Licenciatura, del Instituto Tecnológico de Costa Rica.

Miembros del Tribunal

Dr. José Pablo Alvarado Moya
Profesor Lector

M.Sc. Roberto Pereira Arroyo
Profesor Lector

Dr. Alfonso Chacón Rodríguez
Profesor Asesor

Los miembros de este Tribunal dan fe de que el presente trabajo de graduación ha sido aprobado y cumple con las normas establecidas por la Escuela de Ingeniería Electrónica.

Cartago, 26 de junio, 2013.

Resumen

En Costa Rica existen un aproximado de 170 áreas protegidas, que representan un 26% de la superficie continental y 17% de la superficie marina del país, según un informe provisto por entidades gubernamentales. Los bienes sociales, ecológicos y económicos que generan las áreas de conservación se estiman en cientos de millones de colones, los cuales son generados por actividades como turismo y generación hidroeléctrica.

Por la razón anterior es que la detección de actividades ilegales como la caza y la tala de árboles dentro de las zonas protegidas ha tomado importancia. La detección de tales eventos puede realizarse con el análisis de patrones acústicos, que para fines prácticos es realizado en redes de sensores, que además de analizar, se encargan de capturar el audio y de transmitir las alertas a los nodos principales.

Una parte fundamental de la red de sensores inalámbrica es el sistema que se encarga de analizar el audio capturado del bosque. Para lo anterior se debe realizar un proceso de captura, acondicionamiento de la señal, extracción de características, cálculo de probabilidades y finalmente tomar una decisión. En el marco de un proyecto tan extenso, el Sistema de Reconocimiento de Patrones Acústicos (SiRPA) es el encargado de las fases de extracción de características, cálculo de probabilidades y la toma de decisiones.

En este documento se detalla el desarrollo del hardware del SiRPA, así como los módulos propuestos para completar su funcionamiento, denominados como reductor de dimensiones, generador de símbolos y unidad de modelos ocultos.

Palabras clave: SiRPA, distancia L1 o Manhattan, reducción de dimensiones, generador de símbolos, modelos ocultos de Markov, FPGA, VLSI digital

Abstract

In Costa Rica, according to governmental studies, there are 170 protected areas, which represent 26% of the continental surface and 17% of the ocean surface of the country. Those reserves not only produce social and ecological benefits, but economical too - the economical retail generated by activities like tourism and electricity generation are estimated in hundreds of millions of colons-.

That is why the detection of illegal activities, such as poaching or unauthorized logging, has become a subject of interesting in the electronics area. The detection of this kind of activities can be done using digital processing of acoustic signals. These signals can be sensed by a wireless sensor network, that not only have to capture the sound, but process and transmit decisions based on the sounds.

A fundamental subsystem of the wireless sensor network is the one who analyzes the audio signal of the forest. That system is composed of several stages like: capturing the forest signals, gain control process, characteristics' extraction, probability computation and a decision making stage. Most of this tasks are assign to the Acoustic Pattern Recognition System (SiRPA by its acronym in Spanish) which implements the characteristics extraction, probability computation and the decision-making stage.

The hardware implementation of a dimension reduction subsystem, a symbol generator and a hidden Markov models unit, either programmable or using standard cells from a commercial CMOS process is here documented.

Keywords: SiRPA, L1 o Manhattan distance, dimension reduction, symbols generator, hidden Markov models, FPGA, Digital VLSI

a Dios por darme fuerzas y mis padres por su apoyo

Agradecimientos

Primeramente le doy gracias a Dios por ser mi infinita fuente de fuerzas y bendiciones, ya que sin Él nada de esto pudo ser llevado a cabo.

A mis familiares y de manera muy especial a mis padres, cuya ayuda y apoyo fue fundamental en este proceso.

A mi novia, cuyo cariño y comprensión estuvieron conmigo durante todo este largo proceso que tuvo como fruto este documento.

Al personal docente y administrativo del ITCR, gracias por su trabajo y dedicación, ya que intervinieron en mi formación durante estos años.

Finalmente a todos los que de una u otra manera contribuyeron durante mi formación académica y personal, sin cuya ayuda no hubiese sido posible este proyecto.

Luis Adolfo Alfaro Hidalgo

Cartago, 26 de junio de 2013

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Costa Rica cuenta con un aproximado de 170 áreas protegidas, las cuales representan un 26% de la superficie continental y 17% de la superficie marina del país según el informe del SINAC [12]. Además de los bienes sociales y ecológicos que generan las áreas de conservación, en el 2009 se estima que las mismas generaron 778.148 millones de colones por actividades como turismo y generación hidroeléctrica. Por lo tanto, organismos como el SINAC recomiendan complementar las políticas de conservación con las de desarrollo.

Sin embargo, las zonas de conservación cuentan con retos importantes debido al poco personal dedicado a la vigilancia, el irrespeto de la población a la legislación y los insuficientes recursos destinados al fortalecimiento de las mismas. Debido a esto, se han tratado de implementar diversas soluciones para monitorizar las regiones y mejorar el control sobre las áreas de conservación. Una solución en particular ha sido las redes de sensores inalámbricas para monitorizar las zonas protegidas.

Las redes de sensores inalámbricas son una solución ya que son capaces de medir parámetros, almacenarlos, procesarlos y enviarlos entre sí, y pueden desplegarse en grandes cantidades y de manera densa. Todas estas características convierten a las redes de sensores en una opción viable, siempre y cuando sean de muy baho consumo, pues este es requerimiento fundamental de estos sistemas, porque deben ser capaces de operar por largos periodos de tiempo y sin ningún tipo de mantenimiento [13].

La Escuela de Ingeniería Electrónica del Instituto Tecnológico de Costa Rica ha estado desarrollando el proyecto denominado: "Sistema Electrónico integrado en chip (SoC) para el reconocimiento de patrones de disparos y motosierras en una red inalámbrica de sensores para la protección ambiental", el cual propone el desarrollo de una red de sensores inalámbrica para la protección ambiental, la cual sea capaz de detectar eventos como: motosierras y disparos, a partir del análisis de patrones de audio.

Para realizar la detección de eventos se desarrolla el Sistema de Reconocimiento de Patrones Acústicos (SiRPA), diagrama general en la figura 1.1. En la misma se pueden apreciar las distintas etapas del funcionamiento del sistema de procesamiento del audio del bosque.

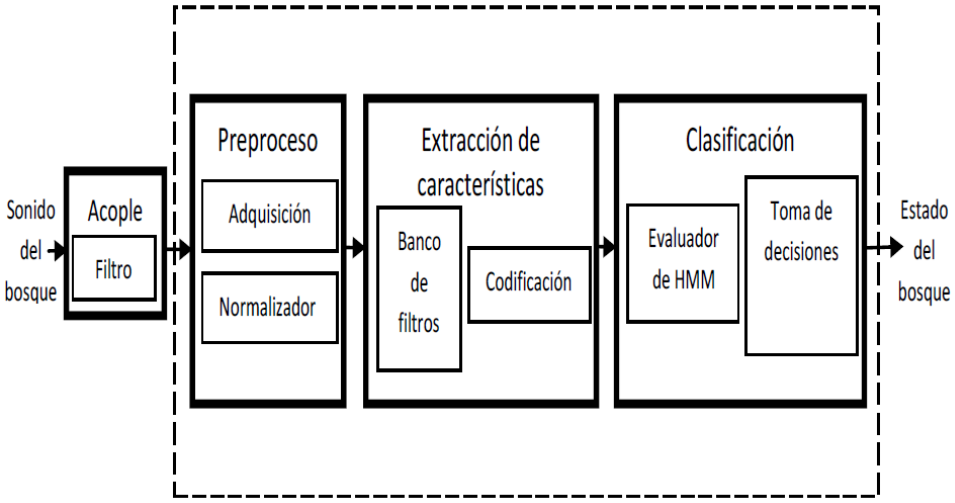

Figura 1.1: Diagrama de funcionamiento general de SiRPA. Tomado de [10]

Las etapas de acople y de preproceso se encargan de acondicionar la señal de entrada, de manera que ésta se encuentre entre los límites que le permiten al sistema procesar la información.

La etapa de extracción de características está compuesta por una fase que se encarga de separar la señal de entrada en distintas bandas de frecuencia (banco de filtros), de manera que sea posible extraer las características tiempo-frecuencia de la señal para ser procesada. En el procedimiento de codificación, el sistema determina valores discretos que representan la información de entrada, los cuales son empleados en la sección final del proceso.

En la última etapa de procesamiento, se realiza la clasificación. En la clasificación, se calculan las probabilidades de que la señal de audio represente el bosque, una motosierra o un disparo y finalmente, estas probabilidades son evaluadas en el módulo de toma de decisiones, el cual notifica al sistema principal cuál modelo representa la señal de entrada.

En el presente documento, se detalla el desarrollo de la implementación en hardware del SiRPA, así como el manejo de datos, los procedimientos que se ejecutan, y los resultados obtenidos con el diseño propuesto.

1.1 Trabajos previos

En el proyecto SiRPA, se han desarrollado diversos trabajos y proyectos, tanto de aplicación como de investigación, los cuales se han centrado en diversos problemas que se han resuelto previo al diseño del chip. A continuación se presentan los trabajos fundamentales que cimentaron las bases de este proyecto.

En [10] se desarrolló una primera implementación del SiRPA, orientada unicamente a

FPGA y no era reconfigurable. Aún así, la unidad de banco de filtros se rescató de esta primera aproximación, y se usaron como punto de partida los procesos propuestos de extracción de características y clasificación usando modelos ocultos de Markov.

En [1] se realizó una síntesis preliminar del diseño de hardware (del trabajo de [10]) para evaluar el consumo de potencia preliminar, el área física esperada, el consumo de potencia esperado, y comprobar tanto el funcionamiento de las herramientas como de la biblioteca definitiva de fabricación que se usará.

De [11] se obtienen los resultados del diseño y desarrollo de hardware de un módulo de reducción de dimensiones autogenerado por código en C. En el mismo se presentan los resultados de dos circuitos, en los que se emplean distintos métodos de entrenamiento para los coeficientes del sistema. Además se muestra el efecto de la reducción de dimensiones y la pérdida de información producto del proceso.

Finalmente en [4] se desarrolló un software para el entrenamiento de los coeficientes que emplea el sistema. Es decir, a partir de un entrenamiento desarrollado en un computador, y empleando distintos algoritmos, se calculan los datos que utilizan la fase de extracción de características y la de clasificación para hacer el procesamiento de datos necesario para identificar el bosque, las motosierras o los disparos.

1.2 Objetivos y estructura

El objetivo principal del desarrollo de la implementación en hardware del SiRPA es lograr un diseño que permita fabricar el chip del sistema, ya que únicamente con un ASIC se pueden alcanzar los requerimientos del sistema.

Al iniciar el proyecto, solo se contaba con una implementación verificada en simulación del banco de filtros. El resto de las etapas no estaban codificadas de manera correcta o al menos estándar -códigos que generaban exceso de errores y advertencias en los procesos de síntesis tanto para FPGA como para celdas estándar-. Además, no se estaba seguro del funcionamiento correcto de algunas de estas etapas, y el consumo de recursos previsto en algunas era prohibitivo. Ante esta situación, se prefirió re-codificar los módulos y verificarlos de nuevo contra la versión escrita en C e implementada en un sistema empotrado. Estas unidades deberían luego verificarse tanto en simulación como su implementación sobre FPGA, utilizando parámetros post-entrenamiento, antes de trasladarse a las herramientas finales de implementación en circuito integrado. Es claro que ello exigiría un código correcto y estándar para las herramientas.

Por último se decidió incorporar al circuito final la capacidad de reprogramar sus coeficientes o variables de trabajo, para volver al sistema adaptable a distintos ambientes o sonidos, ahora que solo se requería un nuevo reentrenamiento *off-line* de los algoritmos para generar nuevos coeficientes del sistema.

Este trabajo se encuentra dividido en diferentes secciones con el fin de facilitar su

lectura y comprensión. En el capítulo II se encuentran los fundamentos teóricos básicos para entender el funcionamiento de cada uno de los sistemas desarrollados. En el capítulo tres se encuentran los detalles del desarrollo del hardware, así como su funcionamiento y formato de los datos. En el capítulo cuatro se detallan los resultados experimentales de las pruebas y simulación del circuito diseñado. Finalmente, en el capítulo cinco se establecen las conclusiones y recomendaciones que se obtuvieron como fruto de este trabajo.

Capítulo 2

Marco teórico

En el presente capítulo se muestran las bases teóricas necesarias para comprender el desarrollo del documento, así como los procesos que realizan los distintos sistemas, el manejo de los datos y finalmente los resultados obtenidos. En el mismo se encuentran los fundamentos relacionados con: formato punto fijo, formato punto flotante, combinación lineal, distancia Manhattan (o L1) y modelos ocultos de Markov.

2.1 Formato punto fijo

El formato punto fijo es una manera de representar y manejar los datos digitales de un sistema, en la misma se define la cantidad de bits que representan los datos y luego, se determina cuál es la cantidad de bit más adecuada para representar la parte entera de los datos y la parte decimal de los mismos, además el bit más significativo se asocia al signo del dato.

Los bits que se asignan para representar la parte entera del valor decimal que representan son llamados *bits de magnitud* [3]. Por su parte, los bits de la sección derecha del punto decimal son denominados *bits fraccionales*.

Para calcular el rango de valores representables en punto fijo, con m bits asignados a la parte entera y f a la parte fraccionaria, se emplea la siguiente ecuación:

$$\left[-2^m : 2^m - \frac{1}{2^f} \right] \tag{2.1}$$

Además la resolución del formato está dado por:

$$\frac{1}{2^f} \tag{2.2}$$

Para el presente proyecto, los datos en punto fijo son de dieciséis bits de longitud, de los cuales se asignaron dos bits a la parte entera y los asignados para representar la parte fraccionaria corresponden a trece. Por lo que el rango de valores representables es [-4:3.9998779296875].

2.1.1 Suma y resta en formato punto fijo

Las operaciones de suma y resta de datos en formato punto fijo se realizan de igual manera que las operaciones de datos en formato binario para número enteros, es decir se aplican las mismas consideraciones. Sin embargo, es necesario que ambos operandos cuenten con la misma cantidad de bits asignados a la parte entera y fraccionaria.

Al sumar o restar dos datos en formato punto fijo, el resultado produce un dato con la misma cantidad de bits, sin embargo puede ocurrir desbordamiento del resultado (si el resultado de la suma excede el máximo valor representable o el resultado de la resta es menor al mínimo valor representable), lo cual produce un valor en el rango representable, pero con un resultado incorrecto.

2.1.2 Multiplicación en formato punto fijo

En el caso de la multiplicación de datos en punto fijo, se debe tener en consideración que al igual que en la multiplicación de datos en decimal, el resultado tiene el doble de longitud que los operandos, es decir, el resultado tiene una longitud del doble de los datos que maneja el sistema.

En la figura 2.1 se muestra un ejemplo en concreto del resultado de una operación de multiplicación en punto fijo. En el ejemplo, se emplean ocho bits para la parte fraccionaria y ocho para la parte entera. En el procedimiento del ejemplo, se descartan los ocho bits menos significativos que corresponde a la sección de menor peso de la parte fraccionaria y se toman los siguientes dieciséis bits (indicados como resultado válido en la figura 2.1).

Figura 2.1: Ejemplo de multiplicación en punto fijo. Tomado de [7].

El proceso de truncar el resultado es inevitable para mantener el formato de los datos que se emplean en el sistema, y es evidente que los valores de los parámetros m y f repercuten directamente en la estabilidad numérica y los resultados de las operaciones en el sistema.

2.2 Formato punto flotante

El formato punto flotante es una representación numérica más moderna que la representación punto fijo, de manera que resulta en un formato de datos que proporciona la exactitud suficiente para la mayoría de aplicaciones digitales. Los datos en punto flotante involucran un número entero (mantisa) que se multiplica con una base (en este caso es dos) y la base se encuentra elevada a un exponente, tal como se muestra a continuación:

$$a = m \cdot b^e \tag{2.3}$$

Existen un compendio de estándares que determinan como se designan los bits del formato, así como la longitud total del mismo y los parámetros que se emplean en las operaciones aritméticas. En este caso, se sigue el estándar de treinta y dos bits, conocido como el formato de precisión simple (definido como IEEE 754).

En el estándar IEEE 754 se define que el bit más significativo corresponde al signo, los siguientes ocho bits son el exponente del dato y finalmente los restantes veintitrés se asocian a la mantisa, tal como se muestra en la figura 2.2.

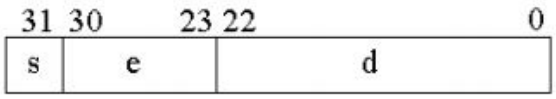

Figura 2.2: Estándar del formato punto flotante: precisión simple. Tomado de [3].

Las unidades que resuelven las operaciones de punto flotante (o FPU por su siglas en inglés) son sistemas diseñados para un limitado número de funciones. En el caso del presente proyecto, se requirió de la suma, la resta, la multiplicación, división y comparación de datos en punto flotante.

2.2.1 Unidad de aritmética punto flotante

La estructura básica de una unidad punto flotante se muestra en la figura 2.3. Los operandos de la unidad son denominados **opa** y **opb**, la señal de control que determina cuál operación se lleva a cabo es la denominada **fpu_op**. El resultado de la operación es la señal **result**. La unidad cuenta con distintas banderas para los casos en los que el resultado es cero o infinito, los operandos no sean números o el divisor sea cero, entre otros.

En el diagrama de la figura 2.3 se aprecian algunos módulos básicos en las operaciones de punto flotante, como lo son los bloques de pre normalización de los datos que se encargan de verificar la validez de los datos de entrada; a la salida los bloques de post

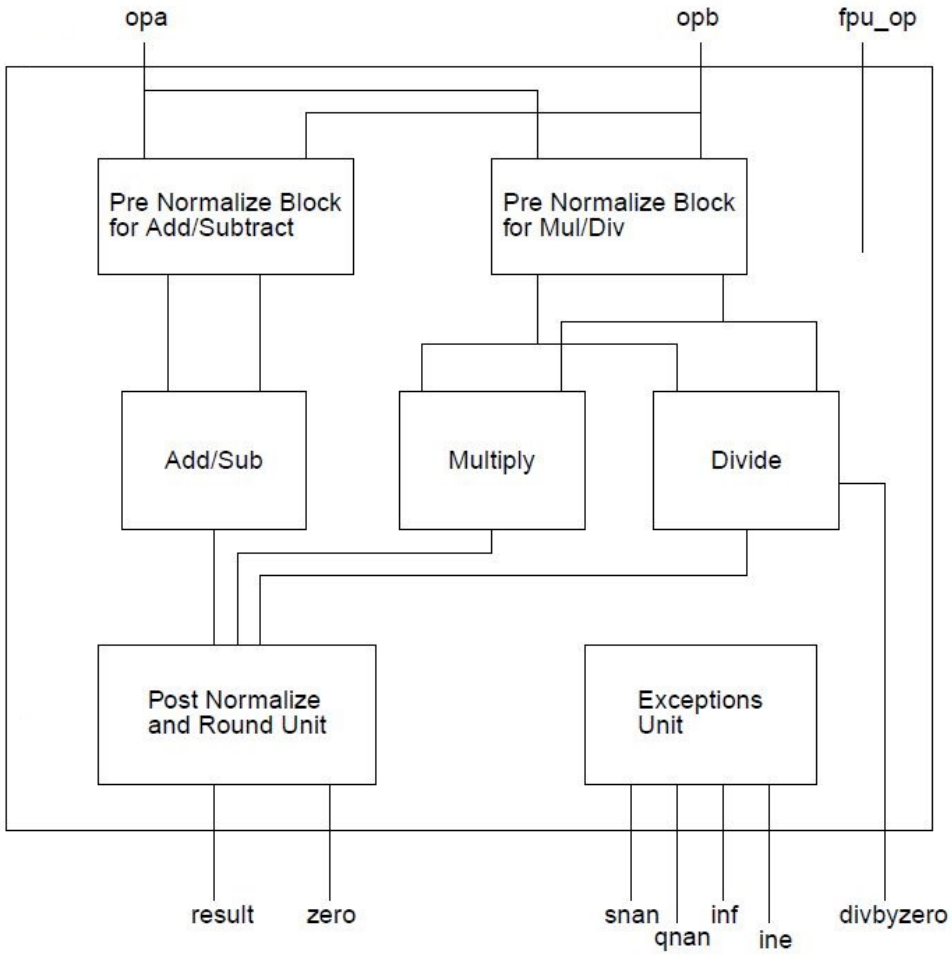

Figura 2.3: Diagrama de unidad punto flotante. Tomado de [14].

normalización y redondeo preparan los datos para garantizar que se encuentren en el rango representable. Si en el proceso ocurren errores, la unidad de excepciones se encarga de indicarlos al sistema.

En el caso particular de SiRPA, la FPU es totalmente combinacional.

2.3 Transformación lineal

Según [2], transformación lineal se define como: "Sean V y W espacios vectoriales. Una **transformación lineal** de V y W es una función T: $V \to W$, que satisface las siguientes propiedades":

$$1. T(u+v) = T(u) + T(v), \forall u, v \in V \tag{2.4}$$

$$2. T(kv) = kT(u), \forall u \in V, \forall k \in \mathbb{R} \tag{2.5}$$

Un caso particular de transformación lineal es la **transformación matricial**. La transformación V_A, una operación definida de $\mathbb{R}^n \rightarrow \mathbb{R}^m$, tal que $V_A = A \cdot W$. En la definición, A es una matriz de dimensión m x n .

Este tipo de transformación lineal es la realizada por el módulo reductor de dimensiones (ver sección 3.1).En este caso particular, n = 8 y m = 3; por lo tanto, la matriz A es de dimensión 8x3 y cuenta con veinticuatro coeficientes, cuya combinación lineal permite pasar de un espacio de entrada de ocho dimensiones a uno de tres dimensiones en la salida.

2.4 Distancia Manhattan o L_1

La distancia L_1 o distancia Manhattan es un método de cálculo que permite determinar que tan separados se encuentran dos puntos entre sí, sin emplear el módulo de vectores y de una manera simplificada. La distanciaL_1 se define como [8]:

$$d_{L1}(A, B) = \sum_{k=1}^{p} |b_k - a_k| \tag{2.6}$$

La distancia L_1 supone ventajas de cálculo ya que no emplea operaciones como elevar a potencias o calcular la raíz de los datos. Por esta razón, en el Módulo Generador de Símbolos se emplea la distancia L_1 para determinar cuál de los nodos se encuentra más cerca al punto de entrada.

2.5 Modelos ocultos de Markov (HMM)

Los modelos ocultos de Markov son una herramienta importante en el reconocimiento de patrones acústicos, que con el desarrollo de las técnicas de cálculo de los parámetros involucrados en el proceso, han sido aplicados a una gran variedad de problemas de reconocimiento de patrones acústicos.

Un HMM se define como $\lambda = (A,B,\pi)$, donde A, B y π son matrices que caracterizan el modelo, además se tiene una cadena de símbolos O, los posibles símbolos V, y la cantidad de estados de los modelos N [6]. Cada elemento está descrito como:

- T: longitud de la cadena de símbolos O.
- N: números de estados en el modelo.
- V: posibles símbolos del modelo.
- M: cantidad de elementos en V.
- $A = \{a_{ij}\} = P_r(q_j \text{ en t+1} \mid q_i \text{ en t})$. A es la matriz de coeficientes que caracterizan las transiciones del modelo de un estado a otro, con dimensiones NxN.

- $B = \{b_j(\mathrm{k})\}$, $b_j(\mathrm{k}) = P_r(v_k \text{ en t} \mid q_j \text{ en } t)$. La probabilidad de observar el símbolo v dado en el estado k, en el momento t, o sea, la matriz de observación del modelo que tiene dimensiones de MxN.
- $\pi = \{\pi_i\}$, $\pi_i = P_r(q_i \text{ en } t{=}1)$. La matriz de inicialización del modelo, con N coeficientes.

Una vez definido el modelo oculto de Markov, existen tres problemas que se necesitan resolver sobre el modelo:

- 1. Dada la cadena de observación $O = O_1, O_2, \ldots, O_T$ y el modelo $\lambda = (A,B,\pi)$, ¿cómo se calcula la probabilidad de que la cadena haya sido emitidad por el modelo λ?.
- 2. Con la cadena de observación $O = O_1, O_2, \ldots, O_T$, cómo se determina la secuencia de estados que mejor representa los símbolos de la cadena.
- 3. Cómo se ajustan los parámetros de $\lambda = (A,B,\pi)$ para optimizar el cálculo de $P_r(O \mid \lambda)$.

En el caso de SiRPA, los problemas de interés son el problema uno y el problema tres (este último resuelto en [4]). En el caso del problema uno, para calcular la probabilidad de que la cadena haya sido emitida por un modelo dado, se emplea el algoritmo *"hacia-adelante hacia-atrás"*.

2.5.1 "Algoritmo hacia-adelante hacia-atrás (forward-backward procedure)

Este procedimiento de manera extensa se encuentra definido en [9]. Se define la variable α_1 como:

$$\alpha_t(i) = P(O_1 O_2 ... O_t, q_t = S_i \mid \lambda). \tag{2.7}$$

La ecuación 2.7 es la probabilidad de que la cadena de observación parcial llamada α_t, haya sido generada por el modelo λ, en el tiempo t y estado S. Esta probabilidad se calcula aplicando tres reglas:

1. Inicialización:

$$\alpha_1(i) = \pi_i b_i(O_1), 1 \leq N. \tag{2.8}$$

2. Inducción:

$$\alpha_{t+1}(j) = \left[\sum_{j=1}^{N} \alpha_t(j) a_{ij} \right] b_j(O_{t+1}), 1 \leq t \leq T-1, 1 \leq j \leq N. \tag{2.9}$$

3. Finalización

$$P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_T(j). \tag{2.10}$$

En el este algoritmo, se requieren cálculos en el orden de $N(N+1)(T-1)+N$ multiplicaciones y $N(N-1)(T-1)$ sumas, lo cual es una ventaja si se compara con los demás algoritmos que se emplean para solucionar el problema 1, según se indica en [6]; por esta razón es que *The forward - backward procedure* es el empleado en el SiRPA para realizar los cálculos de probabilidades en los tres modelos.

Capítulo 3

Implementación en hardware de SiRPA

El Sistema de Reconocimiento de Patrones Acústicos (SiRPA), debe ser implementado en hardware para poder cumplir con las especificaciones del sistema (ver capítulo 1), por lo cuál, se propone el diagrama de entradas y salidas del sistema como se muestra en la figura 3.1

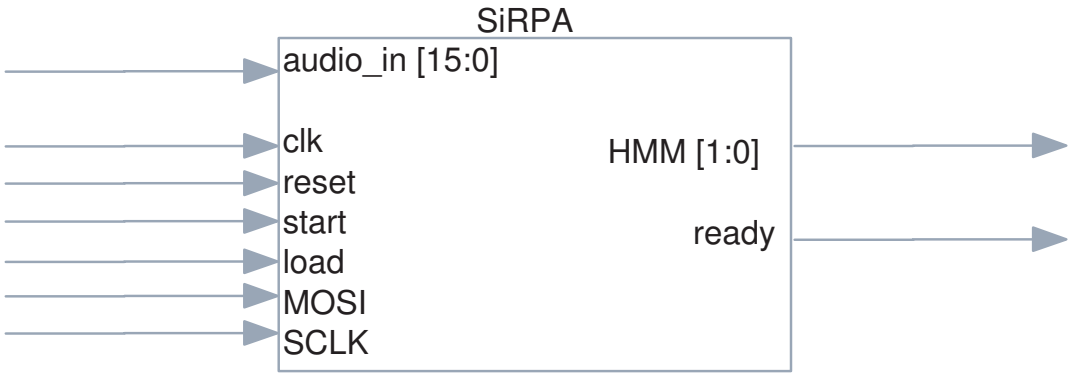

Figura 3.1: Diagrama de entradas y salidas del SiRPA

En el mismo, se muestran las señales de entrada:

- **audio_in:** es la señal digital de entrada del sistema, es decir los datos digitales que representan el audio que se va a analizar. Para representar a los mismos, se emplean dieciséis bits.
- **clk:** la señal de reloj para el SiRPA.
- **reset:** es la señal de reinicio del sistema. Para reiniciar, se debe colocar en alto la misma.
- **load:** con este puerto de entrada se le indica al sistema que se desea realizar una carga de coeficientes, es decir se va a reconfigurar los coeficientes del SiRPA.
- **MOSI:** es la salida de datos del módulo maestro del protocolo serial (SPI) y la entrada de los datos al SiRPA. Los datos que se introducen al SiRPA corresponden

a los coeficientes del sistema.

- **SCLK:** es la señal de reloj de los datos del protocolo serial (SPI).

Las señales de salidas del sistema:

- **HMM:** con esta señal de dos bits se indica que la muestra de audio representa con mayor probabilidad el bosque (con un cero), una motosierra (con un uno) o un disparo (con un dos).
- **ready:** cuando el sistema ha concluido de analizar el audio y ya identificó cual modelo es mejor representado por la muestra de audio, lo indica mediante esta señal al colocar en un valor alto.

Para poder llevar a cabo el procedimiento completo del SiRPA, éste se descompuso en distintos bloques, que ejecutan cada una de las funciones descritas en la figura 1.1. En la figura 3.2, se muestra el diagrama de bloques interno del sistema.

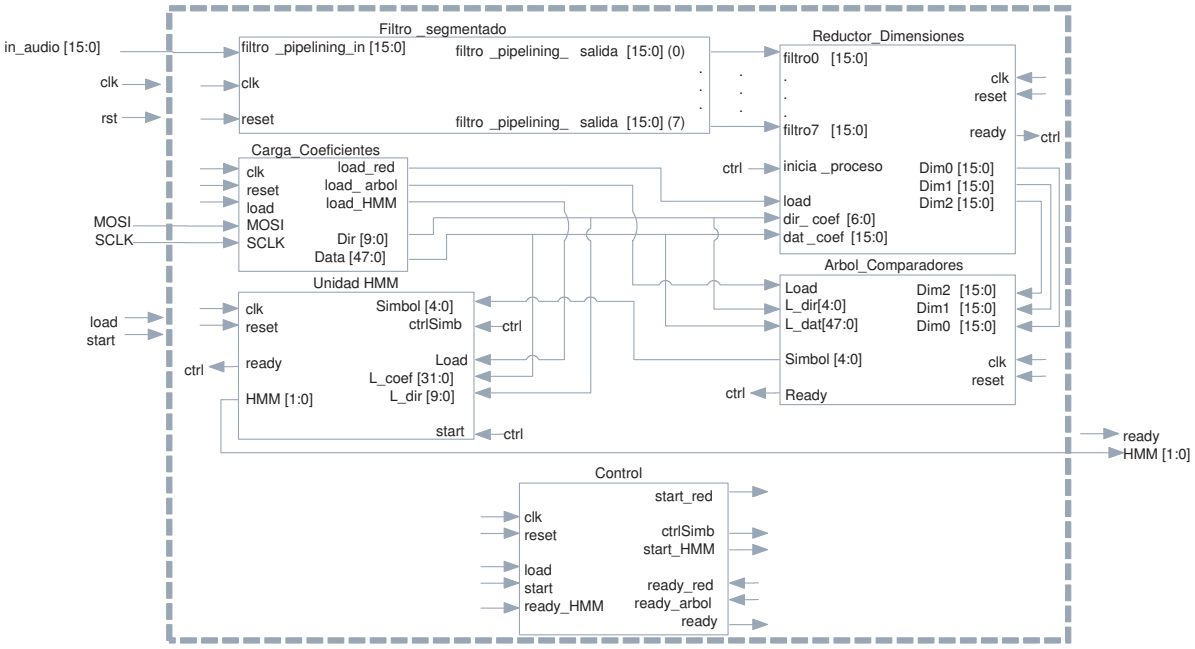

Figura 3.2: Diagrama de segundo nivel del SiRPA

En la figura 3.2, se muestra el módulo denominado filtro segmentado[10], cuya función principal es dividir la señal de entrada (el audio proveniente del exterior) en ocho bandas de frecuencia, para que las mismas sean procesadas por el módulo reductor de dimensiones.

En las siguientes secciones se detalla el desarrollo de los módulos de hardware correspondientes al reductor de dimensiones, el árbol de comparadores, la unidad de modelos ocultos de markov (unidad HMM), el control y el sistema de carga de coeficientes.

3.1 Reductor de Dimensiones

El módulo de reducción de dimensiones realiza una conversión de un espacio de entrada de ocho bandas de frecuencia a uno de salida de tres. Esta transformación se lleva a cabo a partir de una combinación lineal de los parámetros de entrada con los coeficientes del sistema (ver sección 2.3): los coeficientes son calculados por un software de entrenamiento [4]. Todos estos datos se encuentran en formato punto fijo (ver sección 2.1).

El diagrama de bloques del módulo reductor de dimensiones se muestra en la figura 3.3.

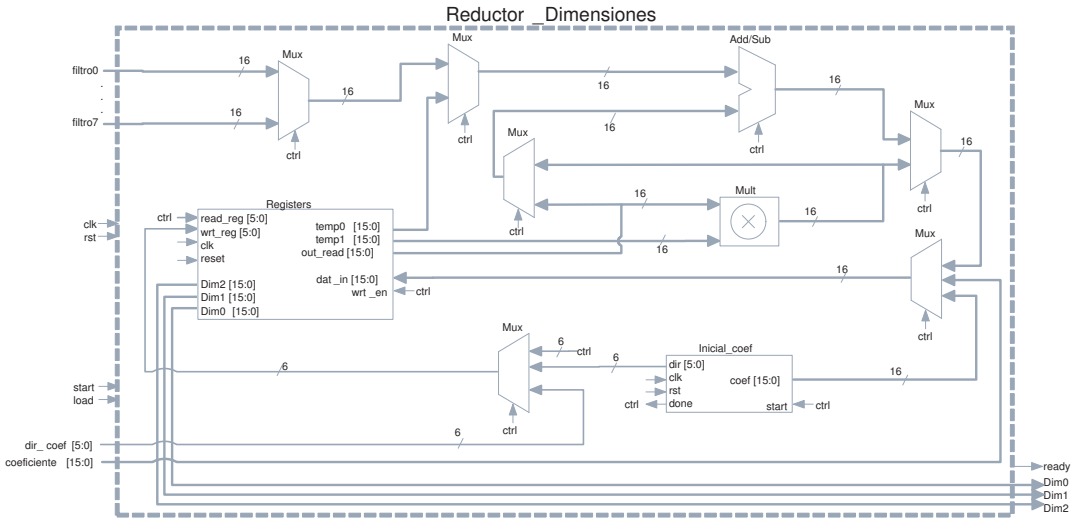

Figura 3.3: Diagrama del reductor de dimensiones

El bloque reductor de dimensiones recibe las señales de **filtro0** hasta **filtro7**, las cuales son la salida del banco de filtros. La señal **start** indica al bloque cuándo debe iniciar el proceso de reducción de dimensiones. La entrada **load** activa el módulo de carga de coeficientes, con lo que los datos que están en la señal **coeficiente** se pasan al banco de registros en la dirección dada por **dir_coef**.

Las salidas son el resultado de la reducción de dimensiones y están identificadas como **Dim0**, **Dim1** y **Dim2**. La señal **ready** en alto indica el estado del sistema, es decir que el módulo se encuentra en estado de inactividad y puede realizar la reducción de dimensiones o la carga de coeficientes del exterior; en el estado bajo, la señal indica que el módulo está procesando una reducción de dimensiones.

Después de un reinicio, el reductor procede a cargar la matriz de coeficientes, definida como **W**, en el banco de registros (bloque llamado ***"Registers"***), a través del ***"Inicia_coef"***. Los números y nombres de los registros se muestran en la tabla 3.1.

Una vez que se concluyó con la inicialización, el sistema se encuentra listo para realizar una carga externa de coeficientes, o un proceso de reducción de dimensiones.

Si se inicia el proceso de reducción de dimensiones, el sistema selecciona la primera señal

Dirección	Registro	Dirección	Registro	Dirección	Registro
0	Offset entrada 1	13	coeficiente w_{23}	26	coeficiente w_{71}
1	Offset entrada 2	14	coeficiente w_{31}	27	coeficiente w_{72}
2	Offset entrada 3	15	coeficiente w_{32}	28	coeficiente w_{73}
3	Offset entrada 4	16	coeficiente w_{33}	29	coeficiente w_{81}
4	Offset entrada 5	17	coeficiente w_{41}	30	coeficiente w_{82}
5	Offset entrada 6	18	coeficiente w_{42}	31	coeficiente w_{83}
6	Offset entrada 7	19	coeficiente w_{43}	32	Temporal 0
7	Offset entrada 8	20	coeficiente w_{51}	33	Temporal 1
8	coeficiente w_{11}	21	coeficiente w_{52}	34	Dimensión 2
9	coeficiente w_{12}	22	coeficiente w_{53}	35	Dimensión 1
10	coeficiente w_{13}	23	coeficiente w_{61}	36	Dimensión 0
11	coeficiente w_{21}	24	coeficiente w_{62}		
12	coeficiente w_{22}	25	coeficiente w_{63}		

Tabla 3.1: Banco de registros del reductor de dimensiones.

de entrada, le resta el valor de *offset 1* y lo almacena en el registro Temporal 0, posteriormente, multiplica el contenido del registro Temporal 0 con el coeficiente correspondiente (en este caso el w_{11}) y lo almacena en el Temporal 1. Luego, selecciona la entrada dos y le resta el *offset 2* y lo almacena en el Temporal 0. Después, multiplica el Temporal 0 con el coeficiente w_{21} y el resultado lo suma al Temporal 0, para finalmente almacenar la suma en el mismo Temporal 0. Este proceso se repite hasta que se hayan abarcado las ocho entradas; al final no se almacena la suma en el Temporal 0, sino que se almacena en el registro de **Dim0**.

El proceso descrito anteriormente se lleva a cabo de manera similar para calcular las salidas de **Dim1** y **Dim2**, pero se emplean los coeficientes particulares.

Si se procede a realizar una carga de coeficientes externa para reconfigurar el sistema, la entrada **load** se debe colocar en alto, así, el sistema habilita la escritura en el banco de registros y procede a escribir los datos que se tienen en la señal **coeficiente** en la dirección **dir_coef**.

3.2 Árbol de Clasificación / Generador de Símbolos

La función del bloque árbol de clasificación (o generador de símbolos), es determinar dentro de treinta y dos puntos ya definidos (determinados por el software de entrenamiento [4]), cuál es el que se encuentra más cerca al punto de entrada. Es decir, se calcula la distancia tipo L1 (ver sección 2.4) entre el punto de entrada y cada uno de los treinta y dos nodos, para poder definir cuál de los nodos se encuentra a menor distancia.

Coordenada "X"	Coordenada "Y"	Coordenada "Z"
47 ··· 32	31 ··· 16	15 ··· 0

Tabla 3.2: Formato del vector de datos del árbol de clasificación.

Para realizar la función antes mencionada, se diseñó el sistema que se muestra en la figura 3.4.

Al igual que el módulo reductor de dimensiones, al iniciar el funcionamiento, el sistema realiza una carga de coeficientes al banco de registros, por medio del módulo denominado ***Inicializacion_Nodo***, que se encarga de enviar los datos a los treinta y dos nodos (llamados ***Nodo***). Los datos se encuentran organizados en un arreglo de cuarenta y siete bits en formato punto fijo, en donde los dieciséis primeros son la coordenada "X", los siguientes dieciséis la coordenada "Y" y finalmente los dieciséis menos significativos son los correspondientes a la coordenada "Z", tal como se muestra en 3.2.

Una vez terminada la inicialización, el módulo indica que se encuentra listo para cargar coeficientes del exterior o ejecutar el proceso de cálculo del símbolo (que en realidad es la identificación del nodo que se encuentra más cercano al punto de entrada, o sea, el resultado del funcionamiento del módulo es un número del cero al treinta y uno). Este estado lo indica al colocar en alto la señal **ready**.

Para iniciar la carga de coeficientes, se coloca en estado alto la señal **load**, con lo que el sistema permite la escritura en uno de los treinta y dos nodos; los datos que se encuentran en **load_coef** se cargan en el nodo que se indica en la dirección de la señal **load_dir**. Además, se coloca en estado bajo la señal **ready**, con lo que se indica que se esta realizando la carga, una ve finalizada la operación, la señal vuelve al estado alto.

En el caso del proceso de cálculo del nodo más cercano, el sistema funciona de manera combinacional, es decir, no necesita de ninguna señal para el inicio del procedimiento, al colocar los datos válidos en las entradas, denominadas como **in_X**, **in_Y** y **in_Z**, y una vez que se hayan propagado las señales correspondientes, se encontrará una salida válida en el sistema, la que indica cuál nodo es el más cercano a la entrada.

Para realizar el cálculo del nodo mas cercano, primeramente se calcula la distancia Manhattan (o distancia L1) entre el punto de entrada y los nodos del sistema, a través de los módulos **Cálculo_L1**. Una vez que se tiene la distancia a cada nodo, se realizan comparación en parejas; por ejemplo, se comparan las distancias del **Nodo_0** y la del **Nodo_1**, las del **Nodo_2** y la del **Nodo_3** y así de manera sucesiva. Del primer nivel de comparación, resultan dieciséis distancias seleccionadas, las cuales pasan al siguiente nivel de comparación.

El segundo nivel de comparación funciona de igual manera, en este caso se operan comparaciones sobre dieciséis distancias, para dar como resultado ocho distancias. Posteriormente, se continúa con un tercer, cuarto y quinto nivel de comparación, lo que finalmente permite identificar cual es el nodo más cercano al punto de inicio.

Finalmente, con las comparaciones y un conjunto de módulos de multiplexación (denominados **Mux 2x1** en el diagrama de la figura 3.4) se produce a la salida el número que identifica el nodo más cercano a la entrada. Las señales de **Dir0** hasta **Dir4** son las salidas del sistema.

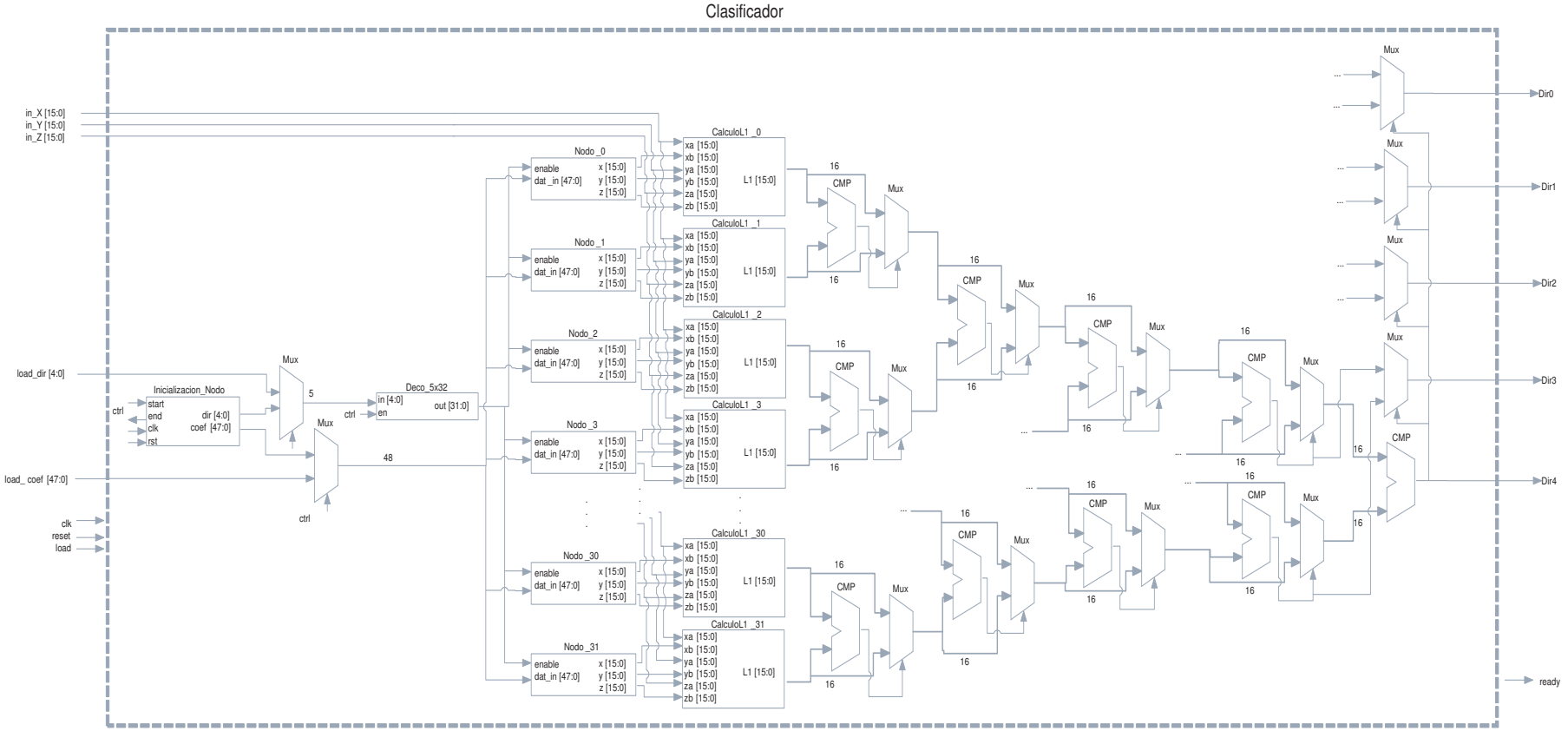

Figura 3.4: Diagrama del árbol de clasificación.

3.3 Unidad de Modelos Ocultos de Markov (HMMU)

El sistema de salida del SiRPA ejecuta el procedimiento descrito en la sección , es decir el algoritmo *Forward-Backward Procedure*, delimitado por los siguientes parámetros: veinte símbolos de largo de cadena, cinco estados para cada modelo oculto de Markov y un alfabeto de treinta y dos símbolos (ver sección 2.5). Para poder llevar a cabo el procedimiento se diseñó el módulo de la figura 3.6.

Para controlar la unidad, se implementó una máquina de estados con el diagrama de la figura 3.5.

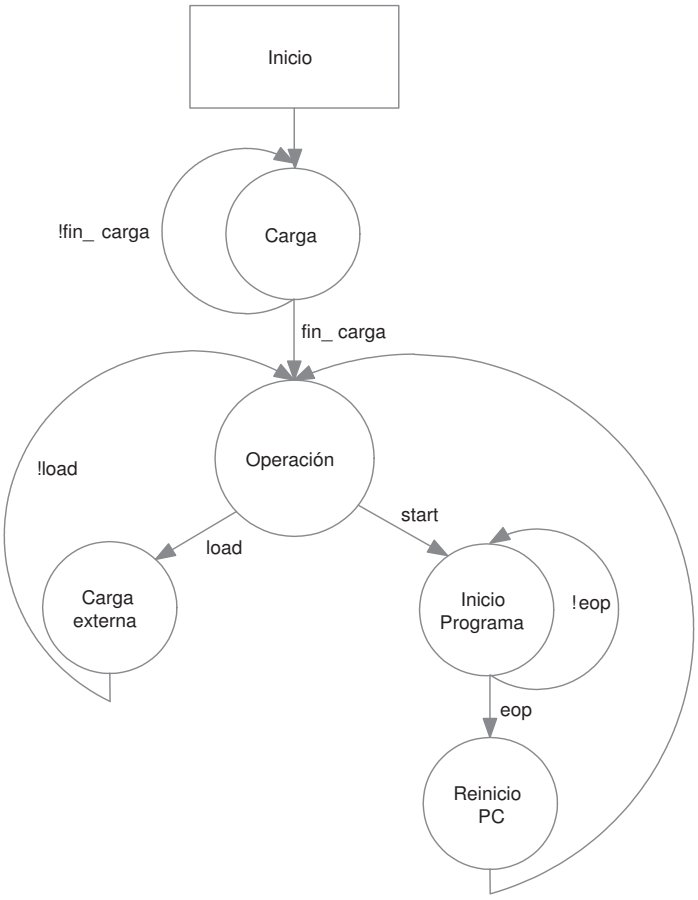

Figura 3.5: Máquina de estados del control del HMMU.

Después de un reinicio, el sistema procede a cargar los coeficientes por defecto en la memoria, a través del módulo **Inicia_coef**. Los datos se encuentran distribuidos según se indica en la tabla 3.3.

Antes de iniciar el procedimiento de cálculo, se deben cargar los veinte símbolos en el módulo denominado **Simbol_Reg**. El control general del SiRPA se encarga de cargar individualmente cada símbolo generado en el registro mediante la señal **ctrlSimb** y de dar la señal de inicio (*start* en la figura 3.5). Por otra parte, se puede proceder a realizar

Dirección de Memoria	Matriz	Modelo
0 ⋮ 4	Inicialización	Bosque
5 ⋮ 29	Transición	
30 ⋮ 189	Observación	
190 ⋮ 194	Inicialización	Motosierras
195 ⋮ 219	Transición	
220 ⋮ 379	Observación	
380 ⋮ 384	Inicialización	Disparos
385 ⋮ 409	Transición	
410 ⋮ 569	Observación	

Tabla 3.3: Direcciones de memoria de los coeficientes del HMMU.

Dirección de Memoria	Registros
570	temporal 0
⋮	⋮
585	temporal 15
586	acumulador 0
587	acumulador 1
588	acumulador 2
589	zero

Tabla 3.4: Registros de temporales del HMMU.

una carga externa de coeficientes al banco de memoria, mediante la activación de la señal **load**.

Si se inicia el proceso del algoritmo *Forward-Backward Procedure*, el sistema recorre la memoria de programa (**Rom_Mem**) hasta que se alcanza la instrucción "eop" (su estructura y funcionamiento se detallan más adelante) con lo que la máquina se detiene y regresa al estado de Operación.

Durante la operación de la unidad, además de los registros con las matrices de coeficientes de cada modelo oculto de Markov, se emplean registros para los cálculos intermedios del proceso. La dirección de memoria de los mismos, así como los nombres de los mismos se encuentran en la tabla 3.4.

El conjunto de instrucciones que ejecuta el módulo se encuentra descrito en la tabla 3.5. La función de cada una se explica a continuación:

- **nope:** es la instrucción que se emplea para que el sistema no realice ninguna acción.
- **add:** la instrucción realiza una suma del registro A con el registro B y lo almacena en el registro destino.
- **addi:** suma el contenido del registro A con el valor del inmediato, para almacenarlo en el registro destino.
- **sub:** al contenido del registro A le resta el registro B y lo almacena en el registro destino.
- **mult:** multiplica el contenido del registro A y el registro B y lo almacena en el registro destino.
- **div:** divide el registro A entre el registro B, el resultado lo almacena en el registro destino.
- **push:** esta instrucción toma los datos del registro indicado en A y los almacena en un registro para la comparación (módulo llamado **Comp_HMM**). Unicamente los registros acumulador 0, acumulador 1 o acumulador 2 son válidos para la comparación, en los mismos se almacenan las probabilidades de que la cadena de símbolos represente al modelo del Bosque, Motosierras o Disparos según corresponde, y por ende sólo éstos registros se debe usar como parámetros (el módulo de **Ctlr_HMM major** se encarga del control).

Ins	Composición						
nope	Datos sin efecto						
add	Reg A	Reg B	Reg Dest	Innecesario	Simbolo	Fila	Dirección
addi	Reg A	Innecesario	Reg Dest	Inmediato	Simbolo	Fila	Dirección
sub	Reg A	Reg B	Reg Dest	Innecesario	Simbolo	Fila	Dirección
mult	Reg A	Reg B	Reg Dest	Innecesario	Simbolo	Fila	Dirección
div	Reg A	Reg B	Reg Dest	Innecesario	Simbolo	Fila	Dirección
push	Reg A	Innecesario	Innecesario	Innecesario	Simbolo	Fila	Dirección
eop	Datos sin efecto						
Cantidad de bits de cada elemento de la instrucción							
3	10	10	10	32	5	3	2

Tabla 3.5: Conjunto de instrucciones del HMMU.

- **eop:** con esta instrucción se indica al sistema que se encuentra en el final del programa, y por ende se detiene el conteo del PC (*program counter*).

Cada instrucción tiene los parámetros: **Simbolo**, **Fila** y **Dirección**, los cuales determinan la dirección de memoria que se lee en el puerto cero del banco de memoria durante la ejecución de cada instrucción. La señal **Simbolo** se encarga de seleccionar cuál de los símbolos de entrada se pasa al módulo **Dir_Deco**; la combinación del símbolo con **Fila** produce las direcciones de memoria del coeficiente para los tres modelos ocultos. Finalmente, según se indique en el parámetro **Dirección** se pasa la dirección correspondiente al parámetro "registro A" de la instrucción, el modelo del Bosque, Motosierra o Disparo.

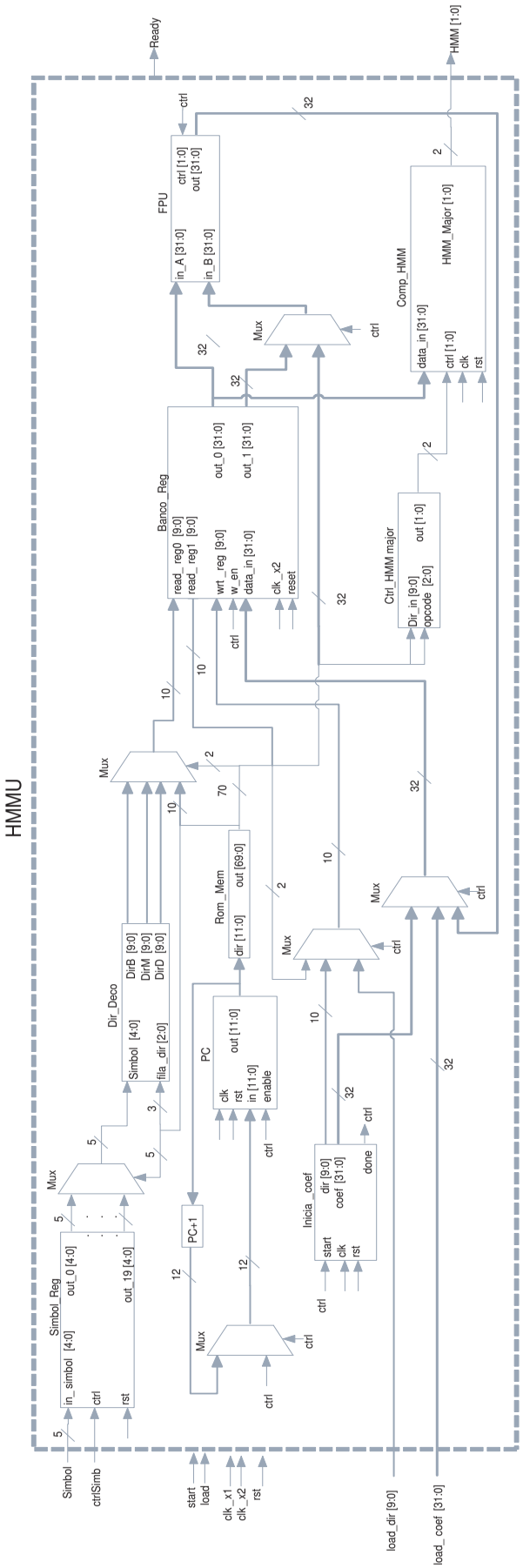

Figura 3.6: Diagrama del HMMU.

Capítulo 4

Resultados y análisis

4.1 Simulación del diseño de SiRPA

Para simular el funcionamiento del diseño del SiRPA, se empleó la herramienta *The Questa® Advanced Simulator* de la compañía *Mentor Graphics*[5]. Y como se desarrolló el código en verilog en la herramienta *ISE* de *Xilinx*, se emplearon las bibliotecas de compilación del ISE.

4.1.1 Simulación reductor de dimensiones

La síntesis del código del reductor de dimensiones da una idea de los recursos que consume la unidad, para éste módulo, los resultados indican que esta etapa emplea 519 "slices", 556 "flip flop" y 543 "LUT (Look up tables)".

Para corroborar el funcionamiento del bloque reductor de dimensiones, se diseñó un módulo de pruebas (llamado "Prueba_Reductor_PFijo"), el cuál se encarga de emular el funcionamiento normal del circuito que controla el reductor. Los datos que se emplean en esta simulación se encuentran en la sección A.1.

Para iniciar la simulación, primero se reinicia el sistema de reductor de dimensiones como lo muestra la figura 4.1. Se puede apreciar que después de un reinicio, las señales que son salidas de registros (**OUT0**, **OUT1** y **OUT2**) toman valores de defecto. En el estado cero del control (señal **Ctrl_State** de la figura 4.1) el reductor de dimensiones inicia la carga de coeficientes, mientras que en el estado uno espera que finalice esa carga, tal como lo muestra la figura 4.2.

Una vez que se inicia la carga de coeficientes, se recorren las treinta y dos posiciones de memoria y se van cargando los datos en el banco de registros. Al finalizar la carga, el módulo de inicialización indica el fin de la misma (señal **EndCoef**), con lo que el control pasa al estado dos. En el estado dos, el sistema se encuentra preparada para realizar una carga externa o una reducción de dimensiones, por lo que lo indica con la señal de

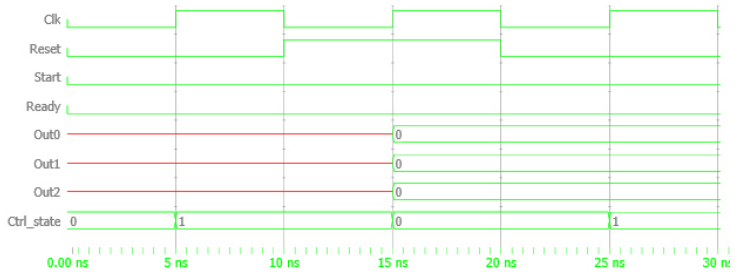

Figura 4.1: Simulación reductor de dimensiones: reinicio.

READY. La inicialización del sistema tarda un total de treinta y dos ciclos de reloj.

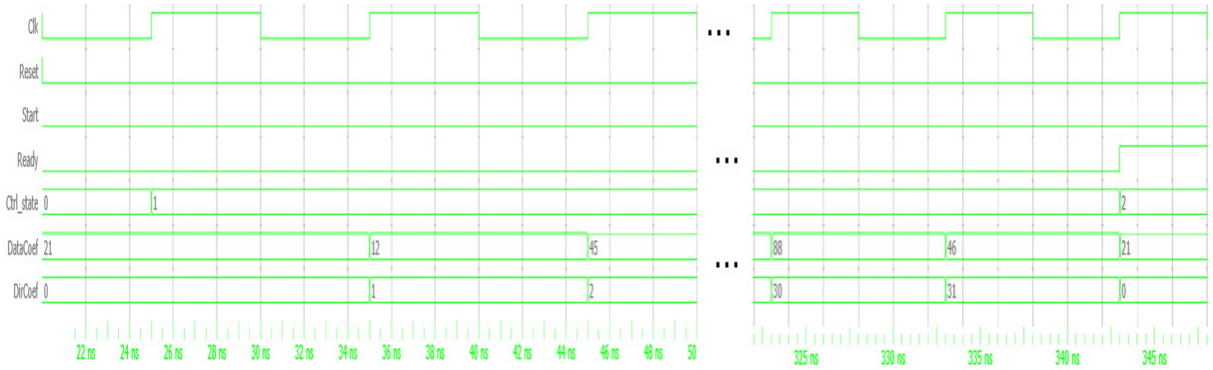

Figura 4.2: Simulación reductor de dimensiones: carga de coeficientes.

Si se inicia el proceso de reducción de dimensiones, el sistema se comporta como lo muestra la figura 4.3. En la misma se puede apreciar que el control ejecuta la transformación lineal de los parámetros de entrada en los estados del cuatro al diecinueve, y este proceso lo recorre tres veces (una para cada dimensión). En el estado diecinueve, el control escribe en el registro que corresponde a las salidas del módulo. En total, el proceso de reducción de dimensiones toma cuarenta y ocho ciclos de reloj.

Cuando el sistema se encuentra en el estado de espera de la señal de inicio o de carga externa de coeficientes (control en estado dos), si se realiza una carga externa, el control pasa al estado tres y el sistema realiza la operación, tal como se muestra en la figura 4.4. En la misma se puede apreciar que la señal **READY** pasa a un estado bajo mientras se cargan los datos al banco de registros.

Una vez simulado el funcionamiento del reductor de dimensiones, se procedió a realizar varias pruebas de reducción y comparar las mismas con los resultados teóricos, en la simulación se obtuvieron datos exactos, aún así se debe recordar que los coeficientes empleado son aleatorios y no los finales del SiRPA. Dichos datos se muestran en la tabla 4.1.

Prueba Uno	Entada 0: 17	Salida 0: 1737
	Entada 1: 37	
	Entada 2: 13	
	Entada 3: 41	Salida 1: 6268
	Entada 4: 43	
	Entada 5: 91	
	Entada 6: 34	Salida 2: 3728
	Entada 7: 55	
Prueba Dos	Entada 0: 57	Salida 0: 12537
	Entada 1: 87	
	Entada 2: 93	
	Entada 3: 61	Salida 1: 16818
	Entada 4: 35	
	Entada 5: 67	
	Entada 6: 70	Salida 2: 11676
	Entada 7: 99	
Prueba Tres	Entada 0: 78	Salida 0: 3933
	Entada 1: 95	
	Entada 2: 46	
	Entada 3: 35	Salida 1: 1583
	Entada 4: 17	
	Entada 5: 21	
	Entada 6: 30	Salida 2: -2756
	Entada 7: 5	
Prueba Cuatro	Entada 0: 11	Salida 0: 5317
	Entada 1: 32	
	Entada 2: 45	
	Entada 3: 68	Salida 1: 10985
	Entada 4: 79	
	Entada 5: 83	
	Entada 6: 75	Salida 2: 10465
	Entada 7: 69	

Tabla 4.1: Pruebas de la reducción de dimensiones.

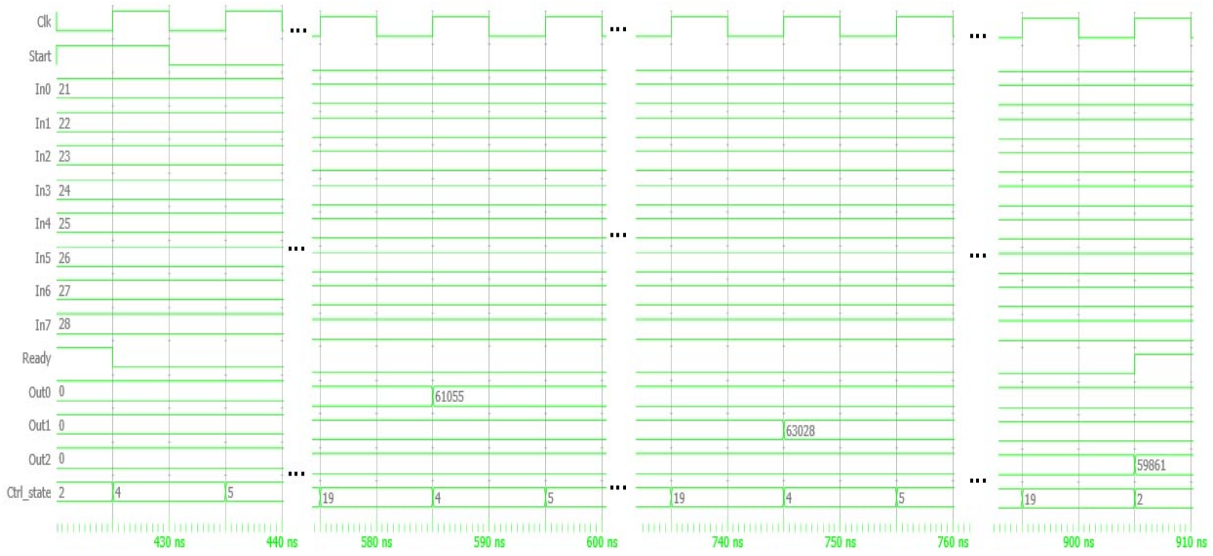

Figura 4.3: Simulación reductor de dimensiones: reducción de dimensiones.

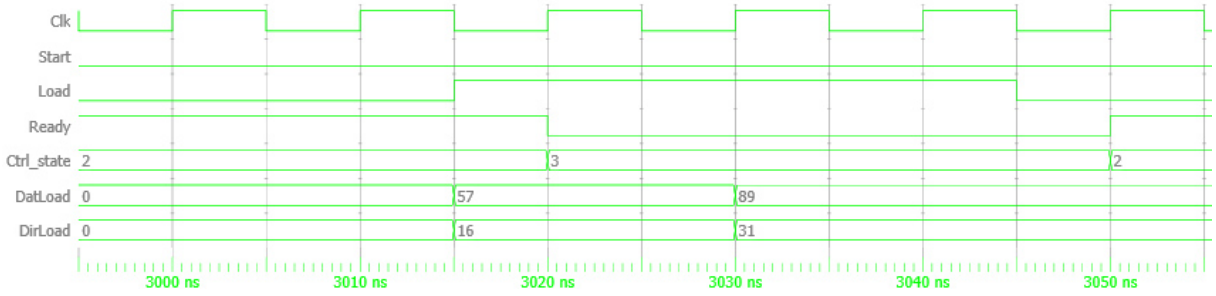

Figura 4.4: Simulación reductor de dimensiones: carga externa de coeficientes.

4.1.2 Simulación árbol de clasificación / generador de símbolos

Después de la síntesis de este módulo, el programa indica que el consumo de recursos de esta unidad es de 3491 "slices", 1544 "flip flops" y 6601 "LUT".

Al igual que en la simulación anterior, se diseñó el módulo "Prueba_Clasificador", con el cual se emula el funcionamiento del circuito del control general del sistema. Para comprobar el funcionamiento se cargó la memoria del árbol de clasificación con coeficientes aleatorios. Éstos se encuentran en la sección A.2.

Para iniciar el funcionamiento del módulo, primero se debe hacer un reinicio del sistema, con lo que se borran los registros y se puede iniciar el proceso de carga. Este proceso consiste en que el control del generador de símbolos (señal **Ctrl_state** en la figura) se coloca en el estado cero despúes del reinicio, e inicia el proceso de carga de coeficientes según se detalla en la figura 4.5.

Una vez que se da inicio a la carga de coeficientes al banco de memoria, el módulo **"InicializacionArbol"** se encarga de recorrer todas las treinta y dos posiciones de memoria y enviar los datos que corresponden a cada posición, mientras esto ocurre, el control del sistema espera en el estado uno por el final de la carga. La simulación de

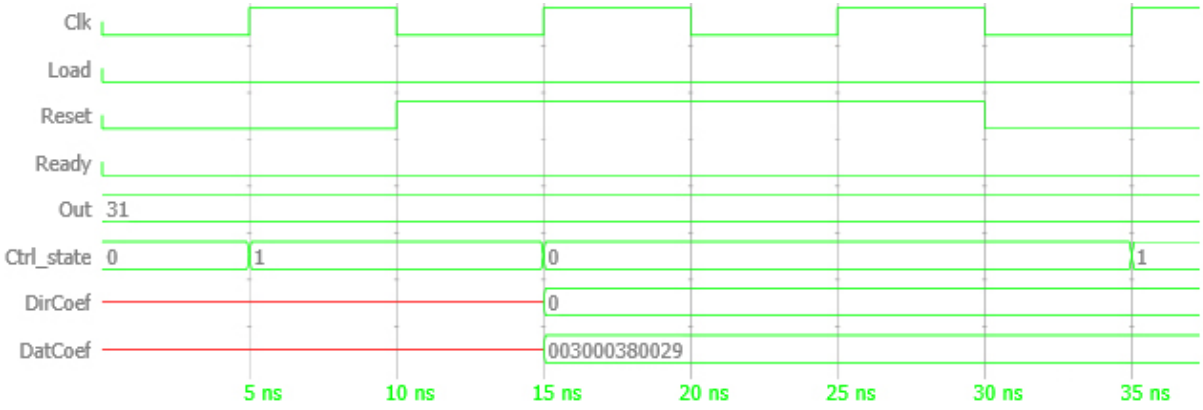

Figura 4.5: Simulación árbol de clasificación: reinicio.

este proceso se encuentra en la figura 4.6.

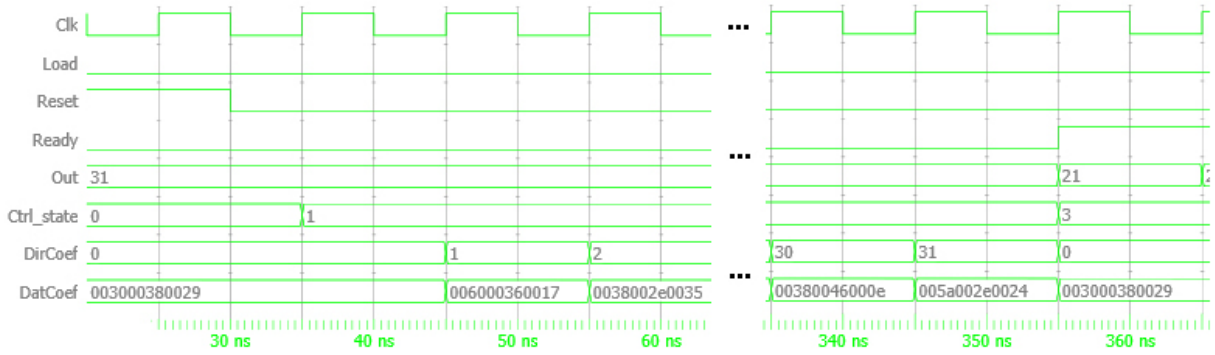

Figura 4.6: Simulación árbol de clasificación: inicialización.

Una vez que se termina la carga de los coeficientes, el sistema pasa a un estado en el que se encuentra listo para operar o realizar una carga de datos desde el exterior del sistema. Cuando esto ocurre, el control pasa al estado tres y la señal **Ready** se coloca en estado alto (ver figura 4.6). Como la operación del sistema es combinacional, basta con variar los datos de entrada para que a la salida, después de transcurrido el tiempo de propagación de las señales, se tenga el resultado.

En la figura 4.7 se muestra el proceso de cálculo del símbolo, que en realidad es identificar el nodo más cercano a los puntos de entrada. Para comprobar el funcionamiento se realizaron siete cálculos, en los que se variaron las señales de entrada (nombradas **InX**, **InY** e **InZ** en la simulación) y se corroboró el resultado (señal **Out**).

Para realizar una carga de externa de coeficientes, se debe comprobar que la señal **Ready** se encuentra en estado alto, si es así, se puede continuar con el proceso. Al colocar la señal **load** en alto, el control habilita la escritura en el banco de registros y los datos que se encuentran en la señal **LoadData** se graban en la dirección indicada por **LoadDir**. Una vez concluida la carga, se regresa al estado de operación. Todo esto se muestra en la figura 4.7.

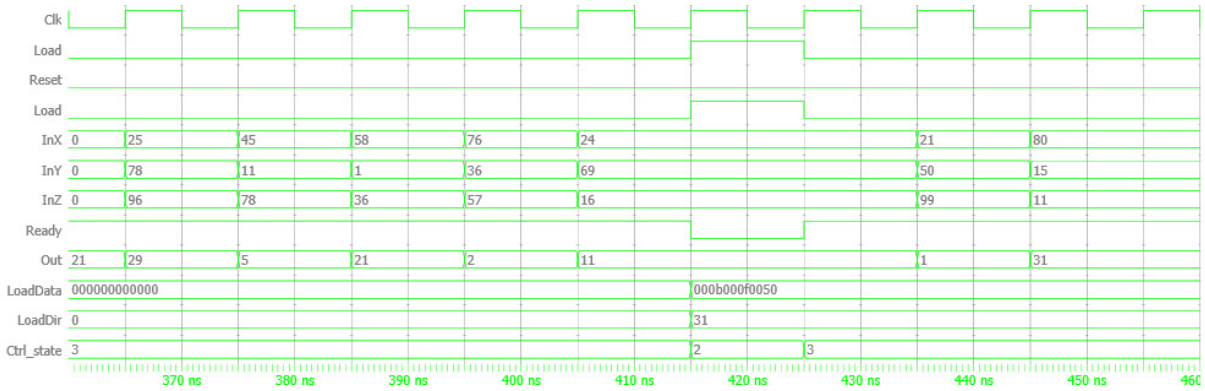

Figura 4.7: Simulación árbol de clasificación: funcionamiento y carga.

4.1.3 Simulación del HMMU

Esta etapa del SiRPA, tras la síntesis, consume recursos equivalentes a: 4700 "slices", 253 "flip flops" y 8616 "LUT".

Para comprobar el funcionamiento del HMMU, se diseñó el modulo de simulación denominado **"Prueba_HMMU"**. Los coeficientes empleados para la simulación son números aleatorios y se encuentran documentados en la sección A.2.

Al iniciar el funcionamiento del HMMU, se debe reiniciar el sistema. En el reinicio, el control (señal **Ctrl_state**) se coloca en el estado cero, con lo que se inicia la carga de coeficientes, este comportamiento está plasmado en la figura 4.8.

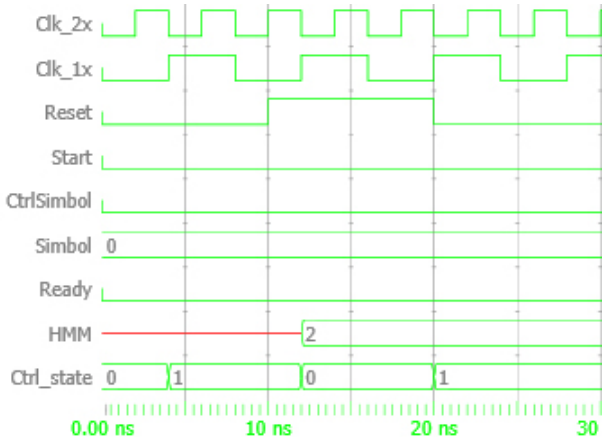

Figura 4.8: Simulación hmmu: reinicio.

Cuando se inicia la carga de coeficientes, el módulo llamado **Inicializacion_HMMU** recorre las quinientas setenta posiciones de memoria, cargando el banco de registros con los datos por defecto. Este funcionamiento se comprobó durante la simulación y se muestra en la figura 4.9. Al terminar la carga de coeficientes, el sistema está preparado para operar o realizar una carga externa y lo indica colocando la señal **Ready** en alto.

Antes de empezar el proceso de cálculo de la probabilidad de la cadena de símbolos, se deben cargar los símbolos generados en el registro de corrimiento en la entrada del

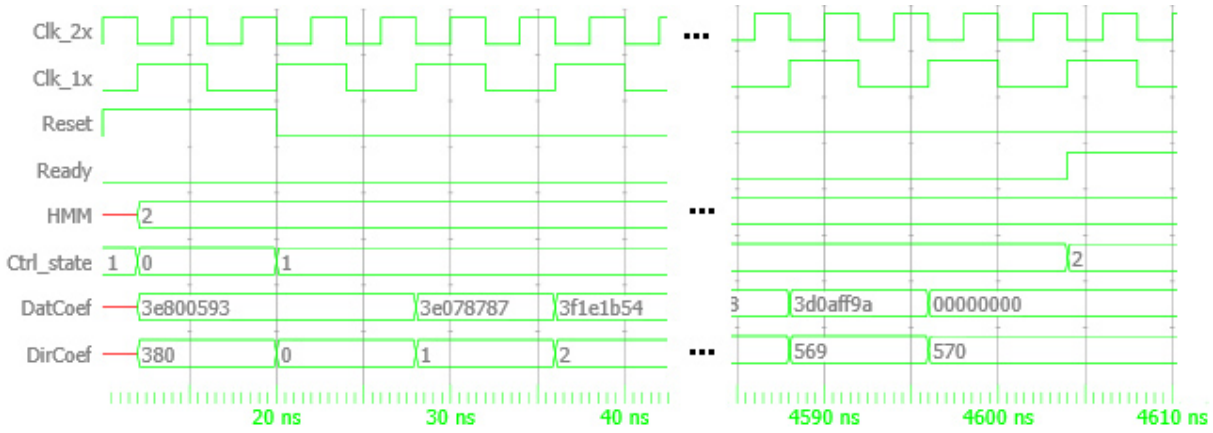

Figura 4.9: Simulación HMMU: inicialización.

hmm. Para corroborar este funcionamiento, se simuló la carga de símbolos tal como se muestra en la figura 4.10. Durante esta carga, se escogieron los símbolos al azar: "6, 9, 3, 10, 26, 31, 16, 22, 25, 1, 29, 28, 14, 15, 18, 20, 23, 29, 30, 1".

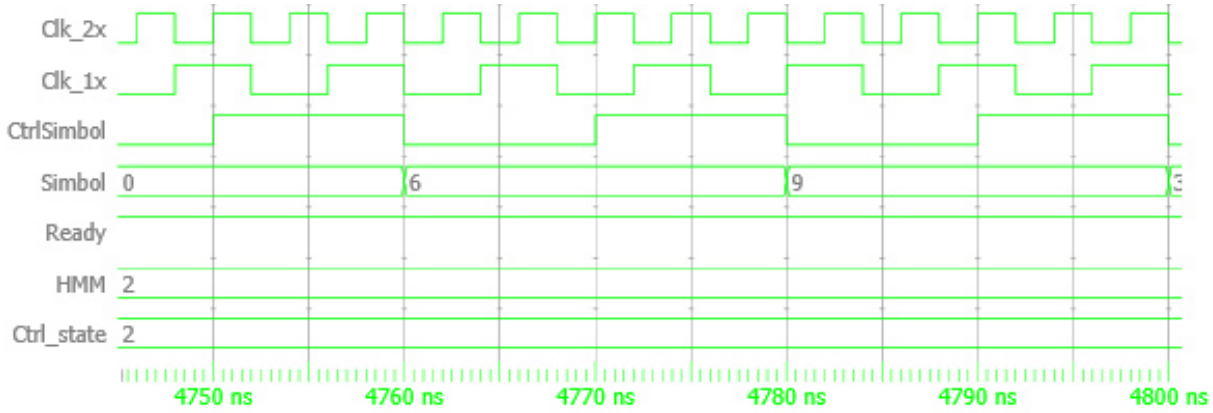

Figura 4.10: Simulación HMMU: carga de símbolos.

Una vez que se concluyó con la carga de los veinte coeficientes, el sistema está preparado para realizar el cálculo de probabilidades. Al recibir la señal **Start**, el sistema inicia el cálculo de las probabilidades de la cadena, para los tres modelos con los que cuenta, es decir el bosque, las motosierras y los disparos. El programa completo cuenta con tres mil trescientas noventa y cuatro instrucciones y al finalizar, la señal **Ready** lo indica colocándose en alto.

En el estado cuatro del control, se recorre el programa, sin embargo cuando se alcanza la instrucción *eop*, el control pasa al estado cinco, con lo que se reinicia el PC y se detiene el programa del hmmu.

En caso de que no se inicie el cálculo de probabilidades, sino que se procede a realizar una carga externa de coeficientes (colocar la señal **Load** en alto), el control pasa al estado tres y habilita la escritura en el banco de memoria. Este funcionamiento se muestra en la figura 4.12.

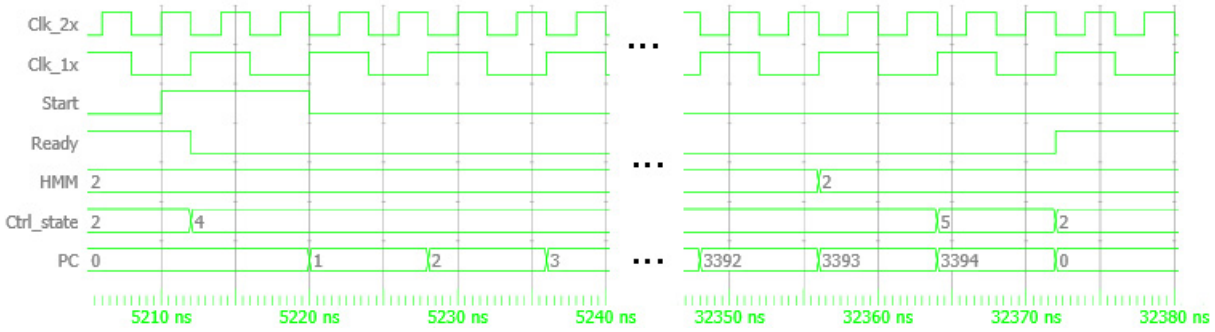

Figura 4.11: Simulación HMMU: funcionamiento.

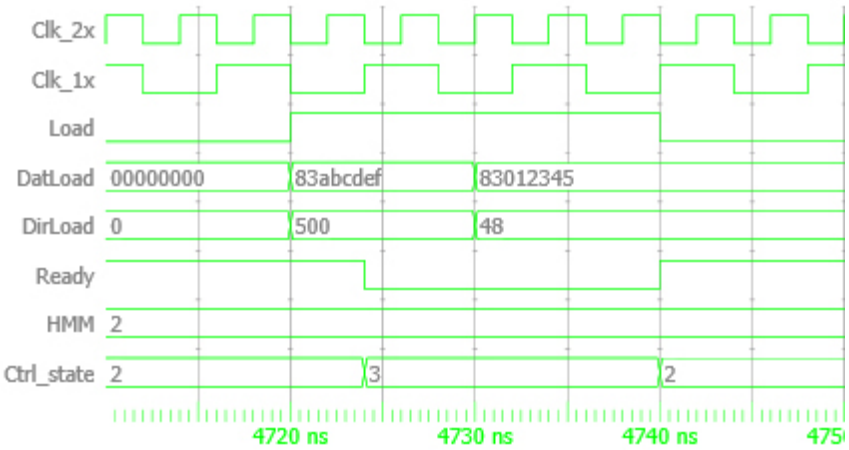

Figura 4.12: Simulación HMMU: carga externa de coeficientes.

Capítulo 5

Conclusiones y Recomendaciones

5.1 Conclusiones

Al desarrollar una implementación digital de un sistema que se encuentra diseñado en software, se deben realizar modificaciones a algunos algoritmos y fases del proceso. En el caso del reductor de dimensiones, por tratarse de operaciones aritméticas simples, no fue necesario modificar el procedimiento que se realiza y se implementó tal como se ejecuta en el soffware. En el árbol de clasificación, también conocido como generador de símbolos, fue necesario pasar de un árbol k-d, a una estructura más simple de árbol de comparadores, ya que el algoritmo en software realizaba cálculos frecuentes de hiperrectángulos, lo cual lo hacía poco práctico y complicado para transformarlo en un diseño de hardware. Finalmente, la unidad que se encarga de evaluar lo modelos ocultos de Markov, al ser implementada por un microprocesador, ejecuta el algoritmo de manera similar a la implementada en el computador.

Con una implementación del reductor de dimensiones y el árbol de clasificación, desarrollada con datos en formatos punto fijo, se logró implementar la extracción de características del audio de entrada de manera satisfactoria. Junto con el banco de filtros y éstos módulos, se genera la cadena de símbolos que se emplea para el cálculo de las probabilidad en el HMMU; por ende, la presición y características que proporciona este formato de datos digitales es suficiente en esta aplicación.

Dado que en la etapa de toma de decisiones se comparan probabilidades, se diseñó el módulo HMMU con datos en formato punto flotante. Este tipo de datos brinda la posibilidad de representar datos con diferencias de $1.4 \cdot 10^{-45}$, mientras que el formato punto fijo $1.22 \cdot 10^{-4}$, por lo que el HMMU posee una mejor presición, y se alcanza el objetivo de determinar cual modelo tiene mayores probabilidades de haber producido la cadena de símbolos.

El diseño digital del SiRPA permite cumplir con la misma funcionalidad que el sistema en software, además que permite reconfigurar los coeficientes que se encuentran en la memoria, así, se tiene una implementación funcional del sistema, que con el debido

proceso puede ser implementada en chip.

5.2 Recomendaciones

Se deben realizar pruebas de funcionamiento una vez que se cuente con los coeficientes definitivos para el sistema. De esta manera, se puede comprobar los cálculos y el manejo de los datos en circunstancias más cercanas al funcionamiento real del sistema.

Realizar pruebas que permitan determinar cuál es la frecuencia óptima de operación para cada módulo del SiRPA sería recomendable. Ya que en el presente proyecto, únicamente se determinó cual es la frecuencia mínima de operación; y las pruebas pueden ayudar a mejorar la función del SiRPA.

Bibliografía

[1] Luis Abrahams. Implementación de una metodología para lograr la integración física "correcta por construcción (cbc)" del sistema de reconocimiento de patrones acústicos (sirpa). Tésis de licenciatura, Escuela de Ingeniería Electrónica, ITCR, 2012.

[2] Hugo Barrantes. *Elementos de algebra lineal*. Editorial UNED, 1998.

[3] Roberto Huerta C. Representación de números en punto fijo y flotante, 2003. URL http://www2.elo.utfsm.cl/elo385/docs/Biblio/Lab4.

[4] Jorge Cárdenas. Estrategias de entrenamiento de modelos ocultos de markov para reconocimiento de patrones acústicos. Tésis de maestría, Escuela de Ingeniería en Computación, ITCR, 2012.

[5] Mentor Graphics. Questa sim user's manual [online]. 2011 [visitado el 26 de junio de 2013]. URL http://rise.cse.iitm.ac.in/people/faculty/kama/prof/questa_sim_user_manual.pdf.

[6] B. H. Juang L. R. Rabiner. An introduction to hidden markov models, 1986. URL http://luthuli.cs.uiuc.edu/daf/courses/Signals.

[7] Andrés Mayo. Punto fijo versus punto flotante: Principios básicos de funcionamiento, ventajas y desventajas [online]. 2007 [visitado el 26 de junio de 2013]. URL http://www.andresmayo.com/data.

[8] Isabel Molina. Capítulo 4: Medidas de proximidad, 2009. URL http://halweb.uc3m.es/esp/Personal/personas/imolina/MiDocencia/TecnicasInvestigacion0910/SlidesTema4MedidasProximidad.pdf.

[9] T. Rydén O. Cappé, E Moulines. *Inference in Hidden Markov Moddels*. Pringer, 2007.

[10] Erick Salas. Reconocimiento en timepo real de patrones acústicos de motosierras y disparos por medio de una implementación en fpga de modelos ocultos de markov. Tésis de licenciatura, Escuela de Ingeniería Electrónica, ITCR, 2010.

[11] Mario Sequeira. Módulo de reducción de dimensiones espectrales en un sistema de reconocimiento de patrones acústicos de motosierras y disparos por medio de una

implementación en fpga. Tésis de licenciatura, Escuela de Ingeniería Electrónica, ITCR, 2011.

[12] SINAC. Políticas para las Áreas silvestres protegidas [online]. 2011 [visitado el 26 de junio de 2013]. URL http://www.sinac.go.cr/documentacion.

[13] Paula Tarrío. Redes de sensores fundamentos y aplicaciones, 2008. URL http://www.bduimp.es/archivo/conferencias/pdf/08_10038_10_PaulaTarrio_idc50141.pdf.

[14] Robert Usselmann. Open floating point unit, 2000. URL http://opencores.org/project,fpuvhdl.

Apéndice A

Coeficientes para las simulaciones

A.1 Datos para el reductor de dimensiones

Para realizar la prueba, se cargó el reductor de dimensiones con los coeficientes aleatorios:

$$W_{8,3} = \begin{pmatrix} 11 & 27 & 43 \\ 84 & 55 & 16 \\ 87 & 48 & 29 \\ 11 & 10 & 28 \\ 16 & 44 & 50 \\ 76 & 77 & 98 \\ 29 & 28 & 91 \\ 2 & 88 & 46 \end{pmatrix} \tag{A.1}$$

$$Offset_{1,8} = \begin{pmatrix} 21 & 12 & 45 & 98 & 38 & 56 & 32 & 8 \end{pmatrix} \tag{A.2}$$

A.2 Datos el árbol de clasificación

Para realizar las pruebas del árbol de clasificación, se colocaron coeficientes aleatorios en el módulo, tal como se muestran en la tabla A.1.

A.3 Datos para el HMMU

Para realizar las pruebas del HMMU, se cargaron matrices aleatorias con los coeficientes que se muestran en las matrices π, A y las figuras A.1, A.2, A.3:

Número Nodo	Coordenada "X"	Coordenada "Y"	Coordenada "Z"
0	41	56	48
1	23	54	96
2	53	46	56
3	84	80	82
4	56	76	13
5	67	18	83
6	71	65	10
7	8	16	73
8	49	49	6
9	16	1	76
10	81	53	14
11	20	76	9
12	34	47	35
13	44	62	73
14	65	87	38
15	36	46	46
16	76	80	50
17	87	28	26
18	95	67	10
19	62	63	28
20	42	37	16
21	28	2	26
22	39	78	16
23	99	96	73
24	50	37	25
25	64	80	69
26	77	46	9
27	80	73	99
28	98	18	63
29	30	76	78
30	14	70	59
31	36	46	90

Tabla A.1: Coeficientes del clasificador de símbolos.

$$\pi_{Bosque} = \begin{pmatrix} 0.2500 \\ 0.1324 \\ 0.6179 \\ 0.1324 \\ 0.6179 \end{pmatrix} \tag{A.3}$$

$$A_{Bosque} = \begin{pmatrix} 0.9994 & 0.0003 & 0.0003 & 0.0003 & 0.0003 \\ 0.0003 & 0.9996 & 0.0001 & 0.9996 & 0.0001 \\ 0.0003 & 0.0001 & 0.9996 & 0.0001 & 0.9996 \\ 0.0003 & 0.9996 & 0.0001 & 0.9996 & 0.0001 \\ 0.0003 & 0.0001 & 0.9996 & 0.0001 & 0.9996 \end{pmatrix} \tag{A.4}$$

```
Columns 1 through 8

  0.0000    0.2950    0.0208    0.0000    0.0000    0.0000    0.0000    0.0004
  0.1268    0.0000    0.0000    0.0095    0.0000    0.0000    0.0000    0.0000
  0.0000    0.0000    0.0000    0.0000    0.2197    0.1769    0.0524    0.0000
  0.1268    0.0000    0.0000    0.0095    0.0000    0.0000    0.0000    0.0000
  0.0000    0.0000    0.0000    0.0000    0.2197    0.1769    0.0524    0.0000

Columns 9 through 16

  0.0930    0.0304    0.0317    0.0000    0.0894    0.0000    0.0000    0.0000
  0.0000    0.0000    0.0000    0.0000    0.0000    0.0375    0.0388    0.0691
  0.0000    0.0000    0.0000    0.0980    0.0000    0.0000    0.0000    0.0000
  0.0000    0.0000    0.0000    0.0000    0.0000    0.0375    0.0388    0.0691
  0.0000    0.0000    0.0000    0.0980    0.0000    0.0000    0.0000    0.0000

Columns 17 through 24

  0.1382    0.0000    0.0000    0.0000    0.0782    0.0000    0.0000    0.0000
  0.0000    0.0234    0.0000    0.0000    0.0000    0.0451    0.2585    0.1886
  0.0000    0.0000    0.0149    0.0938    0.0000    0.0000    0.0000    0.0000
  0.0000    0.0234    0.0000    0.0000    0.0000    0.0451    0.2585    0.1886
  0.0000    0.0000    0.0149    0.0938    0.0000    0.0000    0.0000    0.0000

Columns 25 through 32

  0.0256    0.0000    0.0000    0.0000    0.1860    0.0114    0.0000    0.0000
  0.0000    0.0000    0.0538    0.1489    0.0000    0.0000    0.0000    0.0000
  0.0000    0.1211    0.0000    0.0000    0.0000    0.0000    0.0782    0.1450
  0.0000    0.0000    0.0538    0.1489    0.0000    0.0000    0.0000    0.0000
  0.0000    0.1211    0.0000    0.0000    0.0000    0.0000    0.0782    0.1450
```

Figura A.1: Matrices de observación del bosque para pruebas del HMMU. Generado con Matlab.

$$\pi_{Motosierras} = \begin{pmatrix} 0.3913 \\ 0.5652 \\ 0.0435 \\ 0.5652 \\ 0.0435 \end{pmatrix} \tag{A.5}$$

$$A_{Motosierras} = \begin{pmatrix} 0.9996 & 0.0002 & 0.0001 & 0.0002 & 0.0001 \\ 0.0001 & 0.9998 & 0.0001 & 0.9998 & 0.0001 \\ 0.0002 & 0.0003 & 0.9995 & 0.0003 & 0.9995 \\ 0.9996 & 0.0002 & 0.0001 & 0.0002 & 0.0001 \\ 0.0001 & 0.9998 & 0.0001 & 0.9998 & 0.0001 \end{pmatrix} \tag{A.6}$$

```
Columns 1 through 8

       0    0.0452    0.0007    0.0000    0.0000    0.0000    0.0000    0.0000
       0    0.0000    0.0000    0.0521    0.0085    0.0000    0.0000    0.0000
       0    0.0000    0.0000    0.0000    0.0000    0.0021    0.3564    0.0000
       0    0.0000    0.0000    0.0521    0.0085    0.0000    0.0000    0.0000
       0    0.0000    0.0000    0.0000    0.0000    0.0021    0.3564    0.0000

Columns 9 through 16

  0.0000    0.1073         0    0.0000    0.0009         0    0.0000    0.0079
  0.4320    0.0000         0    0.0000    0.0000         0    0.0689    0.0000
  0.0000    0.0000         0    0.0000    0.0000         0    0.0000    0.0000
  0.4320    0.0000         0    0.0000    0.0000         0    0.0689    0.0000
  0.0000    0.0000         0    0.0000    0.0000         0    0.0000    0.0000

Columns 17 through 24

  0.0000    0.0197    0.4811    0.0053         0    0.0000    0.0000    0.0000
  0.0005    0.0000    0.0000    0.0000         0    0.0014    0.0704    0.0000
  0.0000    0.0000    0.0000    0.0000         0    0.0000    0.0000    0.2646
  0.0005    0.0000    0.0000    0.0000         0    0.0014    0.0704    0.0000
  0.0000    0.0000    0.0000    0.0000         0    0.0000    0.0000    0.2646

Columns 25 through 32

  0.0000    0.0000    0.0001    0.0000    0.0000    0.3317    0.0000    0.0000
  0.0000    0.0000    0.0000    0.1131    0.2454    0.0000    0.0000    0.0077
  0.3768    0.0002    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
  0.0000    0.0000    0.0000    0.1131    0.2454    0.0000    0.0000    0.0077
  0.3768    0.0002    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
```

Figura A.2: Matriz de observación de las motosierras para pruebas del HMMU. Generado con Matlab.

$$\pi_{Disparos} = \begin{pmatrix} 0.0000 \\ 0.0000 \\ 1.0000 \\ 0.0000 \\ 0.0000 \end{pmatrix} \tag{A.7}$$

$$A_{Disparos} = \begin{pmatrix} 0.9996 & 0.0002 & 0.0001 & 0.0002 & 0.0001 \\ 0.0001 & 0.9998 & 0.0001 & 0.9998 & 0.0001 \\ 0.0002 & 0.0003 & 0.9995 & 0.0003 & 0.9995 \\ 0.0001 & 0.9998 & 0.0001 & 0.9998 & 0.0001 \\ 0.0002 & 0.0003 & 0.9995 & 0.0003 & 0.9995 \end{pmatrix} \tag{A.8}$$

```
Columns 1 through 8

  0.0000    0.1632    0.0026    0.0000    0.0000    0.0000    0.0000    0.0814
  0.0000    0.0000    0.0000    0.2019    0.0000    0.0000    0.0000    0.0000
  0.0073    0.0000    0.0000    0.0000    0.1964    0.0153    0.7060    0.0000
  0.0000    0.0000    0.0000    0.2019    0.0000    0.0000    0.0000    0.0000
  0.0073    0.0000    0.0000    0.0000    0.1964    0.0153    0.7060    0.0000

Columns 9 through 16

  0.0423    0.0805         0    0.0000    0.0011         0    0.0000    0.0000
  0.0000    0.0000         0    0.0000    0.0000         0    0.0454    0.0396
  0.0000    0.0000         0    0.0034    0.0000         0    0.0000    0.0000
  0.0000    0.0000         0    0.0000    0.0000         0    0.0454    0.0396
  0.0000    0.0000         0    0.0034    0.0000         0    0.0000    0.0000

Columns 17 through 24

  0.0027    0.0000    0.0000    0.0000    0.0018    0.0000    0.0000    0.4112
  0.0000    0.3900    0.0000    0.0000    0.0000    0.0136    0.2208    0.0000
  0.0000    0.0000    0.0049    0.0145    0.0000    0.0000    0.0000    0.0000
  0.0000    0.3900    0.0000    0.0000    0.0000    0.0136    0.2208    0.0000
  0.0000    0.0000    0.0049    0.0145    0.0000    0.0000    0.0000    0.0000

Columns 25 through 32

  0.0511    0.0000         0    0.0000    0.1370    0.0252    0.0000    0.0000
  0.0000    0.0000         0    0.0887    0.0000    0.0000    0.0000    0.0000
  0.0000    0.0184         0    0.0000    0.0000    0.0000    0.0000    0.0339
  0.0000    0.0000         0    0.0887    0.0000    0.0000    0.0000    0.0000
  0.0000    0.0184         0    0.0000    0.0000    0.0000    0.0000    0.0339
```

Figura A.3: Matriz de observación de los disparos para pruebas del HMMU. Generado con Matlab.