UNIVERSIDAD AUTÓNOMA DE QUERÉTARO

FACULTAD DE INGENIERÍA

IPCORE PARA LA GENERACIÓN DE PERFILES DE MOVIMIENTO BASADO EN TECNOLOGÍA FPGA.

TESIS

Que como parte de los requisitos para obtener el título de

INGENIERO EN ELECTROMECÁNICA.

PRESENTA:

CÁRDENAS RUIZ MARCO ALONSO.

DIRECTOR DE TESIS
DR. ROQUE A. OSORNIO RÍOS.

CO-DIRECTOR DE TESIS
DR. LUIS MORALES VELÁZQUEZ

SAN JUAN DEL RÍO, QRO. SEPTIEMBRE 2010

Universidad Autónoma de Querétaro
Facultad de Ingeniería
Dirección

ACUERDO 086/10

C. U. 15 de febrero de 2010

C. MARCO ALONSO CÁRDENAS RUIZ
Pasante de Ingeniería en Electromecánica
Presente.

Con relación a su oficio enviado al H. Consejo Académico de la Facultad en el que solicita titularse bajo la opción de tesis individual, me permito informarle que en la sesión ordinaria del 15 de febrero del año en curso, este cuerpo colegiado acordó aceptar la opción de titulación por lo que deberá trabajar en el tema **"IPCORE PARA LA GENERACIÓN DE PERFILES DE MOVIMIENTO BASADO EN TECNOLOGÍA FPGA"**, bajo la dirección del DR. ROQUE ALFREDO OSORNIO RÍOS.

El Contenido aceptado por el H. Consejo Académico es el siguiente:

Capítulo 1. Introducción.
- 1.1 Antecedentes.
- 1.2 Objetivos.
- 1.3 Justificación.
- 1.4 Planteamiento General.

Capítulo 2. Revisión de Literatura.
- 2.1 Estado del Arte.
  - 2.1.1 Sistemas de arquitectura abierta.
  - 2.1.2 Dinámica del movimiento en maquinaria.
  - 2.1.3 El FPGA como opción para aplicaciones mecatrónicas.
  - 2.1.4 Trabajos reportados en el ámbito de la generación de trayectorias y perfiles de movimiento.
- 2.2 Tipos de Máquinas-Herramientas.
- 2.3 Dinámica de Movimiento.
- 2.4 Diseño Polinomial para la Optimización de la Dinámica del Movimiento.
  - 2.4.1 Propiedades básicas de los polinomios.
  - 2.4.1.1 Definición de un polinomio.
  - 2.4.1.2 Teorema de complejidad.
  - 2.4.1.3 Polinomios unitarios.
  - 2.4.1.4 Valor inicial y valor final.
  - 2.4.1.5 Máximos y mínimos locales.
  - 2.4.1.6 Invarianza morfológica.
  - 2.4.1.7 Derivadas.
  - 2.4.2 Polinomios en tiempo discreto.
  - 2.4.2.1 Diferencia discreta.
  - 2.4.2.2 Integración discreta.
  - 2.4.2.3 Reconstrucción por diferencias.

Centro Universitario
Cerro de las Campanas
Santiago de Querétaro
Qro., México C.P. 76010
Tel. 01 (442) 192 12 00
Ext. Fax. 6005

También hago de su conocimiento las disposiciones de nuestra Facultad, en el sentido que antes del Examen profesional deberá cumplir con los requisitos de nuestra legislación y deberá imprimir el presente oficio en todos los ejemplares de su tesis.

Atentamente
"EL INGENIO PARA CREAR NO PARA DESTRUIR"

DR. GILBERTO HERRERA RUIZ
Director
c.c.p. Archivo

*GHR/DHM.

Centro Universitario
Cerro de las Campanas
Santiago de Querétaro
Qro., México C.P. 76010
Tel. 01 (442) 192 12 00
Ext. Fax. 6005

DEDICATORIA.

El presente trabajo de tesis está dedicado principalmente a mis padres, que a pesar de la lejanía han estado conmigo y con mis hermanas siempre y en todo momento, brindándome su amor y apoyo incondicional. Gracias por sus consejos y palabras de aliento, que han hecho de mí una mejor persona y por darme el mejor de los ejemplos, demostrándome que mediante trabajo duro, una actitud positiva y perseverante y muchos sacrificios se pueden lograr grandes cosas en la vida. Gracias por darme una educación de calidad de la que no todos podemos gozar, por esto y miles de cosas más, *¡Gracias!*

A mis hermanas que siempre me han demostrado su cariño y amor incondicional.

AGRADECIMIENTOS.

La culminación de esta tesis representa el término de una etapa como estudiante de tiempo completo y el inicio de una vida laboral. En toda la experiencia universitaria y la conclusión del trabajo de tesis ha habido personas que merecen mi agradecimiento porque gracias a su valiosa aportación ha sido posible avanzar en mi educación superior.

A mi director de tesis Dr.Roque A. Osornio, Dr. Luis Morales co-director de tesis, M. I. José de Jesús Rangel, M. I. Jesús Rooney Rivera quienes me apoyaron a lo largo de la realización de esta tesis.

A mis más cercanos amigos y compañeros que han estado conmigo en muchos momentos a lo largo de la licenciatura: Mora, Alex, Piña, Ángel, Julián y demás amigos que siempre demostraron una gran disposición.

A todos los profesores de la facultad de ingeniería que siempre mostraron gran disponibilidad para despejar las dudas y ayudarme a crecer intelectualmente y de manera personal.

A la facultad de ingeniería que es la responsable de brindarme una educación superior de calidad, gracias.

ÍNDICE.

ÍNDICE DE FIGURAS.

CAPÍTULO 1. INTRODUCCIÓN.

Uno de los puntos más importantes a considerar en el proceso de maquinado en máquinas-herramientas CNC es el estudio de la dinámica del movimiento, teniendo en cuenta que si ésta dinámica no es óptima puede llegar a causar serios problemas, viéndose comprometida la calidad del producto y desde luego la integridad de la máquina, pues el desgaste y las vibraciones en el sistema mecánico de la máquina son inminentes, aumentando la frecuencia de mantenimiento para ésta, agravándose cuando se trata de Maquinados de Alta Velocidad (MAV) (Osornio, 2007).

En ésta tesis, el objeto principal de estudio es el desarrollo de un bloque lógico para la generación de perfiles y monitoreo en maquinaria CNC, teniendo en cuenta que para el desarrollo de éste proyecto de tesis, se requiere reunir información suficiente de tal modo que se pueda concluir la generación del bloque lógico el cual será implementado haciendo uso de lógica programable en un FPGA (Arreglo de compuertas programable en campo, *Field Programmable Gate Array*). A este tipo de bloque lógico el cual lleva a cabo una aplicación específica que está integrado en un circuito se le conoce como IP Core (Núcleo de Propiedad Intelectual, Intellectual Property Core), que para cumplir ésta función debe ser completamente portable y capaz de ser insertado en la mayoría de máquinas que lleven a cabo un proceso de generación de perfiles.

El presente trabajo está dividido en cuatro capítulos, además de las secciones dedicadas a las Referencias y Apéndices. En éste capítulo se da a conocer un breve apartado acerca de los parámetros que afectan a la dinámica del movimiento en las máquinas-herramientas CNC, los antecedentes, justificación y planteamiento general que dan lugar al

trabajo de tesis presentado. En el capítulo dos se muestra el estado del arte y la fundamentación teórica que provee de sustento a éste trabajo, dando a conocer las herramientas teóricas y tecnológicas que hacen posible la realización del mismo. El tercer capítulo ilustra la metodología seguida para la realización e implementación de los módulos funcionales que constituyen al IP Core. Los resultados y conclusiones que se alcanzaron a lo largo de éste proyecto son presentados en el capítulo cuatro. Por último se incluye una sección de apéndices en la cual se desarrollan todas las descripciones de hardware que fueron generados para el desarrollo de la tesis.

1.1 Antecedentes.

Las máquinas-herramientas han tenido grandes mejoras, desde sus inicios hasta la actualidad, para los procesos industriales son de gran utilidad, pues ayudan en la transformación de la materia prima en productos terminados, a éste proceso se le llama manufactura. Sin embargo, la realización de ésta actividad sin la ayuda de máquinas-herramientas sería muy complicada, considerando que el costo y tiempo de producción es mayor, sin mencionar el hecho de que no habrá repetibilidad en el producto terminado, tomando esto en cuenta como un punto en contra para el fabricante. Todos estos aspectos antes mencionados se deben de tener en consideración si se desea ser competitivo.

En la industria manufacturera, para tener la oportunidad de destacar en un mercado competitivo hoy en día, se requiere contar con una línea de producción la cual tenga la capacidad de elabora piezas con alto grado de repetibilidad, añadiendo calidad, bajo costo de producción y poco tiempo de elaboración en el producto final. Para poder reunir éstas características en un producto, es altamente necesario contar con procesos automatizados, los cuales sean capaces de reproducir fielmente el proceso de producción.

Con el fin de mejorar las líneas de producción, actualmente se utilizan máquinas-herramientas controladas numéricamente que son capaces de realizar el proceso automáticamente. Una máquina-herramienta CNC *(Computer Numerical Control)* es un sistema controlado por motores en sus diferentes ejes, que, con ayuda de una herramienta de corte, puede realizar distintas formas a partir de un material sólido. Al proceso en el cual se remueve material de una pieza base para darle forma y hacerla útil se le conoce como *maquinado*.

El control numérico tiene sus orígenes alrededor de 1947 gracias a Jhon Parsons con la colaboración del Instituto Tecnológico de Massachusetts que iniciaron éste proyecto por contrato de las fuerzas aéreas de los Estados Unidos de América que necesitaba maximizar y mejorar la productividad de sus procesos. Logrando mejor exactitud en la fabricación de

geometrías complejas por medio de una máquina fresadora. Así, la primera generación de máquinas-herramientas fue desarrollada, pero teniendo la desventaja de no contar con una unidad central de procesamiento, pero para los 70's, con el lanzamiento de la computadora personal se pudo incorporar ésta a las máquinas-herramientas, adquiriendo el nombre de *Control Numérico Computarizado* (Alaniz, 2003).

El control numérico computarizado es el control de máquinas-herramientas usando números y letras (Bergen County Technical Schools', 1996), controlando la velocidad de los motores que accionan los ejes de la máquina, facilitando al usuario la programación de órdenes con una secuencia lógica, los cuales podrá interpretar la máquina CNC llevando a cabo el correcto desplazamiento de los ejes o en su defecto de la herramienta de corte. Así para realizar maquinados tridimensionales se hace uso de un método llamado CAM (Fabricación Asistida por Computadora, *Computer Aided Manufactering*) que implica el uso de computadores y tecnología de cómputo para ayudar en todas las fases de la manufactura de un producto, incluyendo la planificación del proceso y la producción, mecanizado, calendarización, administración y control de calidad, con una intervención del operario mínima (Boon et al., 1991).

Algunos de los antecedentes que se han desarrollado hasta el momento en éste ámbito, a tomar en cuenta como una base y punto de partida para el desarrollo de ésta tesis a nivel local, cabe mencionar que la Universidad Autónoma de Querétaro ha hecho un importante contribución en el área de las máquinas-herramientas, pues cuenta con una línea de investigación referente a éste rubro. Entre los proyectos desarrollados a destacar se encuentra (Romero, 2004) que presenta un algoritmo para el procesamiento de señales para la detección de la ruptura de herramienta en sistemas CNC, este trabajo propone un sistema de monitoreo en tiempo real de las condiciones de la herramienta de corte implementado en FPGA. Otro antecedente importante es el desarrollado por (Colín, 2006) que presenta el desarrollo de los bloque funcionales de un controlador PID, haciendo una breve referencia a la generación de algoritmos para un generador de perfiles, cabe mencionar que no se llegó a la implementación. Osornio (2004) realizó una tarjeta controladora de tres ejes en la cual desarrolla los algoritmos para un generador de perfiles básicos y más tarde Osornio (2007)

con base en su trabajo anterior realizó un generador de perfiles para máquinas CNC de Alta velocidad.

1.2 Objetivos.

A lo largo de lo mencionado anteriormente, se sabe que existe una falta de desarrollo en el aspecto de la dinámica de movimiento en las máquinas-herramientas para el mejoramiento de acabado, costos de producción y alargamiento de la vida útil de las máquinas. Uno de los puntos importantes para lograr esta mejora es la implementación de perfiles de movimiento que sean capaces de beneficiar la dinámica de movimiento.

El *objetivo general* de éste trabajo es desarrollar un IP Core que contenga un algoritmo para la generación de perfiles de movimiento, basado en el desarrollo de perfiles polinomiales capaces de controlar la dinámica de movimiento (posición, velocidad, aceleración y jerk también conocido como jaloneo).

Mediante el desarrollo del presente trabajo se pretenden alcanzar *objetivos particulares* que a continuación se dan a conocer:

- Realizar los módulos fundamentales que por medio de un diseño jerárquico den lugar a la generación del IP Core.

- Simular dichos módulos, para la comprobación de su correcto funcionamiento, para una posterior simulación de forma integral, dando lugar al producto terminado.

- Probar el IP Core en una máquina-herramienta, teniendo en cuenta los parámetros necesarios a cumplir para su correcto funcionamiento.

- Generar un manual de usuario para que cualquier persona con conocimientos básicos en el área pueda hacer uso del IP Core.

- Iniciar trámites para el registro de la documentación del IP Core en la institución correspondiente.

1.3 Justificación.

Día a día la evolución tecnológica requiere de un creciente mejoramiento en los procesos de manufactura, como un mejor acabado en las piezas, mejor tiempo de producción y menor costo de manufactura, sin embargo, también se busca el mejoramiento de la integridad de las máquinas-herramientas que hacen posible la fabricación de tales productos.

Para optimizar algunas de las características mencionadas se requiere de la mejora de varios parámetros claves de una máquina-herramienta. Uno de estos parámetros importantes a mejorar es la dinámica de movimiento, siendo más específico el jerk, definido como la derivada de la aceleración.

Idealmente una máquina-herramienta debería de tener los valores de jerk lo más bajos posibles, pues los altos valores de jerk suponen una alta concentración de esfuerzos mecánicos y desgaste en puntos importantes de la máquina-herramienta además de vibraciones en los ejes y como consecuencia causando desgaste de los mecanismos de la misma, teniendo una grave repercusión en el acabado de la pieza y disminución de precisión.

Para hacer una mejora en la dinámica de movimiento se deben sustituir los perfiles de velocidad y aceleración tradicionales que poseen la característica de cambiar el valor de aceleración de manera repentina por perfiles polinomiales que tiene la característica de po-

seer un perfil de aceleración de forma trapezoidal que disminuye en gran medida el cambio brusco de aceleración, así mismo, los valores para el jerk tienden a ser menores traduciéndolos como un mejor comportamiento de la máquina-herramienta. Al ser capaz de reproducir dichos perfiles, se obtienen varios beneficios más indirectamente, pues con la disminución de las vibraciones se puede conseguir movimientos más suaves haciendo posible reducir el error de posición significativamente y además se puede elevar el valor de la velocidad considerablemente mejorando el tiempo de producción, otro punto importante es disminución de alto consumo de corriente demandada por los servos alargando la vida útil de éstos y además disminuyendo la frecuencia del mantenimiento a la máquina-herramienta.

Teniendo en cuenta todo lo anterior, la presente tesis plantea el desarrollo de un IP Core que sea capaz de cubrir la generación de perfiles con las características mencionadas anteriormente, con el objeto de ser un Core genérico capaz de funcionar en la mayoría de máquinas-herramientas que lo requieran y en conjunto con otros módulos IP Core basados en FPGA, como tornos, fresadoras o centros de maquinado. Planteando la implementación de IP Core basado en arquitectura de descripción de hardware VHDL mediante FPGA's. Teniendo en cuenta que éste desarrollo tecnológico será de bajo costo.

Uno de los puntos de gran importancia es dar una alternativa, principalmente a las Pequeñas y Medianas Empresas (PyMES), puesto que es una solución para poder contribuir con éstas, ya que una de las ventajas importantes de éste trabajo es que éste desarrollo tiene la particularidad de ser de bajo costo, pues si se hace una comparativa con Cores comerciales los cuales tiene precios muy elevados, definitivamente el IP Core a desarrollar es sin lugar a duda de menor costo. Teniendo en cuenta que alrededor del 80% de las PyMES fracasan en los primeros cinco años principalmente porque estas pequeñas empresas no pueden sobrevivir a los cambios tecnológicos que el sector empresarial demanda constantemente debido a que se requiere de importantes inversiones económicas. De esta manera, se puede cumplir con una necesidad social, la cual es apoyar al sector empresarial nacional que necesita soporte en el inicio de su vida productiva.

Al término de éste proyecto, se iniciará los procesos para la gestión de derechos de autor de la documentación del core.

1.4 Planteamiento General.

En éste apartado se explica de manera breve la importancia que tiene el IP Core de generación de perfiles en un proceso de maquinado CNC y la forma de interactuar con los demás componentes que lo rodean para una operación satisfactoria.

A continuación se muestra en la figura 1.1 el diagrama a bloques simplificado del proceso de maquinado el cual está constituido por una PC que tiene la función de interfaz para el usuario que está comunicado mediante un protocolo USB hacia el IP Core generador de perfiles y éste a su vez da los datos de entrada a otro Core encargado del control de los ejes de la máquina-herramienta para llevar a cabo el maquinado.

El bloque remarcado es el objeto de estudio de ésta tesis señalando que el IP Core generador de perfiles dará la referencia al controlador de los ejes que ayudará a mejorar la dinámica de movimiento de la máquina-herramienta en base a algoritmos polinomiales de grado superior.

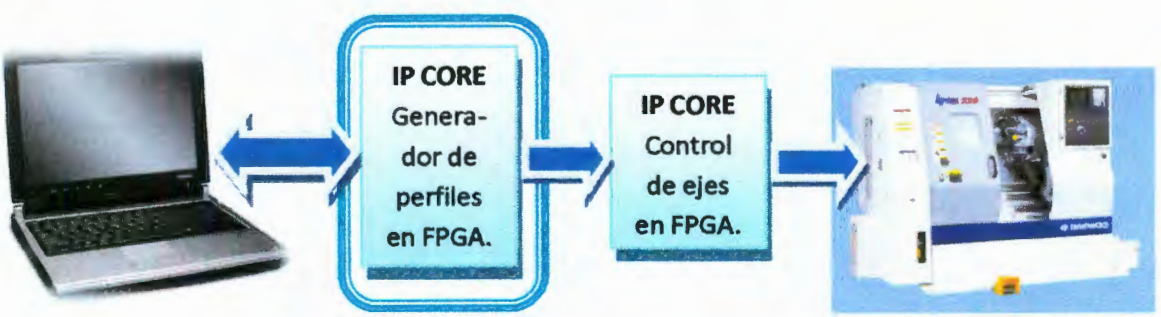


Figura 1.1 Diagrama a bloques general del sistema de maquinaria CNC.

CAPÍTULO 2. REVISIÓN DE LITERATURA.

2.1 Estado del Arte.

Todo sistema mecánico está sujeto a efectos físicos que influyen en el desempeño y el comportamiento dinámico de cualquier mecanismo. Estos efectos se pueden analizar a través de varias propiedades cinemáticas como son: posición, velocidad, aceleración y jerk que en conjunto están definidas como dinámica del movimiento de un sistema mecánico. A la forma en la que se realiza el movimiento se le conoce como perfil de movimiento, describiendo el comportamiento dinámico del sistema.

2.1.1 Sistemas de arquitectura abierta.

Un sistema de arquitectura abierta debe tener las siguientes propiedades: fácil interoperabilidad entre sus componentes, portabilidad, escalable y que sus componentes funcionales sean intercambiables según Pritschow (1992) y Wrigth (1990).

Por otro lado se tiene a los sistemas de control usados en maquinaria comercial CNC a las que se le consideran generalmente de arquitectura cerrada, es decir, solo se muestra el controlador como una caja negra que realiza la función para la cual fue diseñada pero el acceso es denegado, pero en la actualidad se debería optar por el diseño de arquitectura abierta para tener un sistema con más flexibilidad en las aplicaciones.

2.1.2 Dinámica del movimiento en maquinaria.

En una máquina CNC se debe tener en consideración y estudiar detenidamente la dinámica de movimiento, pues es un punto clave que afecta directamente en el tiempo y costo de producción y además en la calidad del mismo.

Para generar un perfil de movimiento que cumpla con las características dinámicas deseables en una máquina CNC se requiere el uso de complejas ecuaciones que deben ser evaluadas dentro de un periodo de muestreo el cual es muy pequeño. Debido a esto es muy complicado utilizar los perfiles optimizados para la reducción del jerk, por esta razón se ha elegido utilizar perfiles simplificados para así poder calcular dichos perfiles en un menor tiempo aunque con graves problemas dinámicos. Las técnicas que tratan de resolver este problema requieren de tiempos de evaluación mayores que el periodo de muestreo máximo permitido en la industria, haciendo poco competitivo su utilización comercialmente. Los perfiles de movimiento que presentan los sistemas CNC comerciales se les conocen como perfiles **convencionales** y tienen serios problemas dinámicos (principalmente en el jaloneo) que aun no han sido resueltos debido a la gran cantidad de recursos necesarios para poder realizar las operaciones del cálculo de perfiles optimizados. Con las técnicas que actualmente están aplicadas a los sistemas comerciales resulta imposible la implementación en tiempo real de perfiles de movimiento que mejoren la dinámica de movimiento en las máquinas-herramientas CNC.

Para realizar un perfil de movimiento se debe de tomar en consideración varios parámetros dinámicos, ya que existe infinidad de maneras para poder realizar un movimiento en una máquina-herramienta CNC.

Posición: Es el lugar geométrico de las posiciones sucesivas por las que pasa un cuerpo en su movimiento (Serway et al., 2004). Debe ser un movimiento suave, sin cambios bruscos. La ecuación matemática que describe a éste parámetro está definida por la tangente hiperbólica, reproduciendo estas características por la máquina-herramienta CNC.

Velocidad: Debe iniciar en cero, incrementarse gradualmente hasta una velocidad máxima (limitada por la fricción cinética y las características de los motores empleados por la maquina CNC) y decrementarse paulatinamente hasta llegar nuevamente a cero manteniendo el mismo signo durante todo el movimiento.

Aceleración: No debe tener cambios bruscos a lo largo de toda la trayectoria dado que es imposible acelerar un motor de forma instantánea, debe de tenerse esto en consideración porque puede originar fuertes vibraciones en los sistemas mecánicos provocando el desgaste mecánico en las uniones y acoplamientos, traducido a errores de precisión en el seguimiento de las trayectorias y por consiguiente un mal acabado en el producto terminado. Otro parámetro a considerar es la inercia mecánica que adquiere la máquina-herramienta CNC al entrar en funcionamiento, por esto se debe de tomar en cuenta que en algunas ocasiones es necesario tener diferentes valores en la aceleración y en la desaceleración (Rivera, 2007).

Jerk: Se define como la razón de cambio de la aceleración con respecto al tiempo y es uno de los parámetros dinámicos más importante a tomar en consideración para éste proyecto. Debido a que a altos niveles de jerk son traducidos a vibraciones mecánicas que afectan la precisión de la máquina CNC además también generan un alto consumo de corriente en los motores. La opción ideal para eliminar todos estos inconvenientes es la máxima reducción del jaloneo en el inicio y final de la trayectoria del movimiento, además de distribuir a lo largo del perfil de movimiento, para así reducir los efectos nocivos tanto en la máquina (alto consumo de corriente en motores y desgaste de piezas mecánicas) como en el acabado de la pieza (Rivera, 2007).

Un sistema CNC convencional cuenta con una computadora capaz de realizar los cálculos necesarios para realizar el movimiento de la máquina, para agilizar el tiempo de procesamiento y tener un mejor tiempo de muestreo se puede optar por mejorar éstas condiciones haciendo uso de un sistema dedicado a la evaluación de los perfiles de movimiento. Este sistema dedicado se encargará de ejecutar exclusivamente una tarea específica, por ejemplo, la ejecución de un algoritmo o la evaluación de una ecuación, que a diferencia de

una computadora no ocurre así, ya que tiene que realizar muchos procesos a la vez haciendo que aumente el tiempo de procesamiento. Para diseñar e implementar un sistema dedicado se debe hacer uso del cómputo reconfigurable del cual se hablará en una sección posterior.

2.1.3 El FPGA como opción para aplicaciones mecatrónicas.

En los últimos años la evolución tecnológica ha impulsado al uso del FPGA en aplicaciones y desarrollo de control debido a que tiene grandes ventajas y además tiene gran confiabilidad para cumplir las necesidades que se necesiten.

Los FPGA's son dispositivos lógicos de propósito general programable por los usuarios, compuestos de bloques lógicos comunicados por conexiones programables. Se basan en una cantidad muy grande de celdas lógicas, muy elementales y una mucho mayor interconectividad que sus contrapartes. Mientras más pequeñas resulten las celdas, mayor aprovechamiento se pueden tener de las mismas, aunque los retardos pueden aumentar (Romero, 2007).

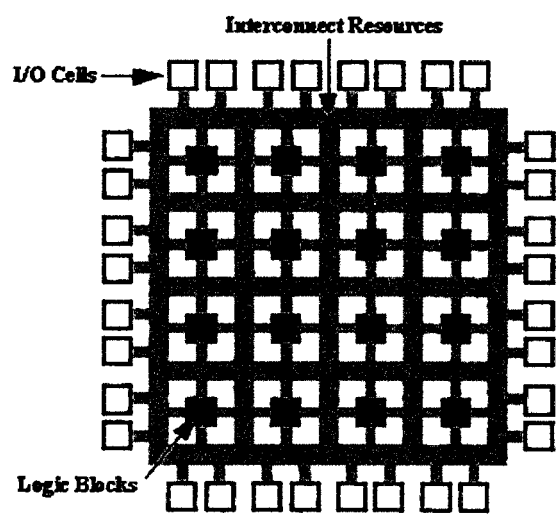


Figura 2.1 Arquitectura interna de un FPGA

El tamaño, estructura, número de bloques y cantidad de las conexiones varían en las distintas arquitecturas. En la figura 2.1 se muestra la arquitectura de un FPGA

Estructura. Arreglo bidimensional de bloques lógicos rodeados por conexiones configurables. Una familia contiene idénticos bloques lógicos y conexiones, pero difieren en el tamaño del arreglo.

Tecnología de programación. Se programa por la carga de celdas de memoria de configuración, que controlan la lógica e interconexiones.

Características. Volatilidad, memoria externa, reprogramabilidad, proceso de fabricación, bajo consumo.

En la actualidad se han realizado una cantidad importante de desarrollos hechos mediante la tecnología FPGA, por ejemplo el desarrollo de un sistema en línea para la detección de ruptura de herramienta en máquinas-herramienta CNC por Romero (2004), además un circuito para aceleración y desaceleración para robots y maquinas herramienta desarrollado en FPGA fue presentado por Wook y Yun-Ki (2002), entre otros. Estos trabajos desarrollados destacan las grandes ventajas que conlleva el uso de lenguaje de descripción de hardware en aplicaciones de ingeniería, algunos de éstos son: el bajo costo, reconfigurabilidad, altas velocidades de procesamiento, etc.

2.1.4 Trabajos reportados en el ámbito de la generación de trayectorias y perfiles de movimiento.

La revisión de literatura para le generación de trayectorias es un punto fundamental para la realización de ésta tesis, pues es la base teórica para el desarrollo del proyecto. A continuación se mencionarán algunos de los trabajos realizados concernientes a éste tópico: Una importante fuente de información a destacar en la actualidad son los dos principales fabricantes de controladores de movimiento para aplicaciones industriales como son PMAC

y Galil Motion, que controlan solamente la dinámica de movimiento hasta la velocidad dejando la aceleración y el jaloneo con severos problemas dinámicos. Algunos modelos disminuyen parcialmente este problema pero a un costo elevado. Los controladores comerciales en su mayoría están basados en DSP's (Digital Signal Processor) y microcontroladores para el cálculo de los algoritmos que se necesitan para realizar el control del movimiento. Erkorkmaz et al. (2001) han desarrollado un algoritmo para generación de trayectoria con limitación del jerk, en ése trabajo destacan la ventaja de generación de perfiles polinomiales en comparación con los perfiles tradicionales (trapezoidales). Mediante una tarjeta DSP TMSC32 plantean la generación de un polinomio quinto (matricial) con el inconveniente de necesitar una gran cantidad de recursos de cómputo para la realización del algoritmo.

Se destaca también los trabajos reportados dedicados a resolver los problemas dinámicos que los controladores comerciales tienen, ejemplo de esto es Yih (2005) presenta el diseño de un filtro jerk lineal para CNC con la finalidad de mejorar el desempeño en el seguimiento de los servomotores, implementando una especie de filtro de tercer orden. Cabe mencionar que en éste trabajo ya hace mención al problema del tiempo de cómputo, para su solución hace uso de tres algoritmos usando tres buffers circular para reducir la demanda de hardware. Tounsi et al. (2003) utilizan filtros digitales para suavizar los cambios bruscos en los perfiles de aceleración convencionales y mencionan varios de los problemas que originan éste movimiento en un maquinado CNC, como grandes fuerzas de corte y sobrecarga en la herramienta de corte, teniendo en cuenta que éste trabajo solamente está centrado en la aceleración y no en el jaloneo, otro problema a destacar es determinar con exactitud el tiempo que tarda en realizar el movimiento.

Wook y Yun-Ki (2002) utilizan una técnica para mejorar la dinámica de movimiento en los controladores comerciales llamada convolución digital en la cual el algoritmo requiere pocas operaciones, haciéndolo una técnica fácil de implementación para aplicaciones que deben hacerse en tiempo real. La desventaja de la convolución digital es que los perfiles obtenidos tienen un intervalo igual en la aceleración y en la desaceleración.

Gasparetto y Zanotto (2006) realizan un trabajo para la planificación de trayectorias suaves en un robot manipulador mediante perfiles polinomiales, haciendo énfasis en la limitación del jerk para la reducción del daño estructural al evitar frecuencias de resonancia que pueden generar errores. En dicha investigación trabajan con la parte proporcional e integral del jaloneo cuadrado y con un polinomio Spline de quinto orden, cabe destacar que solamente presentan resultados mediante simulación. Zhiwei et al. (2005) tiene como objetivo generar un movimiento suave, afirmando la necesidad de seleccionar el perfil de movimiento, en un sistema de control adaptable de movimiento. El algoritmo desarrollado solo ha sido probado en simulaciones mediante Simulink de Matlab.

A pesar de las técnicas utilizadas mencionadas en párrafos anteriores como la convolución digital y los filtros digitales, se observa que resulta extremadamente difícil tener un buen control en la dinámica del movimiento. Una alternativa para la mejora de la generación de perfiles que permita dicho control es mediante la utilización de perfiles polinomiales que entre mayor es el grado del polinomio, la dinámica de movimiento es mejor, teniendo una continuidad en los parámetro del movimiento (velocidad, aceleración y jerk), pero, una de las desventajas observadas en ésta técnica es la necesidad de incrementar los recursos computacionales de manera exponencial a su vez que el grado del polinomio aumenta (Wook et al., 2000). Debido a lo antes mencionado se han realizado varias propuestas usando perfiles polinomiales para la mejora de la dinámica del movimiento, aunque en la mayoría de los casos las investigaciones sólo han llegado hasta la etapa teórica por la dificultad que presenta hacer pruebas en tiempo real.

Debido a la desventaja de la utilización de perfiles polinomiales, en algunas ocasiones se opta por la evaluación fuera de línea, que consta en evaluar el polinomio antes de ejecutar el perfil. Mizoshita et al. (1996) utilizan un perfil polinomial de quinto grado que se basa en el jerk logrando un consumo de corriente menor a los perfiles convencionales y además decrementa las vibraciones generadas.

2.2 Tipos de Máquinas-Herramientas.

Actualmente todas las máquinas-herramientas están en todos los procesos industriales permitiendo la realización de piezas complejas, introduciendo estos principios en gran variedad de robots industriales, y aún más, con el lanzamiento al mercado de las computadoras digitales marcaron un punto decisivo en la mejora de tiempo y costo de producción. En la industria existe gran diversidad de máquinas-herramientas. Algunos tipos de máquinas-herramientas más comunes son los tornos, fresadoras y centros de maquinado, a continuación se realiza una breve descripción de éstos:

Torno. Máquina-herramienta que permite mecanizar piezas de forma geométrica de revolución. Estas máquinas-herramienta operan haciendo girar la pieza a mecanizar (sujeta en el cabezal o fijada entre los puntos de centraje) mientras una o varias herramientas de corte son empujadas en un movimiento regulado de avance contra la superficie de la pieza, cortando la viruta de acuerdo con las condiciones tecnológicas de mecanizado adecuadas (DRAE). En la Figura 2.2 se muestra un torno CNC.

Figura 2.2 Torno CNC

Fresadora. Es una máquina-herramienta utilizada para realizar mecanizados por arranque de viruta mediante el movimiento de una herramienta rotativa de varios filos de corte denominada fresa. En las fresadoras tradicionales, la pieza se desplaza acercando las zonas a mecanizar a la herramienta, permitiendo obtener formas diversas, desde superficies planas a otras más complejas (Enciclopedia Ciencia y Técnica, 1984). En la Figura 2.3 se muestra una fresadora CNC.

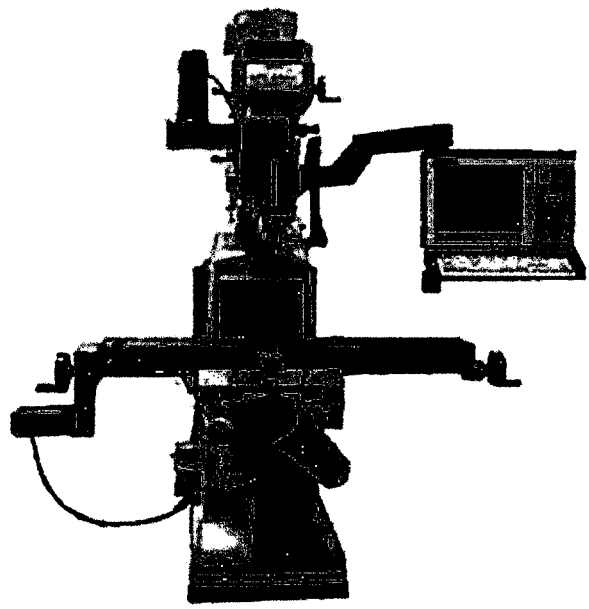


Figura 2.3 Fresadora CNC

Centro de Maquinado. Máquina-herramienta que es capaz de combinar diversas herramientas de corte, seleccionándolas dependiendo de los diferentes tipos de maquinados que se necesiten. En Figura 2.4 se muestra un centro de maquinado CNC.

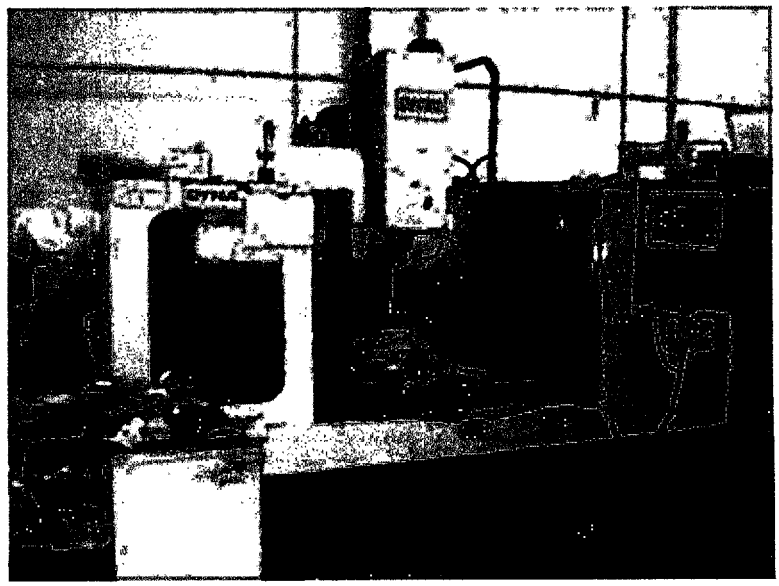


Figura 2.4 Centro de Maquinado CNC

Después de décadas de desarrollo en ésta área de estudio, se puede observar que falta mucho por perfeccionar, ya que día a día, las exigencias de la industria son mayores, siempre con la tendencia de mejorar costos de producción y reducción de tiempo de procesamiento sin descuidar la calidad del producto terminado.

2.3 Dinámica de Movimiento.

El generador de perfiles de movimiento es de gran importancia en una máquina-herramienta CNC pues es el que define la dinámica de movimiento, debido a esto, el generador de perfiles debe ser capaz de proveer una posición de referencia con características suaves (ver 2.1.1) para poder garantizar movimiento precisos y generar las mínimas vibraciones posibles.

Como ya se ha comentado en la sección 2.1.1 los parámetros de la dinámica de movimiento son: posición, velocidad, aceleración y jerk (X, V, A, J, respectivamente) que tienen una relación entre sí mostrada en la Tabla 2.1. Las ecuaciones en la columna de la izquierda muestran la dinámica del movimiento de forma continua, es decir, en función del tiempo (t), y las ecuaciones de la columna a la derecha se presentan en forma discreta (k). Teniendo en cuenta que el uso de las ecuaciones en tiempo discreto se usa para el control de movimiento, ya que actualmente los controladores trabajan de manera digital.

Continuo	Discreto
$X(t) = \int_0^t V(t)dt$	$X(k) = V(k) + X(k-1)$
$V(t) = \frac{d}{dt}X(t)$ o $V(t) = \int_0^t A(t)dt$	$V(k) = X(k) - X(k-1)$ o $V(k) = A(k) + V(k-1)$
$A(t) = \frac{d}{dt}V(t)$ o $A(t) = \int_0^t J(t)dt$	$A(k) = V(k) - V(k-1)$ o $A(k) = J(k) + A(k-1)$
$J(t) = \frac{d}{dt}A(t)$	$J(k) = A(k) - A(k-1)$

Tabla 2.1 Ecuaciones generales de la dinámica de movimiento.

La **posición** debe ser una curva suave similar a la tangente hiperbólica pues este es el movimiento más común en la naturaleza, esta restricción es fácilmente cubierta por los controladores de posición convencionales. En la Figura 2.5 se muestra el perfil de posición obtenido con un controlador convencional.

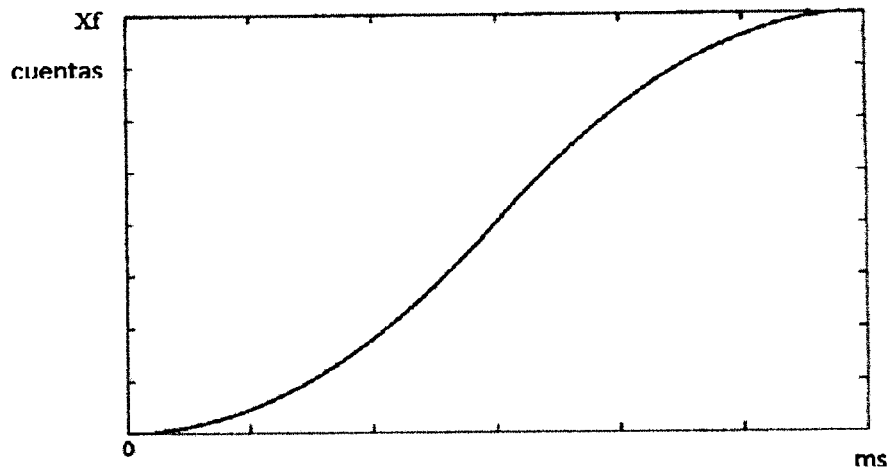


Figura 2.5 Perfil de posición convencional.

La **velocidad**, está limitada por las características del sistema, debe empezar y terminar en cero además de tener el mismo signo a lo largo de la trayectoria. La curva de velocidad se debe de incrementar gradualmente hasta una velocidad pico o máxima y después debe descender de igual forma hasta llegar nuevamente a cero para poder reproducir y cumplir con las restricciones de posición, es decir, la posición obtenida debe ser similar a la tangente hiperbólica. A continuación se observa en la Figura 2.6 la curva convencional de un perfil de velocidad, notando que el perfil cubre con las características mencionadas.

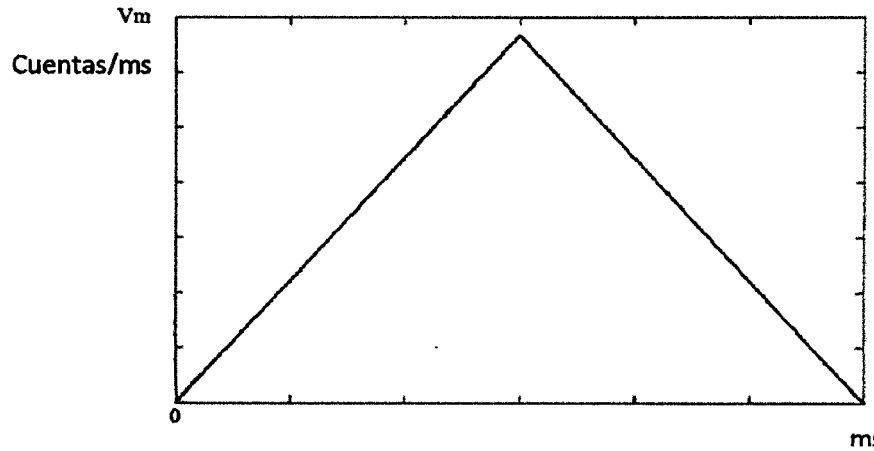


Figura 2.6 Perfil de velocidad convencional.

La **aceleración** no debe tener cambios bruscos en la trayectoria ya que son imposibles de seguir fielmente por el controlador PID debido a las limitaciones físicas de los servomotores y las partes mecánicas, además esto evita vibraciones excesivas y errores severos en el seguimiento (Altintas, 2000). En la Figura 2.7 se muestra el perfil de aceleración que se encuentra comúnmente en los controladores comerciales, se observa que su comportamiento es escalonado, esto es imposible de reproducir en el mundo real debido a las limitaciones físicas de los sistemas mecánicos lo cual provoca, como ya se ha mencionado, vibraciones y severos errores en el seguimiento del perfil.

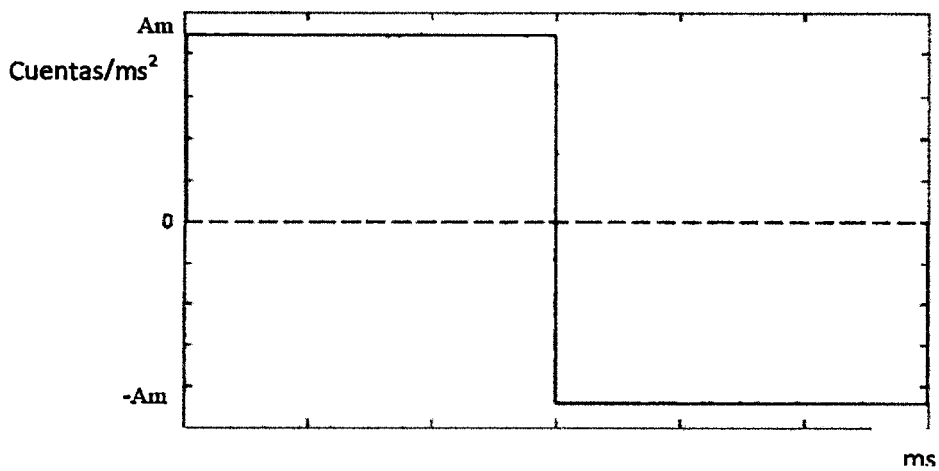


Figura 2.7 Perfil de aceleración convencional.

El **Jerk**, se define como la razón de cambio de la aceleración, este parámetro es una de las principales características dinámicas que se debe controlar ya que es directamente responsable del consumo de corriente en los servomotores, además de generar la vibración y con ello el desgaste mecánico, produciendo errores de precisión en la trayectoria.

El jerk es la variable dinámica que afecta en mayor medida tanto los sistemas electrónicos como los mecánicos, idealmente se esperaría que fuera cero pero en la realidad es imposible reducir el valor del jerk a cero, aumentando el problema del jerk en maquinados de alta velocidad (MAV). En teoría el jerk debe de iniciar en cero y terminar de la misma forma para poder mejorar la dinámica de movimiento (Chang et al., 2006). En la Figura 2.8 se muestra el perfil de jerk presente en los controladores convencionales. En este perfil se puede observar que los valores son considerablemente elevados y en forma de impulso.

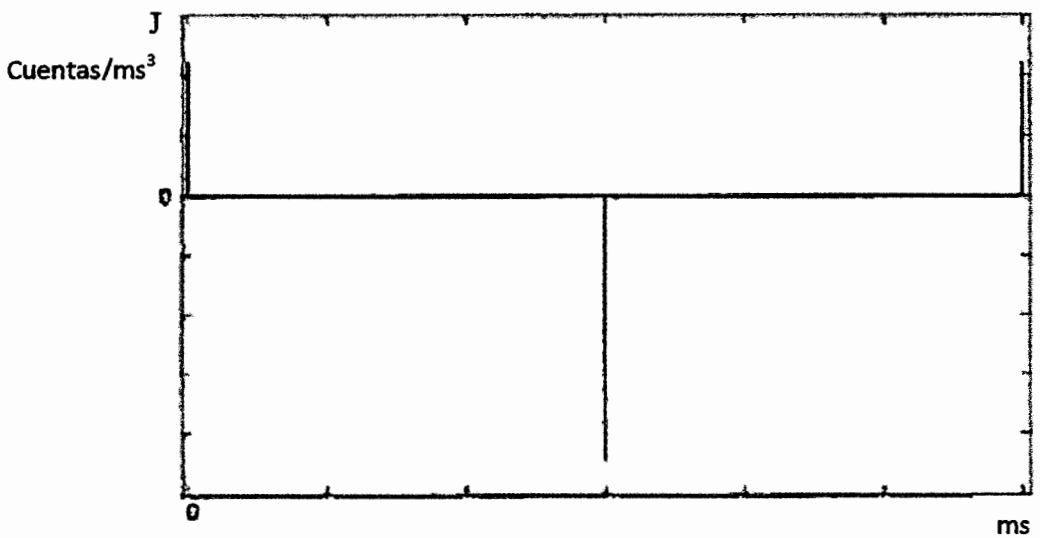


Figura 2.8 Perfil de jerk convencional.

Al hacer un análisis de los anteriores perfiles de movimiento convencionales, se puede observar que dejan mucho que desear en cuando a la dinámica del movimiento se refiere. Para poder corregir estos perfiles se debe hacer un estudio de la dinámica del movimiento que existe en la naturaleza y así poder emplear esta dinámica en máquinas CNC. Flash y Hogan (1985) hacen un estudio de dinámica de movimiento en brazos humanos,

estos movimientos son de gran interés pues consumen poca energía, para hacer una aproximación de éste movimiento se hace uso de una ecuación polinomial de quinto grado (Hogan, 1984), aunque tiene la desventaja de que estos perfiles de movimiento son asimétricos, es decir, tienen un intervalo de aceleración y desaceleración diferente, trabajos de este tipo puede ser un fundamento para mejorar los perfiles de movimiento en maquinaria CNC.

Como ya antes se ha mencionado la dinámica de movimiento presente en la naturaleza se puede aproximar a la tangente hiperbólica. Dicho lo anterior, se puede observar en la Figura 2.9 la dinámica de movimiento en posición que tiene un trazo suave en el inicio y en el fin del movimiento. Si se hace una comparativa entre el perfil convencional de posición y éste, se puede observar una diferencia significativa.

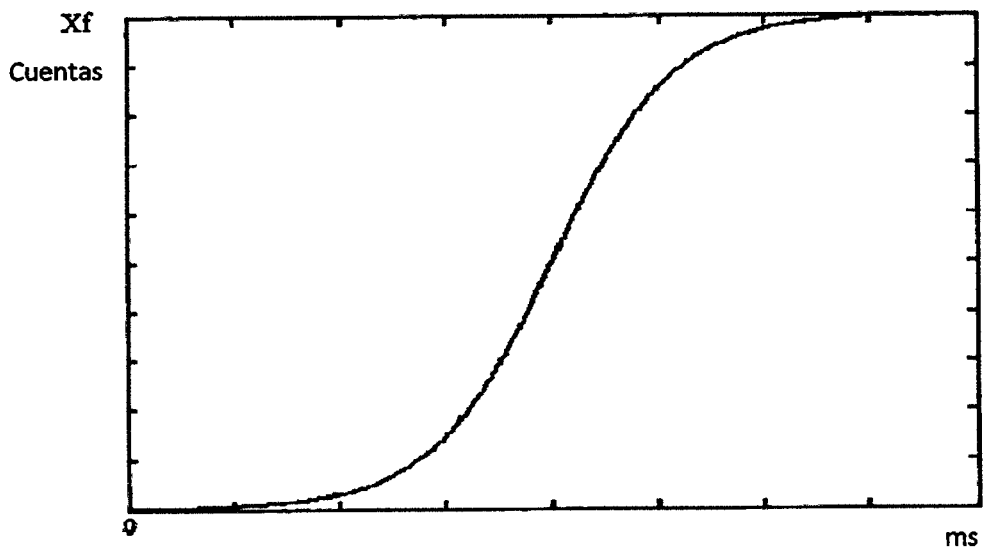


Figura 2.9 Perfil de posición presente con mejor dinámica.

Al igual que el perfil de posición, el perfil de velocidad de la Figura 2.10 también tiene un trazo suave al inicio y al fin del movimiento. Haciendo una comparativa de éste perfil con el perfil de velocidad convencional tiene un comportamiento mucho más suave, ya que el perfil convencional tiene un comportamiento triangular.

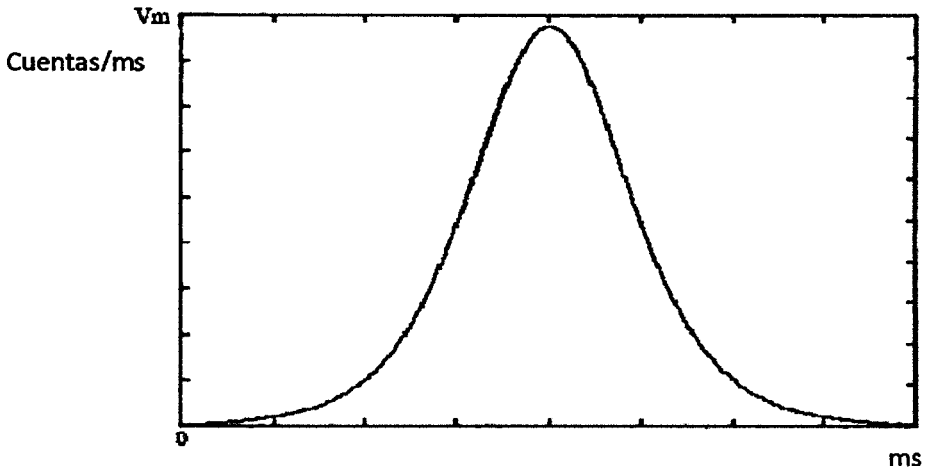


Figura 2.10 Perfil de velocidad con mejor dinámica.

La aceleración idealmente debe cambiar paulatinamente a lo largo del movimiento, haciendo más fácil reproducir estos cambios por el controlador. Si se analiza el perfil de aceleración convencional (Figura 2.4) se puede observar que tiene cambios bruscos de forma escalonada, a su vez, el perfil de aceleración de la Figura 2.11 tiene cambios de aceleración y desaceleración suaves.

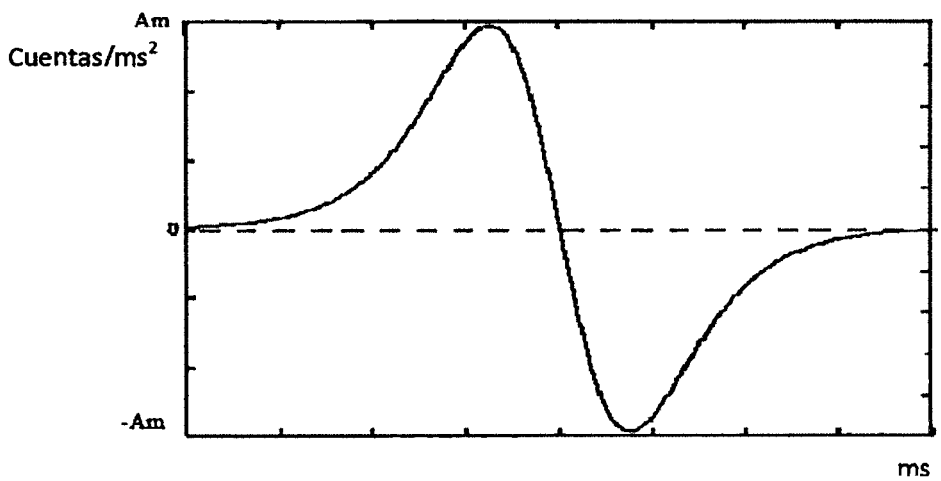


Figura 2.11 Perfil de aceleración con mejora en la dinámica.

Para poder eliminar o disminuir al máximo las vibraciones, se necesita tener un perfil de jerk sin cambios bruscos, si se analiza el perfil de jerk convencional se puede obser-

var (Figura 2.8) que son impulso, mientras que en el perfil de jerk en la naturaleza (Figura 2.12) los incrementos son mucho más distribuidos a los largo del movimiento, disminuyendo los niveles de jerk en el inicio y en el fin del movimiento y por consiguiente disminuye las vibraciones y los efectos nocivos antes mencionados. La máxima magnitud de jerk puede ser hasta 500 veces más grande que el de un perfil de dinámica completamente controlada.

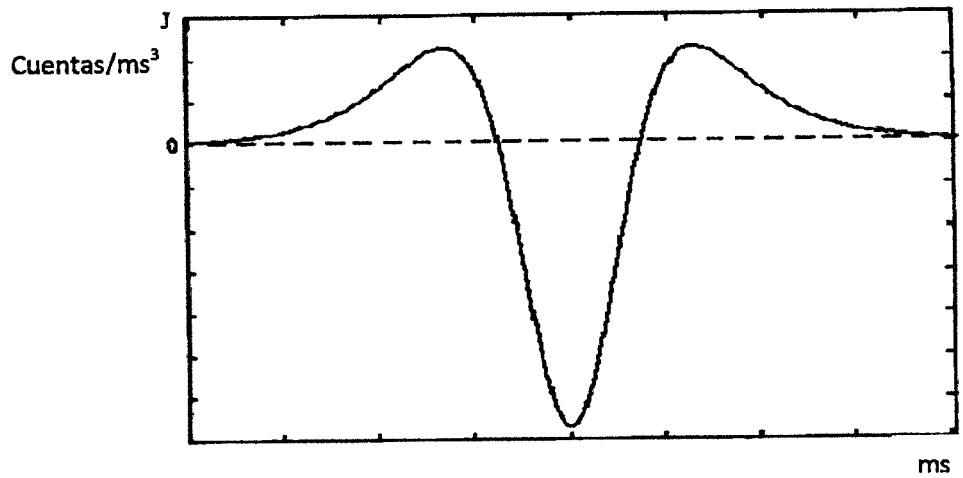


Figura 2.12 Perfil de jerk con mejor dinámica.

Como se ha podido observar anteriormente, los perfiles de movimiento tienen una mejora en gran magnitud en comparación con los perfiles de los controladores comerciaes, la razón por la cual este tipo de perfiles no han sido implementados completamente a los controladores comerciales es que necesitan una gran cantidad de recursos computacionales para poder evaluar y hacer los cálculos de complejas operaciones en sistemas de tiempo real. Otro punto importante que se debe considerar en la maquinaria CNC es las variables no lineales como la inercia, fricción, fuerza de corte, etc. Hay muy pocas o inexistentes bases teóricas que toman en cuenta las variables no lineales con respecto al diseño de generadores de perfiles de movimientos. Por tal razón los perfiles de movimiento comúnmente son controlados a partir de velocidad o aceleración y en algunos casos jerk.

2.4 Diseño Polinomial para la Optimización de la Dinámica del Movimiento.

Las aproximaciones polinomiales para la generación de perfiles y trayectorias en sistemas CNC son de gran relevancia para el área debido a que permiten controlar las características dinámicas del movimiento en sus cuatro principales parámetros: posición, velocidad, aceleración y jaloneo.

El objetivo final de la generación de perfiles es obtener un movimiento que comienza en una posición inicial y termina en una posición final, realizado en un tiempo determinado. Bajo estas especificaciones, exclusivamente, el movimiento puede ser realizado de muchas formas puesto que el problema tiene múltiples soluciones; sin embargo, para cuestiones prácticas es muy conveniente tener algunos parámetros adicionales para el diseño del perfil de movimiento como son: velocidad máxima, aceleración máxima y reducción del jaloneo.

La velocidad máxima es un parámetro dinámico que tiene que ver con las características tecnológicas de los servo-motores y las bancadas o grados de libertad de movimiento. Generalmente este parámetro no puede ser modificado porque depende de especificaciones de fabricación.

La aceleración máxima también es un parámetro dinámico que depende de las especificaciones de fabricación. Por otro lado, el jaloneo, si bien es una consecuencia directa del movimiento, puede ser manejado para que tenga ciertas características particulares que permitan al sistema completo reducir sus efectos indeseables como son: exceso de inercia, exceso de torsión, desgaste de rodamientos, entre otros.

La problemática principal de la generación de perfiles es utilizar todos los parámetros disponibles para lograr un movimiento suave y con pocos efectos secundarios para el

control de posición de un servo-mecanismo y que sea realizado en tiempo real. Si bien los polinomios son funciones cuyo cálculo no es particularmente complejo desde el punto de vista computacional, tampoco es un cálculo trivial y si no es llevado a cabo de manera correcta pueden subestimarse los errores o el tiempo de cómputo puede ser tan alto que impida su realización en tiempo real.

2.4.1 Propiedades básicas de los polinomios.

2.4.1.1 Definición de un polinomio.

Un polinomio con coeficientes reales de grado n es una sumatoria de n+1 términos que contienen potencias crecientes de la variable x desde 0 hasta n y con la restricción que el coeficiente de la potencia mayor es diferente de cero.

$$A(x)=\sum_{p=0}^{n} a_p x^p \qquad (2.1)$$

2.4.1.2 Teorema de complejidad.

Para caracterizar y definir de manera completa a un polinomio de grado n se necesitan n+1 parámetros.

El teorema de complejidad, nos dice que para poder caracterizar completamente a un polinomio de grado n se necesitan n+1 parámetros que pueden ser dados de diferentes formas como se muestra a continuación.

Forma normal. Se denomina forma normal de un polinomio a la caracterización mediante coeficientes de potencias crecientes de la variable x, comenzando en cero como se presenta en (2.1).

Forma vectorial. La ecuación (2.1) puede ser descrita también como el producto de un vector de coeficientes por otro vector de potencias unitarias de x como se presenta en (2.2).

$$A(x) = (a_0 \quad a_1 \quad \dots \quad a_n)\begin{pmatrix} 1 \\ x \\ \dots \\ x^n \end{pmatrix} \qquad (2.2)$$

Forma factorizada. Por el teorema fundamental del álgebra un polinomio puede ser representado como el producto de los ceros del polinomio como se muestra en (2.3). El coeficiente común es diferente de cero. Los ceros pueden ser reales o complejos conjugados.

$$A(x) = a_n(x - z_1)(x - z_2)\dots(x - z_n) \qquad (2.3)$$

2.4.1.3 Polinomios unitarios.

Se denomina polinomio unitario al polinomio cuyo coeficiente a_n es igual a 1.

2.4.1.4 Valor inicial y valor final.

El límite en menos infinito (valor inicial) y en más infinito (valor final) de un polinomio es infinito y tienen el mismo signo si el polinomio es de grado par y el signo contrario si el polinomio es de grado impar.

2.4.1.5 Máximos y mínimos locales.

Un polinomio de grado *n* tiene hasta *n-1* máximos y mínimos locales. La relevancia de esta propiedad consiste en el número de veces que se puede cambiar el signo de la pendiente del polinomio.

2.4.1.6 Invarianza morfológica.

Desde el punto de vista topológico, un polinomio se caracteriza por el número de cambios de signo en la pendiente (máximos y mínimos locales).

Se entiende por invarianza morfológica a las transformaciones no-lineales que conservan las características topológicas del polinomio.

Translación horizontal. Desplazar las características topológicas de un polinomio a lo largo del eje horizontal.

$$x^* \rightarrow x - h \tag{2.4}$$

Traslación vertical. Desplazar las características topológicas de un polinomio a lo largo del eje vertical.

$$a_0 \rightarrow a_0 - v \tag{2.5}$$

Escalamiento horizontal. Escalar las características topológicas de un polinomio a lo largo del eje horizontal.

$$x^* \rightarrow \frac{x}{K} \tag{2.6}$$

Escalamiento vertical. Escalar las características topológicas de un polinomio a lo largo del eje vertical.

$$A(x) \rightarrow \alpha A(x) \tag{2.7}$$

2.4.1.7 Derivadas.

El grado de un polinomio disminuye en 1 cada vez que se obtiene su derivada.

$$\frac{d}{dx}A(x) = \sum_{p=0}^{n-1} b_p x^p \tag{2.8}$$

La n-ésima derivada de un polinomio es igual a una constante.

$$\frac{d^n}{dx^n}A(x) = c \tag{2.9}$$

La (n+1)-ésima derivada de un polinomio es igual a cero.

$$\frac{d^{n+1}}{dx^{n+1}}A(x) = 0 \tag{2.10}$$

2.4.2 Polinomios en tiempo discreto.

La versión discretizada de un polinomio se encuentra dada por (2.11) donde k es cualquier entero.

$$A(k) = \sum_{p=0}^{n} a_p k^p \tag{2.11}$$

2.4.2.1 Diferencia discreta.

La diferencia discreta de un polinomio está dada por (2.12).

$$\Delta A(k) = A(k) - A(k-1) \tag{2.12}$$

El grado de un polinomio disminuye en 1 cada vez que se obtiene su diferencia discreta.

$$\Delta A(k) = \sum_{p=0}^{n-1} b_p k^p \tag{2.13}$$

La n-ésima diferencia discreta de un polinomio es igual a una constante.

$$\Delta^n A(k) = c \tag{2.14}$$

La $(n+1)$-ésima diferencia discreta de un polinomio es igual a cero.

$$\Delta^{n+1} A(k) = 0 \tag{2.15}$$

2.4.2.2 Integración discreta.

De (2.12) se puede obtener una relación de recursividad para obtener la integración discreta de un polinomio a partir de su diferencia.

$$A(k) = \Delta A(k) + A(k-1) \tag{2.16}$$

2.4.2.3 Reconstrucción por diferencias.

Si se conocen las diferencias de orden n de un polinomio en un punto dado y el valor mismo del polinomio en ese punto, se puede reconstruir computacionalmente el polinomio en forma completa mediante relaciones de recursividad.

$$\begin{aligned}\Delta^{n-1}A(k) &= \Delta^n A(k) + \Delta^{n-1}A(k-1)\\ \Delta^{n-2}A(k) &= \Delta^{n-1}A(k) + \Delta^{n-2}A(k-1)\ldots\\ \ldots\Delta A(k) &= \Delta^2 A(k) + \Delta A(k-1)\\ A(k) &= \Delta A(k) + A(k-1)\end{aligned} \qquad (2.17)$$

Nótese que para la reconstrucción son necesarios los puntos: $A(k-1), \Delta A(k-1), \ldots, \Delta^{n-1}A(k-1), \Delta^n A(k-1)$. Por la ecuación (2.14) se tiene que: $\Delta^n A(k) = c = \Delta^n A(k-1)$.

2.4.3 Polinomios truncados.

Desde el punto de vista práctico para la generación de perfiles y trayectorias, los polinomios en su versión discreta solamente son útiles en un segmento de todo su rango, que básicamente incluye el tiempo para ejecutar el movimiento especificado. Bajo esta perspectiva resulta interesante explorar los fenómenos de diferencia e integración discreta de los polinomios en su versión truncada.

La versión truncada del polinomio puede verse como (2.18) y en forma cerrada puede expresarse como en (2.19).

$$A_K(k) = \begin{cases} A(k) & 0 < k \le K \\ 0 & de\ otra\ forma \end{cases} \qquad (2.18)$$

$$A_K(k) = A(k)[1 - u(-k) - u(k-K-1)] \qquad (2.19)$$

Tomando ahora las diferencias de la versión truncada del polinomio se obtiene sucesivamente:

$$\Delta A_K(k) = A_K(k) - A_K(k-1) \qquad (2.20)$$

$$\Delta^2 A_K(k) = A_K(k) - 2A_K(k-1) + A_K(k-2) \tag{2.21}$$

$$\Delta^r A_K(k) = A_K(k) - \binom{r}{1} A_K(k-1) + \binom{r}{2} A_K(k-2) + \ldots + (-1)^j \binom{r}{j} A_K(k-j) + \ldots + (-1)^r \binom{r}{r} A_K(k-r) \tag{2.22}$$

Donde la distribución binomial está dada por:

$$\binom{r}{j} = \frac{r!}{j!(r-j)!} \tag{2.23}$$

Ahora, la versión desplazada del polinomio truncado puede ser simplificada como:

$$A_K(k-1) = A(k-1)[1 - u(-k+1) - u(k-K-2)] \tag{2.24}$$

$$u(-k+1) = u(-k) + \delta(k-1) \tag{2.25}$$

$$u(k-K-2) = u(k-K-1) - \delta(k-K-1) \tag{2.26}$$

$$\delta(k) = \begin{cases} 1 & k = 0 \\ 0 & otherwise \end{cases} \tag{2.27}$$

Sustituyendo (2.25) a (2.27) en la ecuación (2.24) se obtiene:

$$A_K(k-1) = A(k-1)[1 - u(-k) - u(k-K-1)] - A(k-1)\delta(k-1) + A(k-1)\delta(k-K-1) \tag{2.28}$$

Y dado que:

$$A(k-1)\delta(k-1) = A(0)\delta(k-1) \tag{2.28}$$

$$A(k-1)\delta(k-K-1) = A(K)\delta(k-K-1) \tag{2.30}$$

Se obtiene la ecuación (2.31):

$$A_K(k-1) = A(k-1)[1 - u(-k) - u(k-K-1)] - A(0)\delta(k-1) + A(K)\delta(k-K-1) \tag{2.31}$$

Para el desplazamiento de segundo orden se tiene:

$$\begin{aligned} A_K(k-2) &= A(k-2)[1-u(-k+1)-u(k-K-2)]-A(0)\delta(k-2)+A(K)\delta(k-K-2) \\ &= A(k-2)[1-u(-k)-u(k-K-1)-\delta(k-1)+\delta(k-K-1)]+ \\ &\quad -A(0)\delta(k-2)+A(K)\delta(k-K-2) \\ &= A(k-2)[1-u(-k)-u(k-K-1)]-[A(-1)\delta(k-1)+A(0)\delta(k-2)]+ \\ &\quad +[A(K-1)\delta(k-K-1)+A(K)\delta(k-K-2)] \end{aligned} \tag{2.32}$$

Y en general, para el desplazamiento de orden r se puede obtener (2.33):

$$\begin{aligned} A_K(k-r) = A(k-r)[1-u(-k)-u(k-K-1)]-\sum_{j=1}^{r}A(1-j)\delta(k-1-r+j)+ \\ +\sum_{j=1}^{r}A(K+1-j)\delta(k-K-1-r+j) \end{aligned} \tag{2.33}$$

Sustituyendo estos resultados en las expresiones (2.20) a(2.22) de las diferencias se puede obtener:

$$\begin{aligned} \Delta^r A_K(k) = &\left[A(k)-\binom{r}{1}A(k-1)+\ldots+(-1)^r\binom{r}{r}A(k-r)\right][1-u(-k)-u(k-K-1)]+ \\ &-\left[-\binom{r}{1}\sum_{j=1}^{1}A(1-j)\delta(k-2+j)+\binom{r}{2}\sum_{j=1}^{2}A(1-j)\delta(k-3+j)+\ldots+\right. \\ &\left.+(-1)^r\binom{r}{r}\sum_{j=1}^{r}A(1-j)\delta(k-1-r+j)\right]+\left[-\binom{r}{1}\sum_{j=1}^{1}A(K+1-j)\delta(k-K-2+j)+\right. \\ &\left.+\binom{r}{2}\sum_{j=1}^{2}A(K+1-j)\delta(k-K-3+j)+\ldots+(-1)^r\binom{r}{r}\sum_{j=1}^{r}A(K+1-j)\delta(k-K-1-r+j)\right] \end{aligned} \tag{2.34}$$

$$\begin{aligned}\Delta^r A_K(k) = \Delta^r A(k)[1-u(-k)-u(k-K-1)] - \Bigg[-\binom{r}{1}\sum_{j=1}^{1} A(1-j)\delta(k-2+j) + \\ + \binom{r}{2}\sum_{j=1}^{2} A(1-j)\delta(k-3+j) + \ldots + (-1)^r \binom{r}{r}\sum_{j=1}^{r} A(1-j)\delta(k-1-r+j)\Bigg] + \\ + \Bigg[-\binom{r}{1}\sum_{j=1}^{1} A(K+1-j)\delta(k-K-2+j) + \binom{r}{2}\sum_{j=1}^{2} A(K+1-j)\delta(k-K-3+j) + \\ + \ldots + (-1)^r \binom{r}{r}\sum_{j=1}^{r} A(K+1-j)\delta(k-K-1-r+j)\Bigg]\end{aligned} \tag{2.35}$$

Cuando $r = n+1$, por la propiedad (2.15), (2.35) se puede reducir a (2.36)

$$\begin{aligned}\Delta^{n+1} A_K(k) = -\Bigg[-\binom{n+1}{1}\sum_{j=1}^{1} A(1-j)\delta(k-2+j) + \binom{n+1}{2}\sum_{j=1}^{2} A(1-j)\delta(k-3+j) + \ldots + \\ + (-1)^{n+1}\binom{n+1}{n+1}\sum_{j=1}^{n+1} A(1-j)\delta(k-n-2+j)\Bigg] + \Bigg[-\binom{n+1}{1}\sum_{j=1}^{1} A(K+1-j)\delta(k-K-2+j) + \\ + \binom{n+1}{2}\sum_{j=1}^{2} A(K+1-j)\delta(k-K-3+j) + \ldots + (-1)^{n+1}\binom{n+1}{n+1}\sum_{j=1}^{n+1} A(K+1-j)\delta(k-K-n-2+j)\Bigg]\end{aligned} \tag{2.36}$$

Reescribiendo (2.36) como dos conjuntos de deltas de Kronecker, donde el primer conjunto tiene la parte de generación y el segundo conjunto la parte de disolución, esta ecuación puede ser descrita por (2.37).

$$\Delta^{n+1} A_K(k) = \sum_{j=1}^{n+1} G_j \delta(k-j) + \sum_{j=1}^{n+1} D_j \delta(k-K-j) \tag{2.37}$$

Donde:

$$G_1 = -\Bigg[-\binom{n+1}{1}A(0) + \binom{n+1}{2}A(-1) - \binom{n+1}{3}A(-2) + \ldots + (-1)^{n+1}\binom{n+1}{n+1}A(-n)\Bigg] \tag{2.38}$$

$$G_2 = -\Bigg[\binom{n+1}{2}A(0) - \binom{n+1}{3}A(-1) + \ldots + (-1)^{n+1}\binom{n+1}{n+1}A(-n+1)\Bigg] \tag{2.39}$$

Y en general:

$$G_j = -\sum_{i=j}^{n+1}(-1)^i\binom{n+1}{i}A(j-i) \tag{2.40}$$

También se tiene que:

$$D_1 = -\binom{n+1}{1}A(K)+\binom{n+1}{2}A(K-1)-\binom{n+1}{3}A(K-2)+...+(-1)^{n+1}\binom{n+1}{n+1}A(K-n) \tag{2.41}$$

$$D_2 = \binom{n+1}{2}A(K)-\binom{n+1}{3}A(K-1)+...+(-1)^{n+1}\binom{n+1}{n+1}A(K-n+1) \tag{2.42}$$

$$D_j = -\sum_{i=j}^{n+1}(-1)^i\binom{n+1}{i}A(K+j-i) \tag{2.43}$$

Con lo que se demuestra que se necesitan $2(n+1)$ parámetros para caracterizar un polinomio truncado de grado $n+1$ con K muestras localizadas en $k = 1,2,3,...,K$.

Dado que (2.37) es una forma analítica, cerrada, de la parametrización de la n+1-ésima diferencia del polinomio truncado de grado n, la reconstrucción del polinomio se puede dar mediante integraciones sucesivas de ésta forma.

2.4.4 Funciones polinomiales a trazos.

La forma general de la parametrización de un polinomio truncado es la definición a trazos de una función con segmentos polinomiales, no necesariamente del mismo grado. La forma discretizada de esta función a trazos está dada por (2.44).

$$W_K(k) = \begin{cases} A_1(k) & 0 < k \le K_1 \\ A_2(k-K_1) & K_1 < k \le K_1+K_2 \\ A_3(k-K_1-K_2) & K_1+K_2 < k \le K_1+K_2+K_3 \\ \cdots & \cdots \\ A_p\left(k-\sum_{i=1}^{p-1}K_i\right) & \sum_{i=1}^{p-1}K_i < k \le \sum_{i=1}^{p}K_i \\ 0 & otherwise \end{cases} \tag{2.44}$$

La forma cerrada de (2.44) se obtiene haciendo uso de superposición lineal mediante:

$$W_K(k) = A_{1,K_1}(k) + A_{2,K_2}(k-K_1) + A_{3,K_3}(k-K_1-K_2) + \ldots + A_{p,K_p}\left(k - \sum_{i=1}^{p-1} K_i\right) \tag{2.45}$$

$$n = \max(n_1, n_2, \ldots, n_p) \tag{2.46}$$

La diferencia de orden $(n+1)$ está dada por:

$$\Delta^{n+1} W_K(k) = \Delta^{n+1} A_{1,K_1}(k) + \Delta^{n+1} A_{2,K_2}(k-K_1) + \ldots + \Delta^{n+1} A_{p,K_p}\left(k - \sum_{i=1}^{p-1} K_i\right) \tag{2.47}$$

Sustituyendo (2.37) en (2.47) se obtiene:

$$\begin{aligned}
\Delta^{n+1} W_K(k) = & \left[\sum_{j=1}^{n+1} g_{1,j}\delta(k-j) + \sum_{j=1}^{n+1} d_{1,j}\delta(k-K_1-j)\right] + \\
& + \left[\sum_{j=1}^{n+1} g_{2,j}\delta(k-K_1-j) + \sum_{j=1}^{n+1} d_{2,j}\delta(k-K_1-K_2-j)\right] + \ldots + \\
& + \left[\sum_{j=1}^{n+1} g_{p-1,j}\delta\left(k - \sum_{i=1}^{p-2} K_i - j\right) + \sum_{j=1}^{n+1} d_{p-1,j}\delta\left(k - \sum_{i=1}^{p-1} K_i - j\right)\right] + \\
& + \left[\sum_{j=1}^{n+1} g_{p,j}\delta\left(k - \sum_{i=1}^{p-1} K_i - j\right) + \sum_{j=1}^{n+1} d_{p,j}\delta\left(k - \sum_{i=1}^{p} K_i - j\right)\right]
\end{aligned} \tag{2.48}$$

Reagrupando términos similares se obtiene:

$$\begin{aligned}
\Delta^{n+1} W_K(k) = & \sum_{j=1}^{n+1} g_{1,j}\delta(k-j) + \sum_{j=1}^{n+1} (d_{1,j} + g_{2,j})\delta(k-K_1-j) + \\
& + \sum_{j=1}^{n+1} (d_{2,j} + g_{3,j})\delta(k-K_1-K_2-j) + \ldots + \\
& + \sum_{j=1}^{n+1} (d_{p-1,j} + g_{p,j})\delta\left(k - \sum_{i=1}^{p-1} K_i - j\right) + \sum_{j=1}^{n+1} d_{p,j}\delta\left(k - \sum_{i=1}^{p} K_i - j\right)
\end{aligned} \tag{2.49}$$

Definiendo (2.50) como la superposición lineal de términos delta, (2.49) se puede rescribir como (2.51).

$$s_{r,j} = d_{r-1,j} + g_{r,j} \tag{2.50}$$

$$W_K(k) = \sum_{j=1}^{n+1} s_{1,j}\delta(k-j) + \sum_{j=1}^{n+1} s_{2,j}\delta(k-K_1-j) + \sum_{j=1}^{n+1} s_{3,j}\delta(k-K_1-K_2-j) + \ldots + \\ + \sum_{j=1}^{n+1} s_{p,j}\delta\left(k - \sum_{i=1}^{p-1} K_i - j\right) + \sum_{j=1}^{n+1} s_{p+1,j}\delta\left(k - \sum_{i=1}^{p} K_i - j\right) \tag{2.51}$$

Siempre y cuando:

$$d_{0,j} = 0; \quad g_{p+1,j} = 0 \tag{2.52}$$

Con esto se demuestra que se necesitan $(p+1)(n+1)$ parámetros para describir una función polinomial de p polinomios de grado máximo n.

2.4.5 Diseño de perfiles polinomiales.

No todos los polinomios son útiles desde el punto de vista dinámico y es importante ubicar las propiedades que deben tener los polinomios para formar los perfiles de movimiento. Dicho a continuación se enuncian tres propiedades que serán de gran ayuda para el diseño del perfil de movimiento.

2.4.5.1 Polinomio prototipo.

Se denomina polinomio prototipo al polinomio con coeficientes exactos cuya morfología se desea conservar invariante.

2.4.5.2 Forma normalizada.

Es el polinomio prototipo al que se le han aplicado las transformaciones morfológicas de invarianza de tal forma que queda en función de un desplazamiento inicial igual a cero y un desplazamiento final igual a la unidad, en el intervalo de tiempo discreto deseado.

2.4.5.3 Propiedades dinámicas deseables.

Desplazamiento. La forma ideal del movimiento CNC es la curva de mínimo esfuerzo, dada por la tangente hiperbólica, por lo tanto, el perfil de movimiento debe aproximarse a esta función.

Velocidad. Para un desplazamiento del punto A al punto B en una trayectoria CNC la velocidad debe comenzar en cero, incrementarse hasta una velocidad máxima determinada y decrementarse hasta llegar nuevamente a cero. Siempre debe tener el mismo signo.

Aceleración. Para cumplir con la condición de perfil de velocidad, el perfil de aceleración deberá tener un área bajo la curva igual a cero.

Jaloneo. El punto ideal es un jaloneo igual a cero durante todo el movimiento, pero esta situación no es realista. Lo deseable es minimizar el máximo absoluto del jaloneo.

2.5 Cómputo Reconfigurable.

A la unión de la mecánica y la electrónica para resolver un problema se le llama Mecatrónica. En el diseño de un sistema de control para una máquina-herramienta es necesario contar con conocimientos en mecánica y en electrónica. El diseño de un buen sistema CNC se basa en la habilidad para conectar la mecánica con la electrónica (Rivera, 2007). El comportamiento de una máquina-herramienta está regido por características físicas no lineales que deben ser compensadas por el sistema CNC que es la parte electrónica.

Un perfil de movimiento que cumpla con la dinámica de movimiento ideal para una máquina-herramienta CNC requiere de un gran poder de cómputo, que incluso es imposible para una computadora generar el valor de la evaluación dentro del periodo de muestreo máximo permitido en la industria, además la precisión se ve comprometida dada la complejidad de las ecuaciones a evaluar, aunado a su elevado costo. Teniendo en cuenta lo antes mencionado es extremadamente complejo que una computadora realice el perfil de movimiento y a la vez ejecute infinidad de procesos nativos de la computadora.

Una opción para poder solucionar este problema es haciendo uso del computo reconfigurable que consiste en la utilización de hardware que puede adaptarse a un nivel lógico para resolver problemas específicos.

Para la utilización del cómputo reconfigurable se pueden utilizar varios dispositivos programables de alta densidad que se basan en tecnología SRAM, ejemplo de estos son los FPGA (Arreglo de compuertas programable en campo, *Field Programmable Gate Array*) que son circuitos de aplicación específica, lo que significa que mientras una computadora tiene que atender varios procesos a la vez un FPGA se dedicará exclusivamente a ejecutar la tarea que se le programe. La ventaja de un FPGA es que puede ejecutar varios procesos en paralelo, lo cual es imposible con un microprocesador o con un DSP. Debido a que es posible integrar varios bloques dentro del mismo FPGA que trabajen en paralelo y sin que se vea afectado el tiempo de ejecución de cada uno, así que es posible implementar el gene-

rador de perfiles y alguna otra estructura (si existe) en un mismo encapsulado para obtener un controlador de posición que mejore la dinámica de movimiento en maquina CNC. Un punto a favor del cómputo reconfigurable es que distingue el nivel de integración. Al tener la ventaja de implementar estructuras que realicen tareas independientes al mismo tiempo y que trabajen en conjunto se gana velocidad en los cálculos con respecto a una computadora convencional por lo tanto, es posible diseñar un controlador de movimiento basado en un FPGA para su aplicación en la industria, teniendo presente que todas las operaciones se deben realizar en un tiempo menor a 1ms.

2.5.1 El FPGA.

Los FPGA's son dispositivos lógicos de propósito general programable por los usuarios, compuestos de bloques lógicos comunicados por conexiones programables. Se basan en una cantidad muy grande de celdas lógicas, muy elementales y una mucho mayor interconectividad que sus contrapartes. Mientras más pequeñas resulten las celdas, mayor aprovechamiento se pueden tener de las mismas, aunque los retardos pueden aumentar (Troncoso, 2007).

La programación de un FPGA es mucho más compleja que la de un microprocesador o un DSP. Mientras en un microprocesador o DSP se tiene que programar una secuencia de instrucciones preestablecidas que serán ejecutadas una a la vez en un FPGA se describen bloques funcionales que son independientes unos con otros y se ejecutan al mismo tiempo. Para ejemplificar el alcance de un FPGA veamos el modelo XC3S200 que posee internamente, entre otras cosas, 200000 compuertas lógicas y 12 multiplicadores a un costo menor a los 20 dólares (Xilinx Corporation, 2005). La base fundamental para la construcción de un DSP es un multiplicador por lo que en este FPGA es posible integrar 12 DSP's que trabajen en paralelo y de forma coordinada.

Un FPGA es una herramienta versátil de bajo costo que permite implementar cualquier algoritmo e incluso es posible establecer el tiempo de ejecución, aunque hay un precio que pagar, entre más rápido sea la ejecución del algoritmo se requiere una mayor canti-

dad de recursos. El equilibrio entre velocidad y cantidad de recursos consumidos corre a cuenta del diseñador. Debido al alto nivel de abstracción requerido para diseñar estructuras digitales implementables en FPGA's y el hecho de que el lenguaje de programación es completamente diferente a los utilizados comúnmente, muy pocas personas utilizan estos dispositivos aun con la gran ventaja tecnológica que representan.

En resumen, un FPGA es una herramienta de bajo costo que permite la programación y ejecución de prácticamente cualquier algoritmo computacional, con la ventaja de que las operaciones se realizan en menor tiempo que una computadora. La utilización del FPGA para la solución del problema del generador de perfiles da la posibilidad de aplicarlo físicamente al control de una máquina CNC, y establece las bases para el diseño e implementación de un control de movimiento comercial para su aplicación en máquinas CNC.

La aplicación de los FPGA va más allá de la implementación lógica digital, ya que se puede utilizar para la implementación de arquitecturas específicas. Los sistemas que se han basado en FPGA's tienen grandes ventajas ya que proporcionan un mejor desmpeño que sus correspondientes en software. Los FPGA's son adecuados para procesos y/o aplicaciones que necesitan un gran número de operaciones para realizar una tarea (Osornio, 2007).

2.5.2 VHDL.

Los lenguajes de descripción de hardware (HDL) iniciaron alrededor de los años 70's. Aunque al principio no tuvo gran auge sino hasta inicios de los 80's que fue cuando apareció el lenguaje VHDL mediante el programa de circuitos de muy alta velocidad (VHSIC) del departamento de defensa de los Estados Unidos.

El objetivo de este lenguaje es poder describir en un lenguaje de cómputo circuitos y sistemas digitales. Una de las principales características de este lenguaje es que es ejecutable y posibilita la descripción de hardware con diferentes niveles de jerarquización.

Para la realización de la descripción de hardware se tiene que separar en varios bloques denominados componentes. Los cuales deben de tener una forma diferente de diseño para poder formar el sistema. A continuación se hace mención de estos componentes de diseño:

***Entity declaration*:** Describe la interfaz de la unidad de diseño la cual se comunica con otras entidades del mismo ambiente. La interfaz incluye todas las entradas, salidas y señales bidireccionales en la declaración del puerto.

***Architecture Body*:** Una arquitectura describe la composición funcional de un diseño.

***Configuration declaration*:** Es una unidad de diseño primaria usada para enlazar una entidad a una arquitectura con la finalidad de formar los componentes de un diseño.

Package declaration: Es un dispositivo que permite almacenar declaraciones que son comúnmente usadas para ser accedidas por múltiples unidades de diseño. ·

2.6 Propiedad Intelectual.

La propiedad intelectual es un derecho patrimonial de carácter exclusivo que otorga el Estado por un tiempo determinado para usar o explotar en forma industrial y comercial las invenciones o innovaciones, tales como un producto técnicamente nuevo, una mejora a una máquina o aparato, un diseño original para hacer más útil o atractivo un producto o un proceso de fabricación novedoso, también tiene que ver con la capacidad creativa de la mente: las invenciones, las obras literarias y artísticas, los símbolos, los nombres, las imágenes y privilegios.

La Ley de la Propiedad Industrial contempla diferentes figuras jurídicas de protección que se aplican de acuerdo a la naturaleza del producto intelectual. Se expide un título, que constituye un contrato social, mediante el cual se le confiere al titular el derecho temporal de explotar en forma exclusiva la invención y a cambio, el inventor divulga el contenido técnico de su invención para permitir el flujo de la información, lo que se constituye un valioso sistema para el avance científico y tecnológico.

Las instituciones que respaldan y hacen valer a la propiedad intelectual son: en México el IMPI (Instituto Mexicano de la Propiedad Intelectual), INDAUTOR (Instituto Nacional de Derechos de Autor) y mundialmente la OMPI (Organización Mundial de la Propiedad Intelectual).

CAPÍTULO 3. METODOLOGÍA.

En el presente capítulo se muestra el desarrollo de la estructura digital de un sistema para la generación de perfiles polinomiales. Se ha observado en el capítulo anterior que uno de los métodos para mejorar la dinámica de movimiento es por medio de los perfiles polinomiales de orden superior que realizan trayectorias mucho más suaves que los perfiles convencionales.

Uno de los problemas que se tiene para poder implementar estos perfiles en tiempo real se debe a la gran complejidad de cálculos que se requiere para la evaluación de los polinomios y las altas cargas computacionales.

3.1 Diseño en la generación de perfiles.

Basados en las características dinámicas deseables en el control de movimiento para maquinas CNC mencionados en capítulos anteriores, se realiza el desarrollo de perfiles polinomiales de movimiento de grado superior que cumplan con estas restricciones. Los perfiles a diseñar son de múltiples trazos, los perfiles de uno y dos trazos se diseñan a partir de la aceleración, y los perfiles de tres trazos se realizan a partir de la velocidad. La simbología a emplear es la siguiente: Posición X, Velocidad V, Aceleración A, Jaloneo J, Posición final X, Número de muestras de que cada trazo KI, grado del polinomio NC, número de trazos NT. Para calcular alguna de las variables elegidas para obtener una dinámica de movimiento adecuada se elige una variable a integrar o derivar por ejemplo:

$$X(k) = V(k) + X(k-1); V(k) = A(k) + V(k-1); J(k) = A(k) - J(k-1)$$

Todas las gráficas de los perfiles de ejemplo se realizan en base a una posición final cuentas por ejemplo X= 40000 cuentas K= 6000 muestras. Todos los perfiles de ejemplo son simétricos, lo que implica que cada trazo tiene la misma duración. Cabe aclarar que para el caso de los perfiles polinomiales de dos y tres trazos es posible obtener perfiles asimétricos.

3.1.1 Polinomios en tiempo discreto.

La técnica más útil para el diseño de perfiles es la polinomial a trazos. Pero hasta ahora no se ha podido implementar para su uso en una aplicación industrial debido a la gran cantidad de operaciones que requiere. Para resolver este problema se propone una técnica de evaluación polinomial que no requiere multiplicaciones, solamente sumas. Para empezar hay que establecer ciertas condiciones que nos permitirán la generación de la técnica, la primera consiste en remarcar que un polinomio tiene un dominio desde -∞ a +∞ pero para el problema de los perfiles de movimiento solamente nos interesa un intervalo de todo su dominio. La segunda se basa en la aplicación, debido a que el control que se está realizando

es digital el polinomio necesario para la generación de los perfiles debe ser discreto. Con estas dos consideraciones se establece la siguiente teoría:

Si $W_k(k)$ es un polinomio discreto en k de grado n para K muestras a ser evaluadas. La diferencia discreta está dada por la ecuación 3.1.

$$\Delta W_k(k) = W(k) - W(k-1) \tag{3.1}$$

El grado del polinomio disminuye en uno cada vez que se obtiene su diferencia discreta por lo que la n-ésima diferencia discreta de $W_k(k)$ es igual a una constante y la $(n+1)$-ésima diferencia discreta es igual a cero en el caso de un polinomio válido de -∞ a +∞ pero para el caso de un polinomio truncado, válido solamente en un determinado intervalo de todo su dominio, la $(n+1)$-ésima diferencia discreta es igual a un conjunto de parámetros llamados deltas de Kronecker con los cuales es posible reconstruir el polinomio truncado, la ecuación 3.2 describe este conjunto.

$$\Delta^{n+1} W_k(k) = \sum_{j-1}^{n+1} G_j \delta(k-j) + \sum_{j-1}^{n+1} D_j \delta(k-K-j) \tag{3.2}$$

La ecuación 3.11 representa la $(n+1)$-ésima diferencia discreta y está formada por dos conjuntos de deltas de Kronecker, el primero representa la parte de la generación (Ecuación 3.3) del polinomio y el segundo conjunto representa la parte de disolución o cancelación del polinomio (Ecuación 3.4). El número de deltas de Kronecker es n+1 tanto para la parte de la generación como para la parte de disolución del polinomio. A partir de esto se concluye que se necesitan $2(n+1)$ parámetros para reconstruir un polinomio discreto en k de grado n válido en un intervalo finito (Rivera, 2007).

$$G_j = -\sum_{i-j}^{n+1} (-1)^i \binom{n+1}{i} W(j-i) \tag{3.3}$$

$$D_j = -\sum_{i=j}^{n+1} (-1)^i \binom{n+1}{i} W(K+j-i) \qquad (3.4)$$

Donde $\binom{n+1}{i}$ es la distribución binomial y está dada por la ecuación 3.5

$$\binom{n+1}{i} = \frac{(n+1)!}{i!(n+1-i)!} \qquad (3.5)$$

De la ecuación 3.1 se puede obtener una relación de recursividad para obtener la integración discreta de un polinomio a partir de su diferencia el cual se muestra en la ecuación 3.6.

$$W_k(k) = \Delta W_k(k) + W_k(k-1) \qquad (3.6)$$

Teniendo en cuenta las ecuaciones anteriores se puede deducir que si se conocen las diferencias discretas de orden n de un polinomio en un punto dado y el valor mismo del polinomio en ese punto, se puede reconstruir computacionalmente el polinomio en forma completa mediante relaciones de recursividad (ecuaciones 3.1 y 3.6).

La estructura digital para la reconstrucción de un polinomio a partir de su $(n+1)$-ésima diferencia se obtiene a partir de la ecuación 3.6 y puede ser implementada como una serie de integradores discretos como se muestra en la figura 3.1.

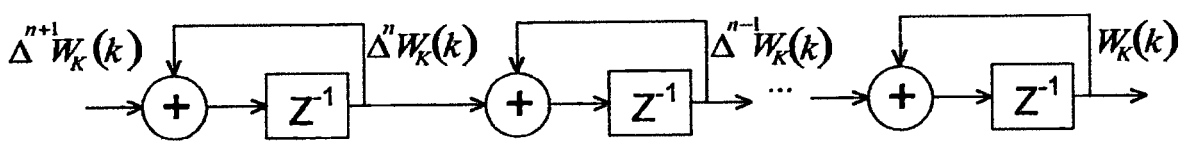


Figura 3.1 Estructura digital para la reconstrucción de un polinomio parametrizado

El principal problema en la implementación de un algoritmo son las multiplicaciones y divisiones porque su evaluación requiere de gran cantidad de recursos computacionales y de tiempo. Con la técnica de parametrización se elimina el uso de multiplicaciones y divisiones quedando solamente sumas. Los problemas de la evaluación de perfiles polinomiales en tiempo real queda resuelto con ésta técnica, por lo que es posible proponer perfiles polinomiales de movimiento de grado superior que mejoren la dinámica de movimiento en máquinas CNC. Debido a la simplicidad de cálculos que requiere una suma, es posible obtener un generador de perfiles de alta precisión sin comprometer la implementación en tiempo real.

Dadas la ventajas de la técnica de parametrización es posible realizar un generador de perfiles polinomiales de alta precisión para aplicaciones en control de movimiento de máquinas CNC. El algoritmo se implementa en un FPGA de bajo costo y se garantiza la ejecución del algoritmo a una frecuencia de trabajo mucho mayor que la mayoría de los sistemas existentes lo que garantiza su funcionamiento en tiempo real .

3.2 Estructura digital del generador de perfiles.

Hasta ahora se ha visto que la técnica de parametrización permite reconstruir polinomios sin necesidad de multiplicaciones, mediante integración discreta, lo que significa la ejecución de sumas de manera iterativa.

El principal objetivo del presente trabajo es la implementación en FPGA de un generador de perfiles a trazos de grado superior, para esto se propone diseñar un generador de perfiles con un número de trazos hasta de 16 bits y un grado de polinomio máximo de 3 bits, un dato importante a tener en cuenta es la cantidad total de muestras que tiene un tamaño de 32 bits para cada trazo.

Debido a que la técnica requiere la ejecución de sumas de manera recursiva, el error se acumula en cada integración discreta y se vuelve necesario manejar una alta resolución, si no se toma esto en consideración se obtendrán errores muy grandes cuando se requieran realizar movimientos muy largos. Dado que la técnica para la reconstrucción polinomial no utiliza multiplicaciones es posible proponer tamaño de palabra de 256 bits sin comprometer el objetivo de diseñar un generador de perfiles de bajo costo. El generador de perfiles debe trabajar en conjunto con un controlador PID. La exactitud del generador de perfiles es necesaria para garantizar una muy alta precisión en los movimientos de la máquina CNC que se esté controlando, aunque cabe aclarar que el controlador PID también influye en la precisión del controlador de movimiento.

Otro aspecto a tomar en consideración en el diseño del generador de perfiles es la velocidad de ejecución del algoritmo. El generador de perfiles debe ser capaz de generar una referencia en posición por lo menos cada milisegundo para garantizar la ejecución en tiempo real ya que éste tiempo de muestreo es un estándar para las aplicaciones industriales. Debido a la ausencia de multiplicaciones en el algoritmo esto es muy fácil de lograr, e incluso es posible realizar las sumas de manera iterativa de 32 bits a la vez, perdiendo un poco de velocidad en la ejecución del algoritmo pero ganando con esto una disminución en

la cantidad de recursos en el FPGA necesaria para el desarrollo del generador de perfiles de esta manera en vez de tener módulos para manejo de grandes cantidades se hace uso de módulos pequeños que funcionan de manera iterativa para conservar la precisión de las operaciones realizadas. Si se quisiera implementar el generador de perfiles de manera directa se necesitarían n+1 sumadores de 256 bits y n+1 registros de 256 bits

Un diagrama simplificado del generador de perfiles propuesto se muestra en la figura 3.2. La figura muestra un generador de perfiles para polinomios de grado n como se mencionó anteriormente el grado máximo de polinomio a generar por la estructura digital es n=*15*. A su vez se muestra en la figura 3.3 la FSM (máquina de estado finitos) del mismo, la cual tiene la tarea de controlar los módulos los diversos módulos con los que cuenta el generador, en el apéndice A se muestra las descripciones en VHDL de la estructura digital que conforma el core.

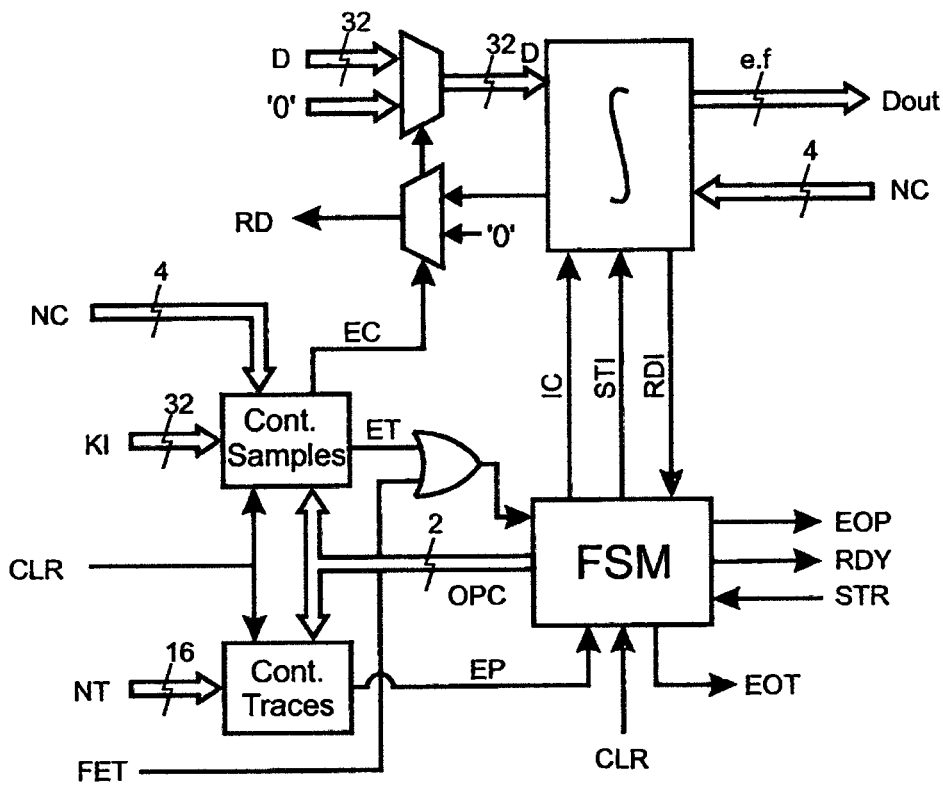


Figura 3.2 Estructura general del generador de perfiles polinomiales.

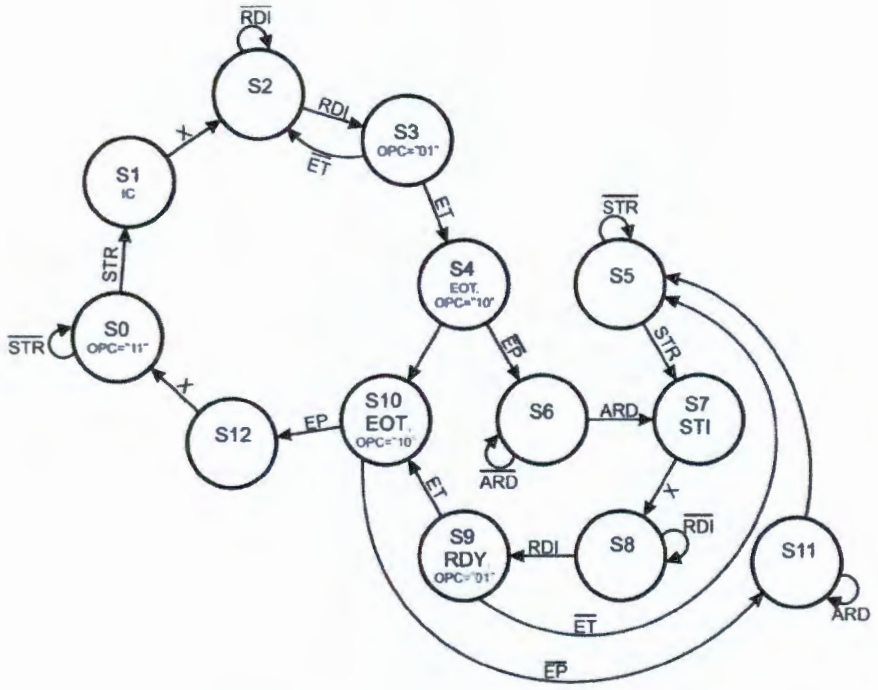


Fig. 3.3 Grafo de la máquina de estados finitos del generador de perfiles.

Nombre	Tipo	Ancho	Descripción
Delta	E	32	Dato de Entrada.
NC	E	4	Grado del Polinomio
NT	E	16	No. de Trazos
KI	E	32	No. de Muestra
FET	E	1	Finalizar Trazo
ARD	E	1	Permitir Lectura
STR	E	1	Inicio del Perfil
CLK	E	1	Reloj Maestro 50MHz
RST	E	1	Reset Asíncrono
CLR	E	1	Borrado Síncrono
RD	S	1	Lectura
EOT	S	1	Fin del Trazo
SOP	S	1	Inicio del Perfil
RDY	S	1	Ready
EOP	S	1	Fin del Perfil
Dout	S	36	Dato de Salida

Tabla 3.1 Terminales de Entrada y Salida del módulo generador de perfiles.

Para tener una mejor visión del generador de perfiles, se presenta en la figura 3.4 en la cual se puede observar detalladamente la manera en la cual se realizan las operaciones de manera iterativa mediante un sumador de 32 bits

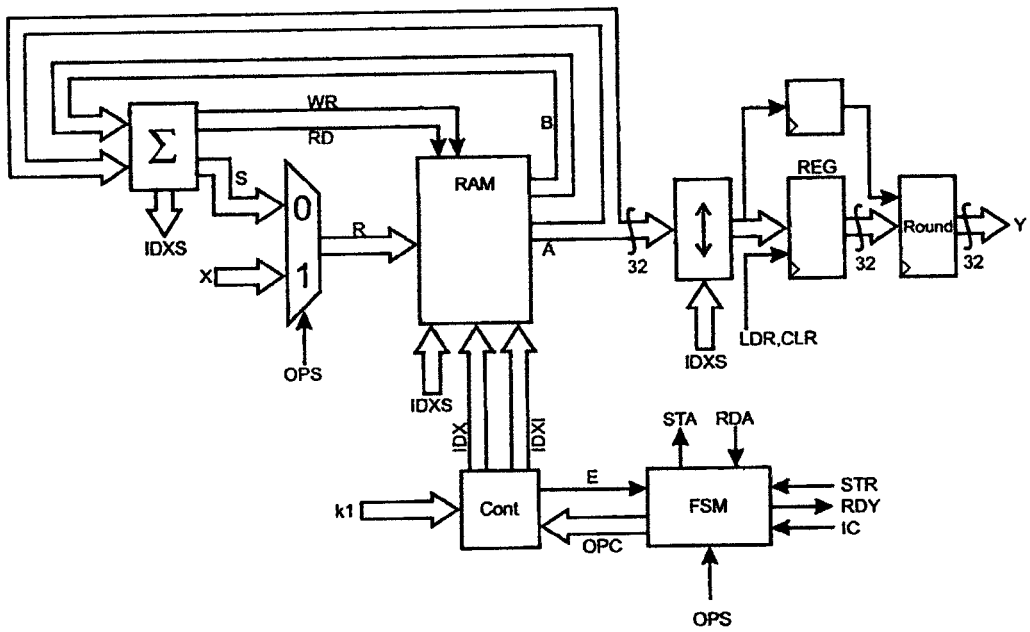


Figura 3.4 Estructura de la integral.

Las sumas son binarias y se realizan en formato complemento a dos, pero siguen el mismo principio de la suma en formato decimal, donde se suman dos dígitos y se guarda el acarreo para la siguiente suma, la unidad de acarreo se encarga de almacenarlo cada vez que se realiza una suma de 256 bits en partes de 32 bits, cuando se termina la suma es necesario limpiar la unidad de acarreo para realizar una nueva operación.

Dado que el sumador utilizado es de 32 bits es necesario utilizar registros del mismo tamaño, debido a esto cada *i*-ésima diferencia discreta de la memoria que está conformado por 8 registros de 32 bits.

En la primera operación el selector OPS de la figura 3.3 direcciona a la entrada de la señal X la cual solo tiene un vector de 32 bits de ceros el cual se almacena en las 8 locali-

dades de la memoria, seguido por la introducción de las deltas de Kronecker después de este ciclo ya se tienen los dos primeros vectores a sumarse (B+A véase en la figura 3.3). El resultado de la suma se almacena en la memoria y a continuación se vuelve a sumar con la siguiente delta de Kronecker.

Para tener una visión más general del módulo visto como una "caja negra" en la cual se puede observar claramente las señales de entrada en la parte lateral izquierda, en la parte lateral derecha se muestran las señales de salida y en las partes superior e inferior se localizan las señales de configuración.

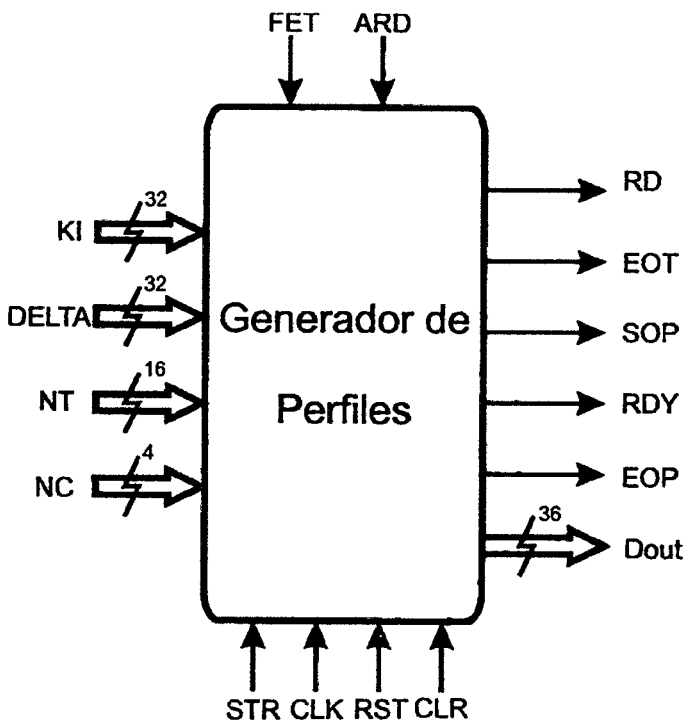


Figura 3.5 Bloque lógico del generador de perfiles.

Las señales de configuración están regidas por un reloj maestro "CLK" que tiene una frecuencia de 50 MHz ya que para la implementación en FPGA se usa por medio de la tarjeta SPARTAN 3 que trabaja con una señal de reloj a ésta frecuencia. En la figura 3.5 se muestra la señales de configuración y la manera de activación.

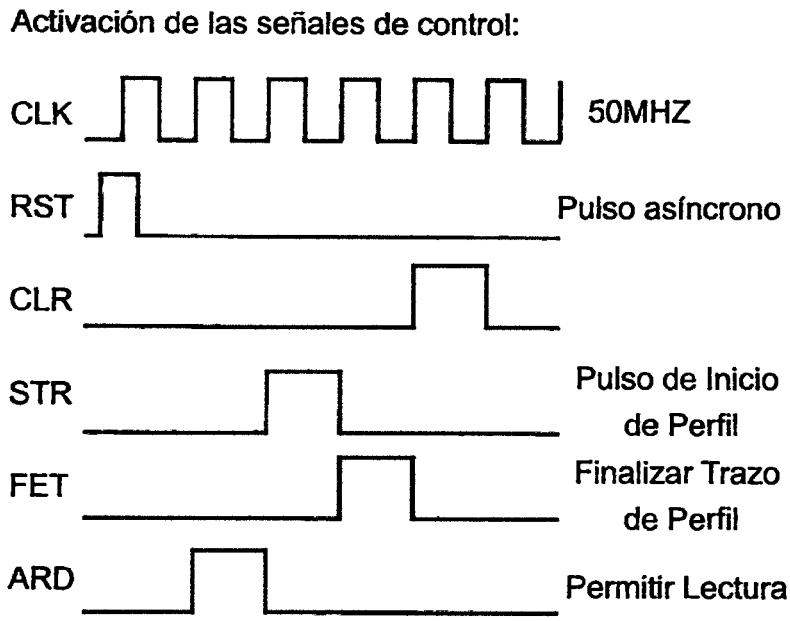


Figura 3.6 Activación de señales de control.

La señal RST se puede activar de manera asíncrona.

La señal CLR se activa de manera síncrona teniendo que estar CLK en alto al momento de la activación

STR es un pulso de activación del perfil y se debe de activar cuando RDY esté en alto

FET se puede activar asíncronamente para la finalización del trazo.

ARD se puede activar como un pulso o bien, después de poner la señal en alto se puede mantener así.

3.2.1 Simulación del generador de perfiles

A continuación se presentan algunas simulaciones del generador de perfiles, así como de la integral, en la cuales, para la simulación del generador de perfiles se tienen de entradas las Deltas de Kronecker, las señales de entrada en la integral son un escalón y una rampa respectivamente.

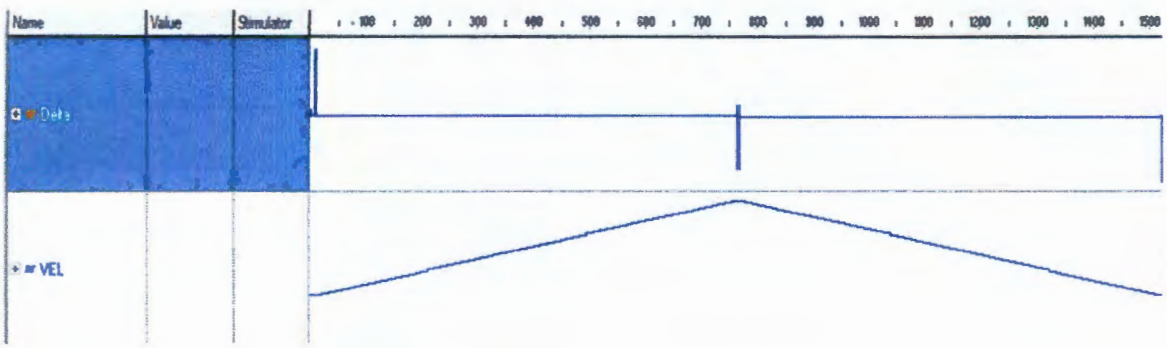


Figura 3.7 Simulación del generador de perfiles.

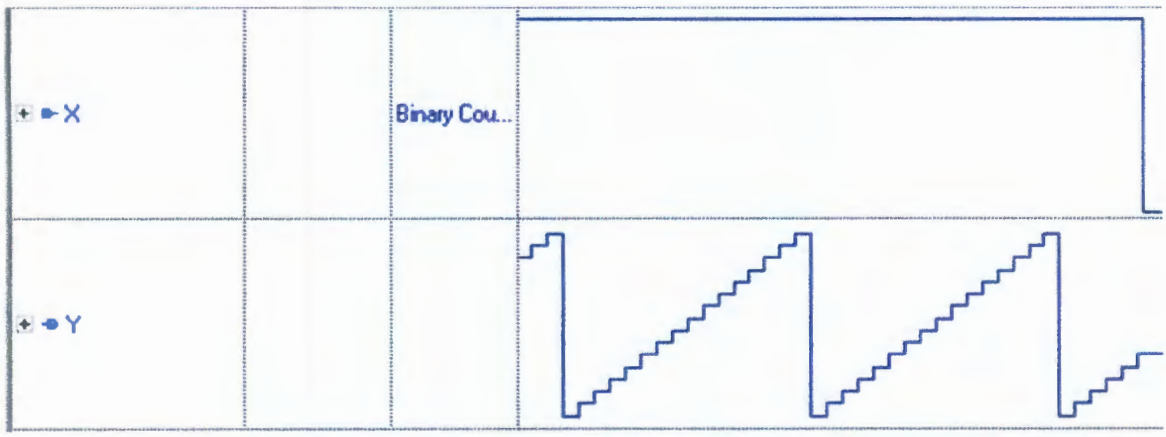


Figura 3.8 Simulación del integral con un escalón como entrada.

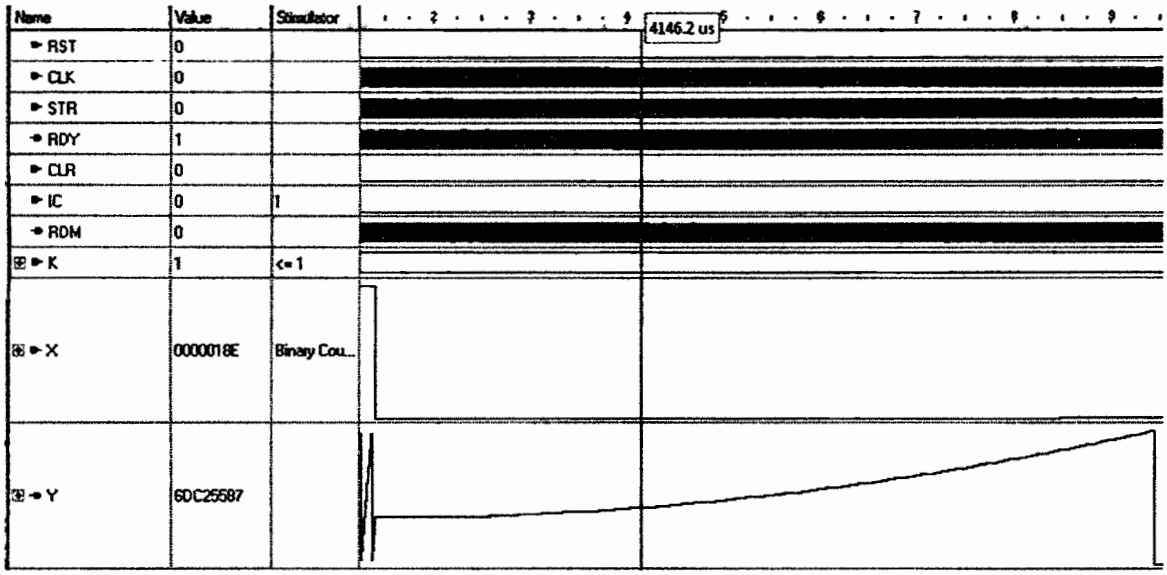


Figura 3.9 Simulación de la integral con una rampa como entrada.

3.3 Propuesta para la puesta en funcionamiento del generador de perfiles.

En todo proyecto es indispensable los resultados teóricos como las simulaciones mostradas anteriormente, sin embargo, al obtener resultados favorables en el área teórica es de gran importancia llevar el proyecto al siguiente nivel, es decir, realizar las pruebas necesarias de forma práctica para obtener resultados físicos y con esto dar una validación al trabajo realizado.

Desde el principio de este trabajo se hizo énfasis en que el core debe ser de propósito general para poderlo usar en cualquier aplicación que se requiera, teniendo la facilidad de ser compatible con otros cores basados en la misma tecnología (FPGA).

Una de las ramas ideales para probar el core es en maquinas-herramientas CNC movidas a través de servomotores en los cuales es deseable una buena dinámica de movimiento y aquí es donde el core para la generación de perfiles de movimiento tiene una excelente oportunidad para ser probado y validado.

Antes de realizar la prueba en una máquina-herramienta se decidió hacer una prueba preliminar usando solamente un servomotor en vacío con movimientos simples esto fue hecho por razones prácticas, debido a la facilidad de manipular un "servo", además de que para realizar la prueba en la máquina-herramienta se debe de tener en cuenta otros factores de riesgo a considerar, al término de esta prueba preliminar se procedió a hacer la prueba definitiva en un torno CNC reconvertido en el cual se probó con movimientos más complejos de los cuales se hará referencia en el capítulo 4. El torno CNC reconvertido fue provisto por la Universidad Autónoma de Querétaro, el cual se muestra en la figura 3.10.

Figura 3.10 Torno CNC reconvertido.

Al realizar dichas prueba se hizo uso de otra característica importante del core, la **compatibilidad,** debido a que para poner en marcha el servomotor o el torno es necesario

hacer uso de otros cores para el correcto funcionamiento de la prueba de esta forma el core para la generación de perfiles de movimiento pasa a formar una parte importante de un lazo de control dando una referencia a seguir por el controlador, en la figura 3.11 se muestra el diagrama a bloques de dicho lazo.

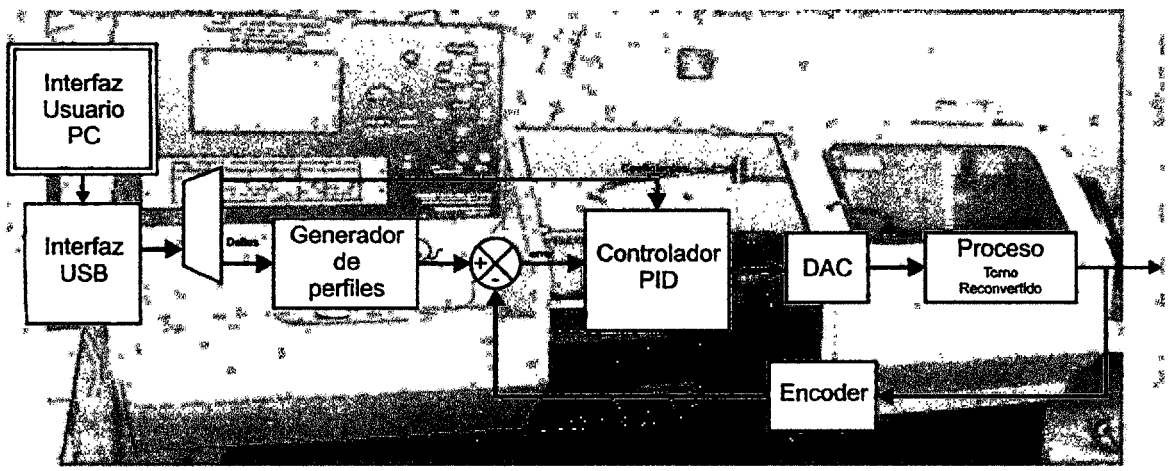


Figura 3.11 Diagrama a bloques del lazo de control.

Otro punto importante en el diagrama a bloques de la figura 3.11 es el uso de una interfaz USB que sirve de comunicación entre la PC y los cores de generación de perfiles y el controlador PID, respectivamente, proporcionando al generador de perfiles la Deltas de Kronecker para la reconstrucción del polinomio y los coeficientes para la correcta sintonización del controlador, la señal de control proveniente del PID es de naturaleza digital, por esto es necesario hacer uso de un DAC (Convertidor Digital-Analógico) para poder conectarlo con la planta a controlar (Torno reconvertido), para éste propósito se empleó sistema de adquisición de datos DAS1612 desarrollado por la Universidad Autónoma de Querétaro, dicho sistema es compatible con la tarjeta Spartan 3 de Digilent-Xilinx, sobre el cual están implementados todos las descripciones en VHDL. En la figua 3.12 se muestra el sistema DAS1612.

Figura 3.12 Sistema de adquisición de datos DAS1612.

Para la comunicación de la PC hacia el lazo de control se hizo uso de una interfaz gráfica para el usuario, este software fue desarrollado por la UAQ de nombre WinCN-CUAQ el cual tiene la capacidad de cargar el archivo de las Deltas de Kronecker y los coerfiecientes del controlador así como de monitorear las señales de referencia (perfil de movimiento) que provee el core en los dos ejes del torno (X y Z) y el seguimiento del controlador que realiza sobre el perfil de movimiento. En la figura 3.13 se muestra la interfaz gráfica del software usado.

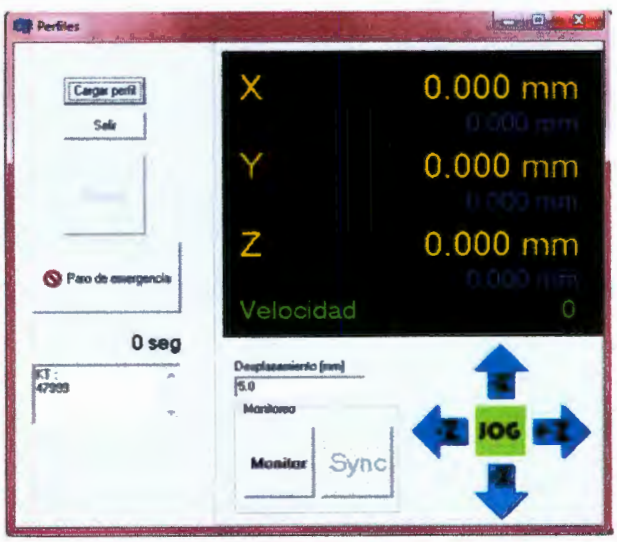


Figura 3.13 Interfaz gráfica del software WinCNCUAQ.

El objetivo principal de este experimento es probar que por medio de los perfiles de movimiento polinomiales, se tiene una significativa disminución en los cambios repentinos de velocidades y por consiguiente hay una mejora para los cambios en aceleración y jerk.

3.4 Descripción del contenido de información en el manual de usuario y la hoja de datos.

Debido a que el IP Core para la generación de perfiles de movimiento es concebido como un núcleo específico para la mejora de la dinámica de movimiento y está basado en tecnología FPGA de arquitectura abierta, teniendo las características de portabilidad y escalabilidad, es decir, se puede emplear para diversos propósitos y tiene la capacidad de interconectividad con otros cores basados en la misma tecnología, se ha realizado una hoja de datos y un manual de usuario para facilitar el uso de este core como un producto terminado. Dando al usuario una forma más accesible de operarlo, así pues, en esta sección se hace una breve descripción de la información que contienen dichos documentos los cuales están más ampliamente descritos en los apéndices B y C, respectivamente.

3.4.1 Hoja de datos.

La hoja de datos contiene de forma muy concisa en una figura en forma de "caja negra" las señales de entrada, salida de datos y señales de configuración, así como el ancho de bus de cada señal referenciada en la tabla, explicando el tipo de señal y el significado de la misma.

Las señales de entrada se encuentran en las partes laterales, y en las partes superior e inferior se localizan las señales de configuración mostradas en la figura, así mismo se cuenta con un diagrama de tiempos en el cual se puede observar la forma de activación de

las señales de configuración, notando que la mayoría son señales síncronas, la operación del core tiene una frecuencia máxima que está limitada a 103.477 MHz de esta manera se puede hacer la síntesis a una frecuencia mayor ya que este core fue sintetizado a una frecuencia de 50 MHz.

En la parte inferior derecha se hace referencia a la forma particular de activación de algunas señales de configuración.

Por último se tienen dos datos de gran importancia que son los formatos de entrada y salida de los buses de datos a procesar, es vital tener en cuenta esta información ya que al no tener configurado los demás cores con estos formatos puede arrojar un error de exactitud en procesamientos posteriores.

3.4.2 Manual de usuario.

El manual de usuario contiene información más detallada acerca de las estructuras digitales que conforman al Core para la generación de perfiles. De forma concisa se presentan las señales que conforman las entradas y salidas del core haciendo una breve descripción de la compatibilidad de éste core con otros.

En la siguiente página se despliega en forma de tabla un sumario de las señales del core, especificando las entradas y salidas y el ancho de bus, así como los formatos de entrada y salida de los datos, se hace énfasis en el porqué el bus de entrada de los datos es de 32 bits, teniendo en cuenta que el ancho de palabra de las Deltas de Kronecker es de 256 bits.

A continuación se hace un recuento de los aspectos más importantes en recursos que ocupa el core, teniendo en cuenta que para la realización de la síntesis se hizo uso de un FPGA modelo xc3s200 el cual cuenta con 200 mil compuertas disponibles. Nótese que éste modelo de FPGA es uno de los modelos más baratos en el mercado y por lo tanto con una cantidad de compuertas disponibles relativamente baja, dado por sentado que los recursos

que ocupa el core sobre el FPGA son mínimos, dando por hecho de que el core realizado es de bajo costo. En la parte inferior de ésta página se localiza un diagrama de tiempos en el cual se muestra la forma de activación de las señales de configuración. Por último se enfatiza la importancia de la señal NC, ya que está configurada para arrojar el perfil de movimiento en velocidad.

En la página 5 del manual de usuario se muestra de forma más detallada la constitución del core en forma de estructura digital, cabe señalar que los módulos vistos en dicha página son descritos en forma de código VHDL en el apéndice A.

En la página 6 se muestra principalmente el grafo de la máquina de estado principal, desplegando mediante una tabla los datos de las señales que maneja la FSM, también se muestra la señal OPC que controla la operación de los contadores.

En la página 7 se hace una descripción en forma de estructura digital del módulo de la integral, mostrando los datos de las señales de entrada y salida, por último se muestra una simulación general del core en la cual se pueden observar las señales y su operación.

Para finalizar el manual de usuario se muestra un pequeño glosario de los pequeños módulos digitales que constituyen a las estructuras presentadas.

CAPÍTULO 4. RESULTADOS

En este capítulo se describen los resultados obtenidos de la propuesta explicada en la sección 3.3, también se hará mención de los recursos que ocupó el core sintetizado en el FPGA.

4.1 Resultados de las pruebas realizadas.

Haciendo una breve referencia en el apartado 3.3 para la puesta en marcha de la prueba, se optó por realizar la puesta en marcha del core en un torno CNC reconvertido, debido a que la aplicación es ideal para la prueba que se necesita hacer.

Antes de hacer la prueba sobre la máquina-herramienta se realizó una prueba preliminar la cual consistió en el control de un servomotor en vacío, esta decisión fue tomada debido al riesgo que conlleva probar un core de ésta naturaleza en una máquina-herramienta ya que los servomotores podrían caer en un rango de inestabilidad y causar un daño en la integridad de la maquinaria, de esta manera se puede obtener un resultado previo y, si el resultado es favorable, se procede a hacer la prueba en el torno CNC reconvertido que se muestra en la figura 3.10.

En dichas prueba se observará la señal de salida del core y se analizará la trayectoria del perfil, esperando que la curva de velocidad no tenga cambios bruscos en su comportamiento, de manera parecida a las de la figura 2.10, que muestra un perfil de velocidad con una dinámica suave.

4.1.1 Prueba preliminar.

En esta prueba se procedió a conectar el servomotor en vacío simulando el proceso a controlar integrando todos los módulos que se observan en la figura 3.11, para este primer experimento se optó en alimentar el generador de perfiles con un polinomio de bajo grado, con el fin de observar el comportamiento de este de una manera física, posteriormente se le alimentaría con un polinomio de mayor complejidad. En la figura 4.1 se muestra el sistema físico armado para su posterior funcionamiento.

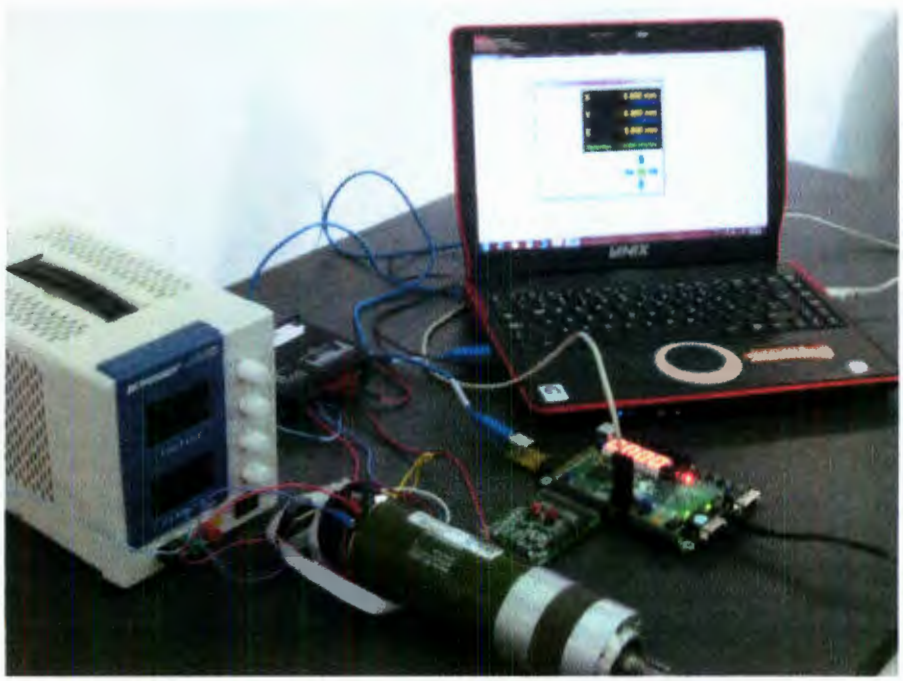


Figura 4.1 Interfaz Sistema armado para la prueba preliminar.

Al realizar dicha prueba se puedo comprobar que los resultados que arrojó el generador de perfiles son favorables, ya que en la gráfica de la figura 4.1 se puede observar que los cambios en la magnitud son suaves y en ningún momento tiene cambios bruscos.

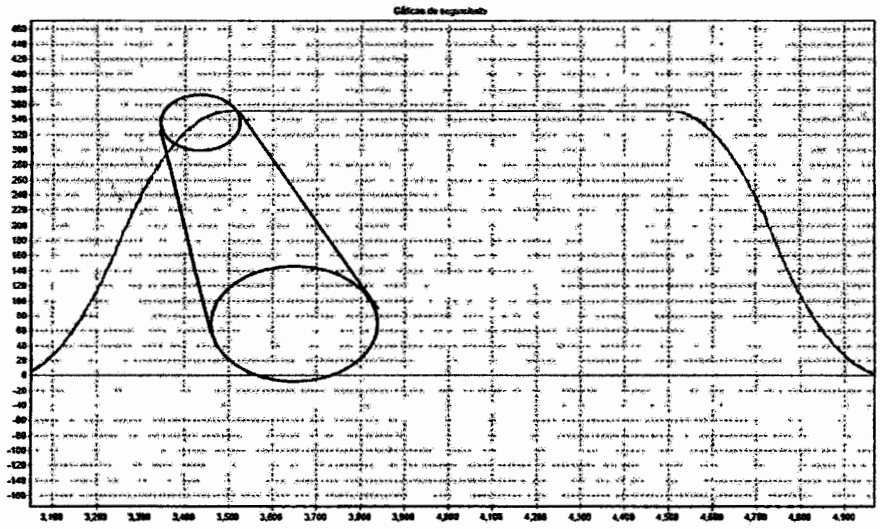


Figura 4.2 Seguimiento del perfil de velocidad.

En la figura 4.3 se puede ver con mucho más detalle un acercamiento de la parte indicada en la figura 4.2, esta parte es un punto crítico en la mayoría de los perfiles pues es aquí cuando se presenta un cambio brusco de magnitud, pero a diferencia de la gráfica mostrada se puede observar que el cambio en la magnitud es paulatino y de manera muy suave, la gráfica azul representa el perfil de movimiento, mientras que la gráfica roja indica el seguimiento del controlador.

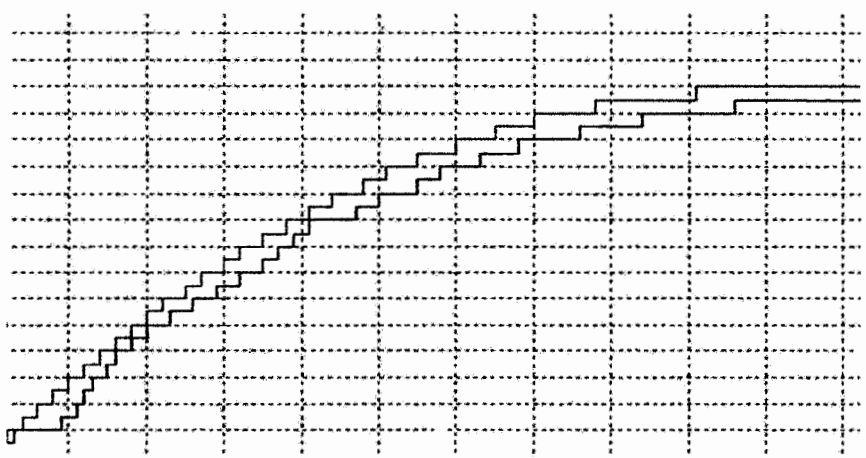


Figura 4.3 Acercamiento de la gráfica 4.2.

4.1.2 Prueba en Torno CNC reconvertido.

Después de haber realizado con éxito la prueba preliminar para el control del servomotor se procede a realizar la misma prueba pero, ahora teniendo como proceso un torno CNC reconvertido el cual se ilustra en la figura 3.10.

El torno, al tener la capacidad de mover dos ejes tiene la necesidad de controlar estos de forma coordinada para tener un maquinado exitoso, gracias a esta capacidad es posible crear figuras mucho más complejas.

Para hacer la prueba en el torno se hizo uso de la ventaja mencionada para maquinar un pequeño peón de ajedrez, al ser esta figura mucho más compleja las gráficas resultantes del maquinado tienen un valor más significativo, ya que se puede observar más a detalle y en diversos puntos de las trayectorias los resultados obtenidos. En la figura 4.4 se muestra la pieza correspondiente a las Deltas de Kronecker que fueron introducidas al sistema, para así reconstruir la trayectoria que diera origen a dicha pieza.

Figura 4.4 Acercamiento de la gráfica 4.2.

En la figura 4.5 se muestra los resultados del seguimiento de la trayectoria.

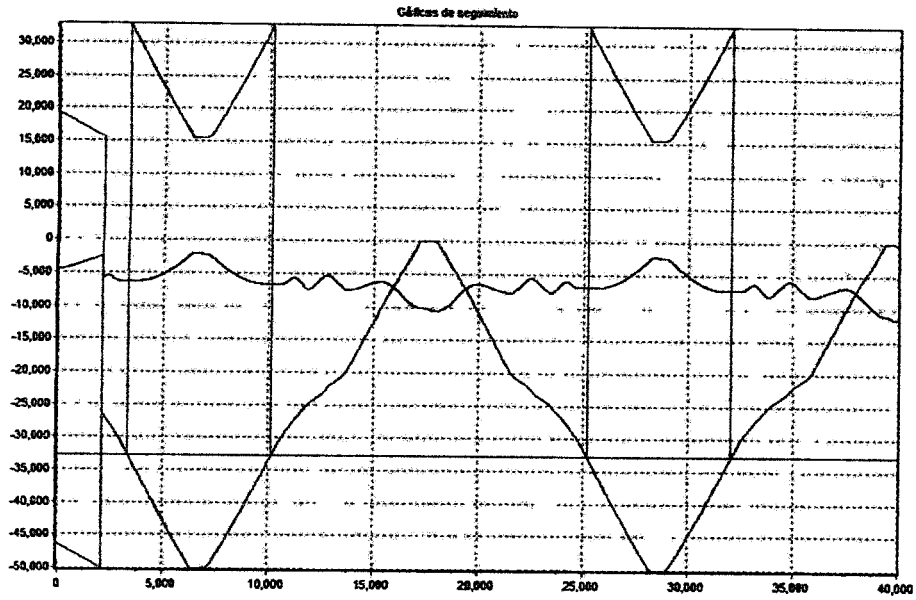


Figura 4.5 Seguimiento de la trayectoria para el peón de ajedrez.

En la figura 4.5 se puede observar la reconstrucción del polinomio a partir de las Deltas de Kronecker dadas, la línea de color rojo representa la trayectoria del eje X, mientras que la línea en color rosa representa la trayectoria seguida por el eje Z.

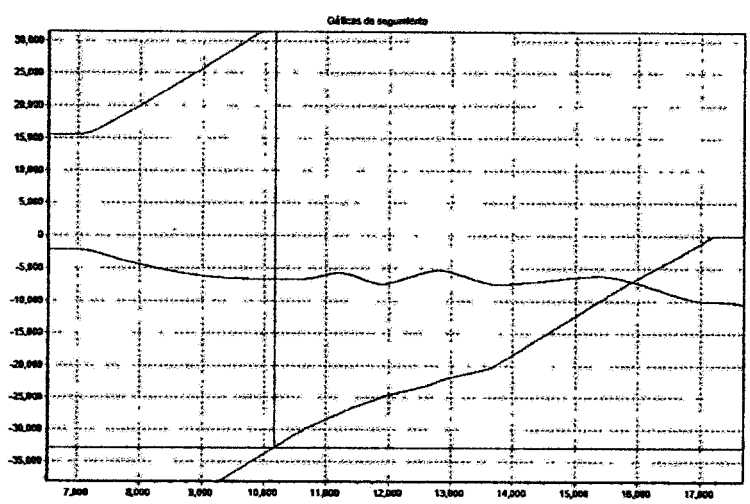


Figura 4.6 Seguimiento de la trayectoria para el peón de ajedrez.

La figura 4.6 y 4.7 muestra una pequeña sección de la trayectoria en donde se puede apreciar la línea azul que representa la referencia, es decir, el dato de salida del core, y en rojo se muestra el seguimiento del controlador a la referencia.

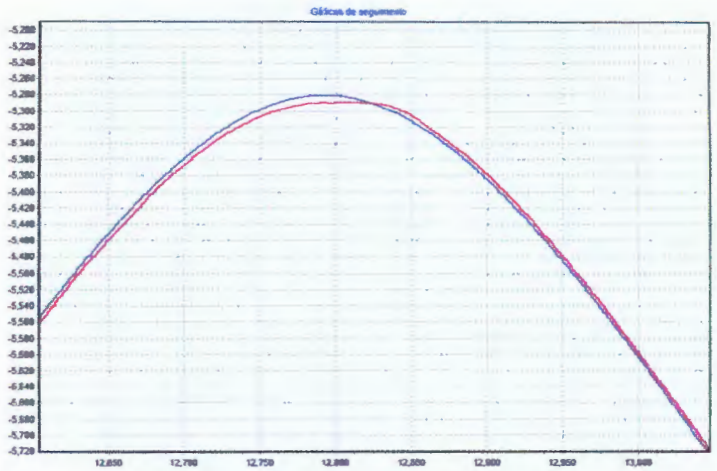


Figura 4.7 Seguimiento de la trayectoria para el peón de ajedrez.

Al inicio de este trabajo de tesis se mencionó la necesidad de que el proceso tuviera la capacidad de poder realizarse en tiempo real, es decir, el periodo de muestreo debe ser de 1 ms. Teniendo el core la capacidad de poder realizar este proceso sin ningún problema a ese rango de tiempo. En la figura 4.8 se muestra un cuadro de diálogo en el software el cual indica el tiempo de muestreo.

Figura 4.8 Cuadro de diálogo ilustrando la frecuencia de muestreo.

4.2 Síntesis del IP Core.

Una de las principales ventajas que se planteó en la sección 1.3 es que este IP Core para la generación de perfiles de movimiento debiera ser de bajo costo.

Basta con observar los resultados de la síntesis para percatarse de que este core es de bajo costo ya que la cantidad de recursos que ocupó en un FPGA de modelo xc3s200 es mínimo, pues este modelo de FPGA es uno de los más económicos en el mercado, con una cantidad de compuertas disponibles de 200 000 que es relativamente baja.

A continuación se muestra mediante una tabla la cantidad de recursos ocupados en la síntesis en porcentaje.

Slice Flip Flops	3%
Slices	10%
Bloques de RAMs	16%
GCLKs	12%

Tabla 4.1 Recursos del FPGA ocupados por el Core.

4.3 Conclusiones.

En la actualidad la industria en general tiene una necesidad básica: la constante mejora en la calidad de sus productos y menores tiempos de producción sin que se vean comprometidos los costos de producción, y de ser posible, sacar el máximo provecho a las máquinas-herramientas con las que se trabaja para alargar la vida útil de las mismas. La creciente globalización tecnológica ha hecho posible que estos aspectos sean satisfechos en gran medida, pero con una gran desventaja: elevados costos en productos para la actualización de las líneas de manufactura. Dejando en una gran desventaja a pequeñas industrias que no son capaces de adquirir dichas innovaciones tecnológicas.

Debido a lo antes mencionado se buscó una manera de contribuir para satisfacer las necesidades tecnológicas tomando en cuenta las características que deben tener dichas innovaciones. De esta manera se optó por mejorar la dinámica de movimiento de máquinas-herramientas.

Como producto de este trabajo de tesis se desarrolló un IPcore para la generación de perfiles de movimiento basado en tecnología FPGA, el cual maneja algoritmos polinomiales para su funcionamiento con la característica de ser genérico. El resultado de esta investigación fue satisfactorio, ya que los datos arrojados tanto en la simulación de los módulos de forma aislada como de forma integral manifestaron un correcto funcionamiento, de igual manera lo hizo al funcionar en una máquina-herramienta. De esta manera se puede inferir que cumplió con los estándares requeridos (tiempo de muestreo estándar para la industria, mejora en el calidad de la pieza maquinada, menor costo de producto, opción asequible para las pequeñas industrias, etc).

Como resultados derivados de este proyecto de investigación se generaron documentos para el correcto uso del IPcore (hoja de datos y manual de usuario) y al tener la característica de ser propiedad intelectual se iniciaron los trámites para el registro de la documentación de la hoja de datos y el manual de usuario.

Como observación general se puede destacar las ventajas que el FPGA posee sobre otros dispositivos como DSP's, microprocesadores y microcontroladores teniendo en cuenta que además de tener la capacidad de reconfigurar la arquitectura del core es posible incluir otros cores de la misma tecnología en el mismo chip, aunado a el bajo costo que un chip representa (aproximadamente $20 US)

De manera personal se puede afirmar que la presente tesis ha servido para conocer una faceta diferente del conocimiento científico el cual sin duda contribuirá en un futuro al mejor desempeño individual que se podrá ver reflejado en la vida profesional.

BIBLIOGRAFÍA.

[1]. http://users.bergen.org/jdefalco/CNC/index.html

[2]. Boon, G.K.; Mercado, A.; Automatización Flexible en la Industria; Ed. LIMUSA-Noriega, México, 1991.

[3]. DRAE

[4]. Varios autores. 1984. Enciclopedia de Ciencia y Técnica. Salvat Editores S.A.

[5]. Osornio. 2004. Diseño y Construcción de una Tarjeta Controladora de 3 Ejes. Tesis de Maestría. Universidad Autónoma de Querétaro.

[6]. Osornio. 2007. Diseño de Sistema de Control para CNC de Alta Velocidad. Tesis de Doctorado. Universidad Autónoma de Querétaro.

[7]. http://www.sitaltech.com/pdf/mil-std-1553_ip_cores.pdf

[8]. Altintas. 2000. Manufacturing Automation, Metal Cutting Mechanics, Machine-Tool Vibrations and CNC Design. Cambridge University Press.

[9]. Serway, Raymond A.; Jewett, John W. (2004). *Physics for Scientists and Engineers*, 6ª edición, Brooks/Cole.

[10]. Romero Troncoso. 2007. Electrónica Digital y Lógica Programable. Universidad de Guanajuato.

[11]. Romero T. 2004. Procesamiento de Señales para la Detección de Ruptura de Herramienta en Sistemas de Manufactura por Control Numérico Computarizado. Tesis de Doctorado. Universidad Autónoma de Querétaro.

[12]. Wook, J. y Yun K. 2002. FPGA Based Acceleration and Desceleration Circuit for Indutrial Robots and CNC Machine Tools. J. Mechatronics.

[13]. Erkorkmaz, K. y Altintas, Y. 2001. High Speed CNC System Design Part I: jerk Limited Trajectory Generation and Quintic Spline Interpolation, J. Machine Tools &Manufacture

[14]. Delta Tau, Data Systems Corporation. (2005). Turbo PMAC-Lite PCI Data Sheet. Delta Tau Data Systems.

[15]. Galil Motion Corporation. (2004). DMC-18x2 Series Data Sheet. Galil Motion Corporation.

[16]. Yih, F. 2005. Design and Implementation of a Linear jerk Filter for a Computerized Numerical Controller, J. Control Engineering Practice, 13: 567-576.

[17]. N. Tounsi, T. Bailey, M.A. Elbestawi (2003). Identification of acceleration deceleration profile of feed drive systems in CNC machines. International Journal of Machine Tools & Manufacture, 43, 441–451.

[18]. A. Gasparetto, V. Zanotto (2006). A New Method for Smooth Trajectory Planning of Robot Manipulators. Mechanism and Machine Theory.

[19]. Zhiwei, Y., Fengfeng, X. y Bin, W. 2005. A Shape Adaptive Motion Control System With Application to Robotic Polishing, Robotics and Computer-Integrated Manufacturing, 21: 355–367.

[20]. Jae Wook Jeon, Young Youl Ha (2000). A Generalized Approach for the Acceleration and Deceleration of Industrial Robots and CNC Machine Tools. IEEE Transactions On Industrial Electronics, 47, 1.

[21]. Y. Mizoshita, S. Hasegawa, and K. Takaishi (1996). Vibration Minimized Access Control for Disk Drives. IEEE transactions on Magnetics, 32, 3, 1793-1798.

[22]. Byung-Hoon Chang, Yoichi Hori (2006). Trajectory design considering derivative of jerk for head-positioning of disk drive system with mechanical vibration. IEEE/ASME transactions on Mechatronics, 11, 3, 273-279.

[23]. Flash, T., Hogan, N. (1985). The Coordiantion of Arm Movements: An Experimetally Confirmed Mathematical Model. J. of Neuroscience, 5:1688-1703

[24]. Xilinx Corporation. (2005). Spartan-3 Family FPGAs Data Sheet. Xilinx Corporation.

[25]. Colín, J. 2006. Descripción VHDL de los Bloques Funcionales de un Controlador Digital PID. Tesis de Maestría. Universidad Autónoma de Querétaro.

[26]. Alaniz, D. 2003. Tesis de Grado Maestría. Instrumentación y Control de una Máquina-Herramienta de Dos Ejes. Universidad Autónoma de Querétaro, Facultad de Ingeniería.

[27]. Rivera. 2007. Tesis de Grado Maestría. Perfiles Polinomiales de Movimiento para Máquinas CNC. Universidad de Guanajuato.

[28]. Hogan, N. (1984). An organizing Principle for a Class of Voluntary Movements. J. of Neuroscience

[29]. Pritschow, G. 1992. Open System Controllers a Challenge for the Future of the Machine Tool Industry, Annals of CIRP, Vol. 42: 449-452.

[30]. Wrigth, P.K. 1990. Open Architecture Manufacturing: The Impact of Open Computer Systems on Self Sustaining Machinery and the Machine Tools Industry. Proc. Manufacturing Industry 90' Part 2: 41-47.

APÉNDICE A. MÓDULOS VHDL

En este apéndice se presentan las descripciones en VHDL concernientes a la estructura digital del generador de perfiles presentada en el capítulo 3 en la figura 3.2.

A.1 Generador de perfiles.

```
2     -- Generador de perfiles polinomiales
3
4     library IEEE;
5     use IEEE.STD_LOGIC_1164.all;
6
7
8     entity mrpProfile is
9          port(
10             RST : in STD_LOGIC;
11             CLK : in STD_LOGIC;
12             STR : in STD_LOGIC;
13             RDY : out STD_LOGIC;
14             CLR : in STD_LOGIC;
15             ARD : in STD_LOGIC;
16             RD : out STD_LOGIC;
17             EOT : out STD_LOGIC;
18             EOP : out STD_LOGIC;
19             SOP : out STD_LOGIC;
20             FET : in STD_LOGIC;
21            KI : in STD_LOGIC_VECTOR(31 downto 0);
22             NT : in STD_LOGIC_VECTOR(15 downto 0);
23             NC : in STD_LOGIC_VECTOR(5 downto 0);
24             Delta : in STD_LOGIC_VECTOR(31 downto 0);
25             VEL : out STD_LOGIC_VECTOR(35 downto 0)
26             );
27    end mrpProfile;
28
29    architecture mrpProfile of mrpProfile is
30
31        component mrpCountTrace
32         generic(n : integer := 8);
33         port(
34             RST : in STD_LOGIC;
35             CLK : in STD_LOGIC;
36             OPC : in STD_LOGIC_VECTOR(1 downto 0);
37             E : in STD_LOGIC_VECTOR(n-1 downto 0);
38             E : out STD_LOGIC);
39        end component;
40
41        component mrpCountSmp
42         generic(m : integer := 3;
43                 n : integer := 8);
44         port(
45             RST : in STD_LOGIC;
46             CLK : in STD_LOGIC;
47             OPC : in STD_LOGIC_VECTOR(1 downto 0);
48             NC : in STD_LOGIC_VECTOR(m-1 downto 0);
```

```
            KI : in STD_LOGIC_VECTOR(n-1 downto 0);
            E : out STD_LOGIC;
            EC : out STD_LOGIC);
        end component;

        component mriIntegral
         generic(e : integer := 8;
                 f : integer := 8);
         port(
            RST : in STD_LOGIC;
            CLK : in STD_LOGIC;
            STB : in STD_LOGIC;
            RDY : out STD_LOGIC;
            CLR : in STD_LOGIC;
            IC : in STD_LOGIC;
            RDM : out STD_LOGIC;
            K   : in  STD_LOGIC_VECTOR(3 downto 0);
            X   : in  STD_LOGIC_VECTOR(31 downto 0);
            Y   : out STD_LOGIC_VECTOR(e+f-1 downto 0));
        end component;

       component mrpFSM
         port(
            RST : in STD_LOGIC;
            CLK : in STD_LOGIC;
            STB : in STD_LOGIC;
            RDY : out STD_LOGIC;
            CLR : in STD_LOGIC;
            EF : in STD_LOGIC;
            ET : in STD_LOGIC;
            ARD : in STD_LOGIC;
            EOP : out STD_LOGIC;
            STI : out STD_LOGIC;
            RDI : in STD_LOGIC;
            IC : out STD_LOGIC;
            EOI : out STD_LOGIC;
            OPC : out STD_LOGIC_VECTOR(1 downto 0));
       end component;

       component mrpGReg
         generic(n : integer := 8);
         port(
            RST : in STD_LOGIC;
            CLK : in STD_LOGIC;
            CLR : in STD_LOGIC;
            LDR : in STD_LOGIC;
            Din : in STD_LOGIC_VECTOR(n-1 downto 0);
            Dout : out STD_LOGIC_VECTOR(n-1 downto 0));
       end component;

        signal RDI,STI,IC,ET,LDR,EP,EC,RDM,ETE : std_logic;
        signal OPC       : std_logic_vector(1 downto 0);
        signal DEG : std_logic_vector(3 downto 0);
        signal LD : std_logic_vector(4 downto 0);
        signal Di : std_logic_vector(31 downto 0);
        signal Do : std_logic_vector(35 downto 0);

    begin

       DEG <= NC;
       RDY <= LDR;
       SOP <= IC;
       ETE <= ET OR FET;

       Mux: process(EC,Delta,RDM)
       begin
          if EC = '0' then
             RD <= RDM;
             Di <= Delta;
          else
             RD <= '0';
             Di <= (others => '0');
          end if;
       end process Mux;

        Traces:       mrpCountTrace generic map (16)  port map (RST,CLK,OPC,NI,EP);

        Samples:       mrpCountSmp generic map (4,32) port map (RST,CLK,OPC,NC,KI,ET,EC);

        Integral:       mriIntegral generic map(18,18) port map (RST,CLK,STI,RDI,CLR,IC,RDM,DEG,Di,Do);

       OutputReg:     mrpGReg generic map(36) port map (RST,CLK,CLR,LDR,Do,VEL);

        Control:            mrpFSM port map (RST,CLK,STB,LDR,CLR,EF,ETE,ARD,EOP,STI,RDI,IC,EOI,OPC);

    end mrpProfile;
```

A.2 Contador de trazos.

```
3   library IEEE;
4   use IEEE.STD_LOGIC_1164.all;
5   use ieee.numeric_std.all;
6
7   entity mrpCountTrace is
8       generic(n : integer := 3);
9       port(
10          RST : in STD_LOGIC;
11          CLK : in STD_LOGIC;
12          OPC : in STD_LOGIC_VECTOR(1 downto 0);
13          K : in STD_LOGIC_VECTOR(n-1 downto 0);
14          E : out STD_LOGIC
15          );
16  end mrpCountTrace;
17
18  architecture mrpCountTrace of mrpCountTrace is
19  signal Qn,Qp : std_logic_vector(n-1 downto 0);
20  begin
21
22      Count: process(Qp,K,OPC)
23      begin
24          case OPC is
25              when "00"   => Qn <= Qp;
26              when "01"   => Qn <= Qp;
27              when "10"   => Qn <= std_logic_vector(unsigned(Qp) + 1);
28              when others => Qn <= (others => '0');
29          end case;
30
31          if Qp = K then
32              E <= '1';
33          else
34              E <= '0';
35          end if;
36      end process Count;
37
38      Reg: process(RST,CLK,Qn)
39      begin
40          if RST = '1' then
41              Qp <= (others => '0');
42          elsif CLK'event AND CLK = '1' then
43              Qp <= Qn;
44          end if;
45      end process Reg;
46
47  end mrpCountTrace;
```

A.3 Contador de muestras.

```
3   library IEEE;
4   use IEEE.STD_LOGIC_1164.all;
5   use ieee.numeric_std.all;
6
7   entity mrpCountSmp is
8       generic(m : integer := 3;
9                n : integer := 5);
10      port(
11          RST : in STD_LOGIC;
12          CLK : in STD_LOGIC;
13          OPC : in STD_LOGIC_VECTOR(1 downto 0);
14          NC : in STD_LOGIC_VECTOR(m-1 downto 0);
15          KI : in STD_LOGIC_VECTOR(n-1 downto 0);
16          E : out STD_LOGIC;
17          EC : out STD_LOGIC
18          );
19  end mrpCountSmp;
20
21  architecture mrpCountSmp of mrpCountSmp is
22  signal EQU : std_logic;
23  signal Qn,Qp : std_logic_vector(n-1 downto 0);
24  signal Gn,Gp : std_logic_vector(m-1 downto 0);
25  begin
```

```
26
27      EC <= EQU;
28
29      Comp: process(NC,Gp)
30      begin
31         if NC = Gp then
32            EQU <= '1';
33         else
34            EQU <= '0';
35         end if;
36      end process Comp;
37
38       Mux: process(Qp,Gp,EQU,KI,OPC)
39       begin
40         case OPC is
41               when "00"   => Gn <= Gp;
42                              Qn <= Qp;
43               when "01"   => if EQU = '1' then
44                                 Gn <= Gp;
45                              else
46                                 Gn <= std_logic_vector(unsigned(Gp) + 1); --Gp + 1;
47                              end if;
48                              Qn <=  std_logic_vector(unsigned(Qp) + 1);--Qp + 1;
49               when others => Gn <= (others => '0');
50                              Qn <= (others => '0');
51           end case;
52
53           if Qp = KI then
54               E <= '1';
55           else
56               E <= '0';
57           end if;
58       end process Mux;
59
60       Reg: process(RST,CLK,Qn,Gn)
61       begin
62           if RST = '1' then
63               Qp <= (others => '0');
64               Gp <= (others => '0');
65           elsif CLK'event AND CLK = '1' then
66               Qp <= Qn;
67            Gp <= Gn;
68           end if;
69       end process Reg;
70
71   end mrpCountSmp;
```

A.4 Integral.

```
3       library IEEE;
4       use IEEE.STD_LOGIC_1164.all;
5       use ieee.numeric_std.all;
6
7       entity mriIntegral is
8            generic(e : integer := 20;
9                    f : integer := 12);
10           port(
11               RST : in STD_LOGIC;
12               CLK : in STD_LOGIC;
13               STR : in STD_LOGIC;
14               RDY : out STD_LOGIC;
15               CLR : in STD_LOGIC;
16               IC : in STD_LOGIC;
17               RDM : out STD_LOGIC;
18               R   : in  STD_LOGIC_VECTOR(3 downto 0);
19               X   : in  STD_LOGIC_VECTOR(31 downto 0);
20               Y   : out STD_LOGIC_VECTOR(e+f-1 downto 0)
21              );
22      end mriIntegral;
23
24      architecture mriIntegral of mriIntegral is
25
26          component mraAdd
27           generic(
28               n : Integer := 8;
29               m : Integer := 3
30              );
31           port(
```

```
32            RST : in STD_LOGIC;
33            CLK : in STD_LOGIC;
34            STB : in STD_LOGIC;
35            RDY : out STD_LOGIC;
36            A   : in  STD_LOGIC_VECTOR(n-1 downto 0);
37            B   : in  STD_LOGIC_VECTOR(n-1 downto 0);
38            S   : out STD_LOGIC_VECTOR(n-1 downto 0);
39            IDX : out STD_LOGIC_VECTOR(m-1 downto 0);
40            WR : out STD_LOGIC;
41            RC : out STD_LOGIC);
42        end component;
43
44        component mriRAM_2P
45         generic(
46             m : Integer := 8;
47             n : Integer := 7;
48             w : Integer := 128
49               );
50         port(
51             RST : in STD_LOGIC;
52             CLK : in STD_LOGIC;
53             WR : in STD_LOGIC;
54             DI : in STD_LOGIC_VECTOR(m-1 downto 0);
55             ADW : in STD_LOGIC_VECTOR(n-1 downto 0);
56             RDa : in STD_LOGIC;
57             DOa : out STD_LOGIC_VECTOR(m-1 downto 0);
58             ARa : in STD_LOGIC_VECTOR(n-1 downto 0);
59             RDb : in STD_LOGIC;
60             DOb : out STD_LOGIC_VECTOR(m-1 downto 0);
61             ARb : in STD_LOGIC_VECTOR(n-1 downto 0));
62        end component;
63
64        component mriCount
65         generic(n : integer := 3);
66         port(
67             RST : in STD_LOGIC;
68             CLK : in STD_LOGIC;
69             OPC : in STD_LOGIC_VECTOR(1 downto 0);
70             R : in STD_LOGIC_VECTOR(n-1 downto 0);
71             E : out STD_LOGIC;
72             Q : out STD_LOGIC_VECTOR(n-1 downto 0);
73             QI : out STD_LOGIC_VECTOR(n-1 downto 0));
74        end component;
75
76        component mriFSM
77         port(
78             RST : in STD_LOGIC;
79             CLK : in STD_LOGIC;
80             STR : in STD_LOGIC;
81             RDY : out STD_LOGIC;
82             E : in STD_LOGIC;
83             IC : in STD_LOGIC;
84             STA : out STD_LOGIC;
85             RDA : in STD_LOGIC;
86             OPS : out STD_LOGIC;
87             OPC : out STD_LOGIC_VECTOR(1 downto 0));
88        end component;
89
90        component mriGReg
91         generic(n : integer := 8);
92         port(
93             RST : in STD_LOGIC;
94             CLK : in STD_LOGIC;
95             CLR : in STD_LOGIC;
96             LDR : in STD_LOGIC;
97             Din : in STD_LOGIC_VECTOR(n-1 downto 0);
98             Dout : out STD_LOGIC_VECTOR(n-1 downto 0));
99        end component;
100
101       signal STA,RDA,WR,RC     : std_logic;
102       signal OPS,EI            : std_logic;
103       signal OPC,LDR           : std_logic_vector(1 downto 0);
104       signal IDXS              : std_logic_vector(2 downto 0);
105       signal IDX,IDXI,DEG      : std_logic_vector(3 downto 0);
106       signal ADW,ARa,ARb       : std_logic_vector(6 downto 0);
107       signal A,B,S,R           : std_logic_vector(31 downto 0);
108       signal YP                : std_logic_vector(e+f-1 downto 0);
109       signal YE                : std_logic_vector(e+f downto 0);
110   --  signal Ys                : std_logic_vector(1 downto 0);
111
112   begin
113
114       ARa <= IDX & IDXS;
115       ARb <= IDXI & IDXS;
116       DEG <= R;
117
```

```
118     Mux: process(OPS,S,X,IDXS,IDX,IDXI,RD)
119     begin
120         if OPS = '1' then
121             R <= X;
122             ADW <= IDX & IDXS;
123          RDW <= RD;
124         else
125             R <= S;
126             ADW <= IDXI & IDXS;
127          RDW <= '0';
128         end if;
129     end process Mux;
130
131     Suma:    mraAdd generic map (32,3) port map (RST,CLK,STA,RDA,A,B,S,IDXS,WR,RD);
132
133     RAM:     mriRAM_2P generic map (32,7,128) port map (RST,CLK,WR,R,ADW,RD,A,ARa,RD,B,ARb);
134
135     Conta:   mriCount generic map (4) port map (RST,CLK,OPC,DEG,EI,IDX,IDXI);
136
137     Control: mriFSM port map (RST,CLK,STR,RDY,EI,IC,STA,RDA,OPS,OPC);
138
139     Load: process(IDXS,EI,WR,OPS)
140     begin
141         if EI = '1' then
142             case IDXS is
143                 when "101"  => LDR(1) <= WR AND NOT OPS;     -- Primeros 32 bits fraccionarios 64.192
144                                LDR(0) <= '0';
145                 when "110"  => LDR(0) <= WR AND NOT OPS;     -- Primeros 32 bits enteros 64.192
146                                LDR(1) <= '0';
147                 when others => LDR <= "00";
148             end case;
149         else
150             LDR <= "00";
151         end if;
152     end process Load;
153
154 --  Ys <= '0'&YE;
155     Y <= std_logic_vector(unsigned(YE(e+f downto 1)) + unsigned('0'&YE(0)));        -- Redondeo
156
157     Reg0:    mriGReg generic map (e)
158                    port map (RST,CLK,CLR,LDR(0),A(e-1 downto 0),YE(e+f downto f+1));  -- Parte entera
159     Reg1:    mriGReg generic map (f+1)
160                    port map (RST,CLK,CLR,LDR(1),A(31 downto 31-f),YE(f downto 0));    -- Parte fraccionaria
161
162 end mriIntegral;
```

A.5 FSM para el control del generador de perfiles.

```
3     library IEEE;
4     use IEEE.STD_LOGIC_1164.all;
5
6     entity mrpFSM is
7          port(
8              RST : in STD_LOGIC;
9              CLK : in STD_LOGIC;
10             STR : in STD_LOGIC;
11             RDY : out STD_LOGIC;
12             CLR : in STD_LOGIC;
13             EP : in STD_LOGIC;
14             ET : in STD_LOGIC;
15             ARD : in STD_LOGIC;
16             EOP : out STD_LOGIC;
17             STI : out STD_LOGIC;
18             RDI : in STD_LOGIC;
19             IC : out STD_LOGIC;
20             EOI : out STD_LOGIC;
21             OPC : out STD_LOGIC_VECTOR(1 downto 0)
22             );
23    end mrpFSM;
24
25    architecture mrpFSM of mrpFSM is
26    signal Qn,Qp : std_logic_vector(5 downto 0);
27    begin
28
29        Comb: process(Qp,STR,RDI,EP,ET,ARD)
30        begin
31            case Qp is
32                when "000000" => if STR = '1' then
```

```
                                Qn <= "0001";
                            else
                                Qn <= Qp;
                            end if;
                    RDY <= '0';
                    EOP <= '0';
                    STI <= '0';
                    IC  <= '0';
                    EOT <= '0';
                    OPC <= "11";
            when "0001" => Qn <= "0010";
                                RDY <= '1';
                    EOP <= '0';
                    STI <= '0';
                    IC  <= '1';
                    EOT <= '0';
                    OPC <= "00";
            when "0010" => if RDI = '1' then
                                Qn <= "0011";
                            else
                                Qn <= Qp;
                            end if;
                    RDY <= '0';
                    EOP <= '0';
                    STI <= '0';
                    IC  <= '0';
                    EOT <= '0';
                    OPC <= "0C";
            when "0011" => if ET = '1' then
                                Qn <= "0100";
                            else
                                Qn <= "0010";
                            end if;
                    RDY <= '0';
                    EOP <= '0';
                    STI <= '0';
                    IC  <= '0';
                    EOT <= '0';
                    OPC <= "01";
            when "0100" => if EP = '1' then
                                Qn <= "1010";
                            else
                                Qn <= "0110";
                            end if;
                    RDY <= '0';
                    EOP <= '0';
                    STI <= '0';
                    IC  <= '0';
                    EOT <= '1';
                    OPC <= "16";
            when "0101" => if STR = '1' then
                                Qn <= "0111";
                            else
                                Qn <= Qp;
                            end if;
                    RDY <= '0';
                    EOP <= '0';
                    STI <= '0';
                    IC  <= '0';
                    EOT <= '0';
                    OPC <= "00";
            when "0110" => if ARD = '1' then
                                Qn <= "0111";
                            else
                                Qn <= Qp;
                            end if;
                    RDY <= '0';
                    EOP <= '0';
                    STI <= '0';
                    IC  <= '0';
                    EOT <= '0';
                    OPC <= "00";
            when "0111" => Qn <= "1000";
                                RDY <= '0';
                    EOP <= '0';
                    STI <= '1';
                    IC  <= '0';
                    EOT <= '0';
                    OPC <= "00";
            when "1000" => if RDI = '1' then
                                Qn <= "1001";
                            else
                                Qn <= Qp;
                            end if;
                    RDY <= '0';
                    EOP <= '0';
```

```
                        STI <= '0';
                        IC  <= '0';
                        EOT <= '0';
                        OPC <= "00";
            when "1001" => if ET = '1' then
                                Qn <= "1010";
                           else
                                Qn <= "0111";
                           end if;
                        RDY <= '1';
                        EOP <= '0';
                        STI <= '0';
                        IC  <= '0';
                        EOT <= '0';
                        OPC <= "01";
            when "1010" => if EP = '1' then
                                Qn <= "1110";
                           else
                                Qn <= "1011";
                           end if;
                        RDY <= '0';
                        EOP <= '0';
                        STI <= '0';
                        IC  <= '0';
                        EOT <= '1';
                        OPC <= "10";
            when "1011" => if ARD = '1' then
                                Qn <= "0101";
                           else
                                Qn <= Qp;
                           end if;
                        RDY <= '0';
                        EOP <= '0';
                        STI <= '0';
                        IC  <= '0';
                        EOT <= '0';
                        OPC <= "00";
            when others => Qn <= "0000";
                        RDY <= '0';
                        EOP <= '1';
                        STI <= '0';
                        IC  <= '0';
                        EOT <= '0';
                        OPC <= "00";
        end case;
    end process Comb;

    Sec: process(RST,CLK,Qn,CLR)
    begin
        if RST = '1' then
            Qp <= (others => '0');
        elsif CLK'event AND CLK = '1' then
            if CLR = '1'then
                Qp <= (others => '0');
            else
                Qp <= Qn;
            end if;
        end if;
    end process Sec;

end mrpFSM;
```

A.6 Registro.

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity mrpGReg is
    generic(n : integer := 8);
    port(
        RST : in STD_LOGIC;
        CLK : in STD_LOGIC;
        CLR : in STD_LOGIC;
        LDR : in STD_LOGIC;
        Din : in STD_LOGIC_VECTOR(n-1 downto 0);
        Dout : out STD_LOGIC_VECTOR(n-1 downto 0)
        );
end mrpGReg;
```

```
17
18  architecture mrpGReg of mrpGReg is
19  signal Qn,Qp : std_logic_vector(n-1 downto 0);
20  begin
21
22      Mux: process(CLR,LDR,Din,Qp)
23      begin
24          if CLR = '1' then
25              Qn <= (others => '0');
26          elsif LDR = '1' then
27              Qn <= Din;
28          else
29              Qn <= Qp;
30          end if;
31          Dout <= Qp;
32      end process Mux;
33
34      Reg: process(RST,CLK,Qn)
35      begin
36          if RST = '1' then
37              Qp <= (others => '0');
38          elsif CLK'event AND CLK = '1' then
39              Qp <= Qn;
40          end if;
41      end process Reg;
42
43  end mrpGReg;
```

A.7 Sumador con multiresolución de 255 bits.

```
1   -- Multiresolution 255 bits adder
2
3   library IEEE;
4   use IEEE.STD_LOGIC_1164.all;
5
6   entity mraAdd is
7       generic(
8           n : Integer := 8;
9           m : Integer := 3
10         );
11      port(
12          RST : in STD_LOGIC;
13          CLK : in STD_LOGIC;
14          STR : in STD_LOGIC;
15          RDY : out STD_LOGIC;
16          A   : in  STD_LOGIC_VECTOR(n-1 downto 0);
17          B   : in  STD_LOGIC_VECTOR(n-1 downto 0);
18          S   : out STD_LOGIC_VECTOR(n-1 downto 0);
19          IDX : out STD_LOGIC_VECTOR(m-1 downto 0);
20          WB : out STD_LOGIC;
21          RE : out STD_LOGIC
22         );
23  end mraAdd;
24
25  architecture mraAdd of mraAdd is
26
27      component mraAdder
28       generic(n : integer := 16);
29       port(
30          A  : in  STD_LOGIC_VECTOR(n-1 downto 0);
31          B  : in  STD_LOGIC_VECTOR(n-1 downto 0);
32          Ci : in  STD_LOGIC;
33          S  : out STD_LOGIC_VECTOR(n-1 downto 0);
34          Co : out STD_LOGIC);
35      end component;
36
37      component mraGCount
38       generic(n : integer := 3);
39       port(
40          RST : in STD_LOGIC;
41          CLK : in STD_LOGIC;
42          OPC : in STD_LOGIC_VECTOR(1 downto 0);
43          R : in STD_LOGIC_VECTOR(n-1 downto 0);
44          E : out STD_LOGIC;
45          Q : out STD_LOGIC_VECTOR(n-1 downto 0));
46      end component;
47
48      component mraCarry
```

```
49        port(
50            RST : in STD_LOGIC;
51            CLK : in STD_LOGIC;
52            OP : in STD_LOGIC;
53            Co : in STD_LOGIC;
54            Ci : out STD_LOGIC);
55        end component;
56
57        component mraFSM
58         port(
59            RST : in STD_LOGIC;
60            CLK : in STD_LOGIC;
61            STR : in STD_LOGIC;
62            RDY : out STD_LOGIC;
63            E : in STD_LOGIC;
64            OP : out STD_LOGIC_VECTOR(1 downto 0);
65            WR : out STD_LOGIC;
66            RD : out STD_LOGIC;
67            LDC : out STD_LOGIC);
68        end component;
69
70        signal Ci,Co,E,LDC : std_logic;
71        signal OP          : std_logic_vector(1 downto 0);
72        signal F           : std_logic_vector(m-1 downto 0);
73
74    begin
75
76        F <= (others => '1');
77
78        Suma:    mraAdder generic map (n) port map (A,B,Ci,S,Co);
79        Carry:   mraCarry port map (RST,CLK,LDC,Co,Ci);
80
81        Conta:  mraGCount generic map (m) port map (RST,CLK,OP,F,E,IDX);
82
83        Control: mraFSM port map (RST,CLK,STR,RDY,E,OP,WR,RD,LDC);
84
85    end mraAdd;
```

A.8 RAM con 2 puertos.

```
1     -- Internal RAM two ports
2
3     library IEEE;
4     use IEEE.STD_LOGIC_1164.all;
5     use ieee.numeric_std.all;
6
7     entity mriRAM_2P is
8          generic(
9              m : Integer := 8;
10             n : Integer := 7;
11             w : Integer := 128
12                );
13         port(
14             RST : in STD_LOGIC;
15             CLK : in STD_LOGIC;
16             WR : in STD_LOGIC;
17             DI : in STD_LOGIC_VECTOR(m-1 downto 0);
18             ADW : in STD_LOGIC_VECTOR(n-1 downto 0);
19             RDa : in STD_LOGIC;
20             DOa : out STD_LOGIC_VECTOR(m-1 downto 0);
21             ARa : in STD_LOGIC_VECTOR(n-1 downto 0);
22             RDb : in STD_LOGIC;
23             DOb : out STD_LOGIC_VECTOR(m-1 downto 0);
24             ARb : in STD_LOGIC_VECTOR(n-1 downto 0)
25             );
26    end mriRAM_2P;
27
28    architecture mriRAM_2P of mriRAM_2P is
29    subtype bit_width_RAM is std_logic_vector (m-1 downto 0);
30    type    Memoria is array(natural range <>) of bit_width_RAM;
31    signal MEM: Memoria(0 to (w-1));
32
33    begin
34
35        Write:process(RST,CLK,WR,ADW,DI)
36        begin
37            if (CLK'event AND CLK = '1') AND WR = '1' then
38                if RST = '1' then
```

```
39                  MEM(to_integer(unsigned(ADW))) <= (others => '0');
40              else
41                  MEM(to_integer(unsigned(ADW))) <= DI;
42              end if;
43          end if;
44      end process Write;
45
46      Read: process(RST,CLK,MEM,RDa,RDb,ARa,ARb)
47      begin
48          if (CLK'event AND CLK = '1') AND RDa = '1' then
49              if RST = '1' then
50                  DOa <= (others => '0');
51              else
52                  DOa <= MEM(to_integer(unsigned(ARa)));
53              end if;
54          end if;
55
56          if (CLK'event AND CLK = '1') AND RDb = '1' then
57              if RST = '1' then
58                  DOb <= (others => '0');
59              else
60                  DOb <= MEM(to_integer(unsigned(ARb)));
61              end if;
62          end if;
63      end process Read;
64
65  end mriRAM_2P;
```

A.9 Contador genérico.

```
3   library IEEE;
4   use IEEE.STD_LOGIC_1164.all;
5   use ieee.numeric_std.all;
6
7   entity mriCount is
8        generic(n : integer := 3);
9        port(
10           RST : in STD_LOGIC;
11           CLK : in STD_LOGIC;
12           OPC : in STD_LOGIC_VECTOR(1 downto 0);
13           K : in STD_LOGIC_VECTOR(n-1 downto 0);
14           E : out STD_LOGIC;
15           Q : out STD_LOGIC_VECTOR(n-1 downto 0);
16           QI : out STD_LOGIC_VECTOR(n-1 downto 0)
17           );
18  end mriCount;
19
20  architecture mriCount of mriCount is
21  signal Qn,Qp,Inc : std_logic_vector(n-1 downto 0);
22  begin
23
24      Increment: process(Qp)
25      begin
26          Inc <= std_logic_vector(unsigned(Qp) + 1);
27      end process;
28      QI <= Inc;
29
30      Count: process(Qp,K,OPC,Inc)
31      begin
32          case OPC is
33              when "00"   => Qn <= Qp;
34              when "01"   => Qn <= Inc;
35              when "10"   => Qn <= Qp;
36              when "11"   => Qn <= Inc;
37              when others => Qn <= (others => '0');
38          end case;
39
40          if Qp = K then
41              E <= '1';
42          else
43              E <= '0';
44          end if;
45          Q <= Qp;
46      end process Count;
47
48      Reg: process(RST,CLK,Qn)
```

```
47      begin
48          if RST = '1' then
49              Qp <= (others => '0');
50          elsif CLK'event AND CLK = '1' then
51              Qp <= Qn;
52          end if;
53      end process Reg;
54
55  end mriCount;
```

A.10 FSM para el control de la integral.

```
1   -- Integral control
2
3   library IEEE;
4   use IEEE.STD_LOGIC_1164.all;
5
6   entity mriFSM is
7        port(
8            RST : in STD_LOGIC;
9            CLK : in STD_LOGIC;
10           STB : in STD_LOGIC;
11           RDY : out STD_LOGIC;
12           E : in STD_LOGIC;
13           IC : in STD_LOGIC;
14           STA : out STD_LOGIC;
15           RDA : in STD_LOGIC;
16           OPS : out STD_LOGIC;
17           OPC : out STD_LOGIC_VECTOR(1 downto 0)
18           );
19  end mriFSM;
20
21  architecture mriFSM of mriFSM is
22  signal Qn,Qp : std_logic_vector(3 downto 0);
23  begin
24
25      Comb: process(Qp,STB,E,RDA,IC)
26      begin
27          case Qp is
28              when "0000" => if IC = '1' then
29                                  Qn <= "0001";
30                             elsif STB = '1' then
31                                  Qn <= "0100";
32                             else
33                                  Qn <= Qp;
34                             end if;
35                             RDY <= '1';
36                             STA <= '0';
37                             OPS <= '0';
38                             OPC <= "10";
39              when "0001" => Qn <= "0010";
40                             RDY <= '0';
41                             STA <= '1';
42                             OPS <= '1';
43                             OPC <= "01";
44              when "0010" => if RDA = '1' then
45                                  Qn <= "0011";
46                             else
47                                  Qn <= Qp;
48                             end if;
49                             RDY <= '0';
50                             STA <= '0';
51                             OPS <= '1';
52                             OPC <= "00";
53              when "0011" => if E = '1' then
54                                  Qn <= "0000";
55                             else
56                                  Qn <= "0001";
57                             end if;
58                        RDY <= '1';
59                             STA <= '0';
60                             OPS <= '1';
61                             OPC <= "00";
62              when "0100" => Qn <= "0101";
63                             RDY <= '0';
64                             STA <= '1';
```

```
                                    OPS <= '1';
                                    OPC <= "03";
                when "0101" => if RDA = '1' then
                                        Qn <= "0110";
                                    else
                                        Qn <= Qp;
                                    end if;
                                    RDY <= '0';
                                    STA <= '0';
                                    OPS <= '1';
                                    OPC <= "00";
                when "0110" => Qn <= "0111";
                                    RDY <= '0';
                                    STA <= '1';
                                    OPS <= '0';
                                    OPC <= "00";
                when "0111" => if RDA = '1' then
                                        Qn <= "1000";
                                    else
                                        Qn <= Qp;
                                    end if;
                                    RDY <= '0';
                                    STA <= '0';
                                    OPS <= '0';
                                    OPC <= "00";
                when others => if E = '1' then
                                        Qn <= "0000";
                                    else
                                        Qn <= "0110";
                                    end if;
                                    RDY <= '0';
                                    STA <= '0';0110";
                                    end if;
                                    RDY <= '0';
                                    STA <= '0';
                                    OPS <= '0';
                                    OPC <= "01";
            end case;
        end process Comb;

        Sec: process(RST,CLK,Qn)
        begin
            if RST = '1' then
                Qp <= (others => '0');
            elsif CLK'event AND CLK = '1' then
                Qp <= Qn;
            end if;
        end process Sec;

    end mriFSM;
```

A.11 Unidad de acarreo.

```
    library IEEE;
    use IEEE.STD_LOGIC_1164.all;

    entity mraCarry is
        port(
            RST : in STD_LOGIC;
            CLK : in STD_LOGIC;
            OP : in STD_LOGIC;
            Co : in STD_LOGIC;
            Ci : out STD_LOGIC
            );
    end mraCarry;

    architecture mraCarry of mraCarry is
    signal Qn,Qp : std_logic_vector(1 downto 0);
    begin

        Mux: process(OP,Co,Qp)
        begin
            if OP = '1' then
                Qn(0) <= Co;
            else
                Qn(0) <= '0';
            end if;
            Qn(1) <= Qp(0);
```

```
27          Qp(1) <= Qp(0);
28          Ci <= Qp(1);
29      end process Mux;
30
31      Reg: process(RST,CLK,Qn)
32      begin
33          if RST = '1' then
34              Qp <= (others => '0');
35          elsif CLK'event AND CLK = '1' then
36              Qp <= Qn;
37          end if;
38      end process Reg;
39
40  end mraCarry;
```

A.12 Sumador completo.

```
3   library IEEE;
4   use IEEE.std_logic_1164.all;
5   use ieee.numeric_std.all;
6
7   entity mraAdder is
8        generic(n : integer := 16);
9        port(
10           A  : in  STD_LOGIC_VECTOR(n-1 downto 0);
11           B  : in  STD_LOGIC_VECTOR(n-1 downto 0);
12           Ci : in  STD_LOGIC;
13           S  : out STD_LOGIC_VECTOR(n-1 downto 0);
14           Co : out STD_LOGIC
15          );
16  end mraAdder;
17
18  architecture mraAdder of mraAdder is
19  begin
20
21      process(A,B,Ci)
22      variable Suma : unsigned(n downto 0);
23      begin
24          Suma := unsigned('0' & A) + unsigned('0' & B) + unsigned('0'& Ci);
25          S    <= std_logic_vector(Suma(n-1 downto 0));
26          Co   <= Suma(n);
27      end process;
28
29  end mraAdder;
```

A.13 Contador genérico.

```
3   library IEEE;
4   use IEEE.STD_LOGIC_1164.all;
5   use ieee.numeric_std.all;
6
7   entity mraGCount is
8        generic(n : integer := 3);
9        port(
10           RST : in STD_LOGIC;
11           CLK : in STD_LOGIC;
12           OPC : in STD_LOGIC_VECTOR(1 downto 0);
13           K : in STD_LOGIC_VECTOR(n-1 downto 0);
14           E : out STD_LOGIC;
15           Q : out STD_LOGIC_VECTOR(n-1 downto 0)
16           );
17  end mraGCount;
18
19  architecture mraGCount of mraGCount is
20  signal Qn,Qp : std_logic_vector(n-1 downto 0);
21  begin
22
23      Count: process(Qp,E,OPC)
24      begin
25          case OPC is
```

```
            when "00"   => Qn <= Qp;
            when "01"   => Qn <= std_logic_vector(unsigned(Qp) + 1);--Qp + 1;
            when others => Qn <= (others => '0');
        end case;

        if Qp = K then
            E <= '1';
        else
            E <= '0';
        end if;
        Q <= Qp;
    end process Count;

    Reg: process(RST,CLK,Qn)
    begin
        if RST = '1' then
            Qp <= (others => '0');
        elsif CLK'event AND CLK = '1' then
            Qp <= Qn;
        end if;
    end process Reg;

end mraGCount;
```

A.14 FSM para el control del sumador con multiresolución.

```
-- Multiresolution control
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity mraFSM is
     port(
         RST : in STD_LOGIC;
         CLK : in STD_LOGIC;
         STR : in STD_LOGIC;
         RDY : out STD_LOGIC;
         E : in STD_LOGIC;
         OP : out STD_LOGIC_VECTOR(1 downto 0);
         WR : out STD_LOGIC;
         RD : out STD_LOGIC;
         LDC : out STD_LOGIC
         );
end mraFSM;

architecture mraFSM of mraFSM is
signal Qn,Qp : std_logic_vector(1 downto 0);
begin

    Comb: process(STR,E,Qp)
    begin
        case Qp is
            when "00"   => if STR = '1' then
                                Qn <= "01";
                           else
                                Qn <= Qp;
                           end if;
                           RDY <= '1';
                           OP  <= "10";
                           WR  <= '0';
                           RD  <= '0';
                           LDC <= '0';
            when "01"   => Qn <= "10";
                           RDY <= '0';
                           OP  <= "00";
                           WR  <= '0';
                           RD  <= '1';
                           LDC <= '1';
            when others => if E = '1' then
                                Qn <= "00";
                           else
                                Qn <= "01";
                           end if;
                           RDY <= '0';
```

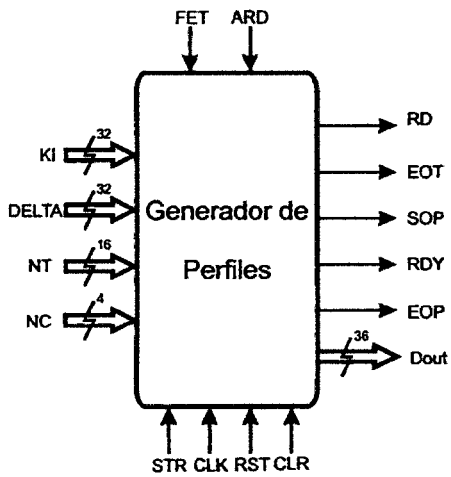
```
48                                  OP  <= "01";
49                                  WR  <= '1';
50                                  RD  <= '0';
51                                  LDC <= '1';
52          end case;
53      end process Comb;
54
55      Sec: process(RST,CLK,Qn)
56      begin
57          if RST = '1' then
58              Qp <= (others => '0');
59          elsif CLK'event AND CLK = '1' then
60              Qp <= Qn;
61          end if;
62      end process Sec;
63
64  end mraFSM;
```

APÉNDICE B.

Hoja de Datos del generador de perfiles en la cual se presentan los aspectos más importantes a tener en cuenta para el correcto funcionamiento del generador de perfiles.

Generador de perfiles
Hoja de Datos

Configuración de entradas y salidas para el generador de perfiles.

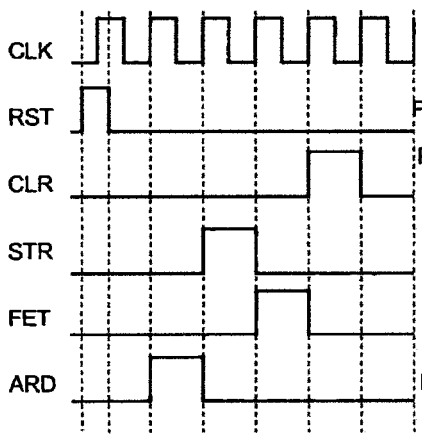


Nombre	Tipo	Ancho	Descripción
Delta	E	32	Dato de entrada
NC	E	3	Grado del polinomio
NT	E	16	No. de trazos
KI	E	32	No. de muestra de cada trazo
FET	E	1	Finalizar trazo
ARD	E	1	Permitir lectura
STR	E	1	Inicio del perfil
CLK	E	1	Reloj maestro 50Mhz
RST	E	1	Reset asíncrono
CLR	E	1	Borrado síncrono
RD	S	1	Lectura
EOT	S	1	Fin del trazo
SOP	S	1	Inicio del perfil
RDY	S	1	Ready
EOP	S	1	Fin del perfil.
Dout	S	36	Dato de salida

Tiempo de procesamiento por muestra: 12.003 ns

Frecuencia máxima: 103.477MHz

Activación de las señales de control:



Las señales STR, FET y ARD deben ser activadas en una parte activa del reloj maestro

El Formato de salida mínimo debe de tener cuando menos un bit en la parte fraccionaria

La señal FET puede ser activada arbitrariamente después de la señal STR.

Formato de los datos de entrada: 64.192

Formato de los datos de salida:18.18

APÉNDICE C.

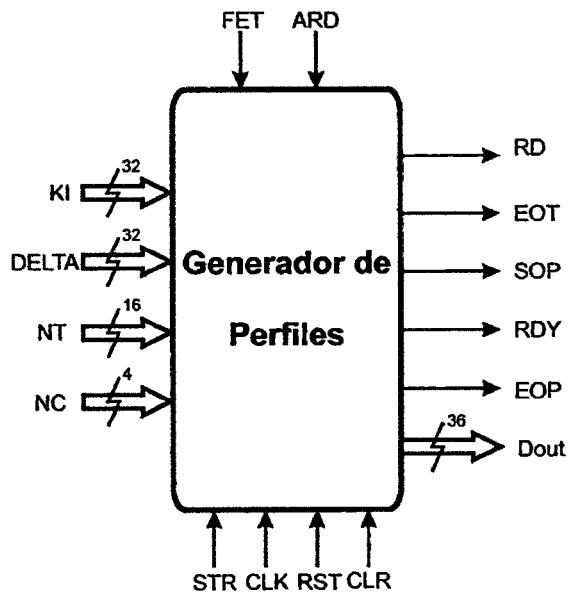
En esta sección se presenta el manual de usuario, que contiene datos más específicos sobre el funcionamiento del generador de perfiles y su estructura digital, así como los formatos de los datos de entrada y salida, el grafo de la FSM y la simulación del core.

Generador de Perfiles Polinomiales

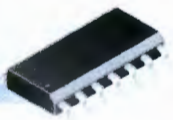
Manual de Usuario

Universidad Autónoma de Querétaro
Campus San Juan del Río

Generador de Perfiles Polinomiales



Core basado en tecnología FPGA diseñado de forma genérica con el propósito de poderse utilizar en diferentes aplicaciones teniendo la capacidad de compatibilidad con otros sistemas basados en dicha tecnología.



Descripción de señales de entrada y salida.

Nombre	Tipo	Ancho	Descripción
Delta	E	32	Dato de Entrada.
NC	E	4	Grado del Polinomio
NT	E	16	No. de Trazos
KI	E	32	No. de Muestra
FET	E	1	Finalizar Trazo
ARD	E	1	Permitir Lectura
STR	E	1	Inicio del Perfil
CLK	E	1	Reloj Maestro 50MHz
RST	E	1	Reset Asíncrono
CLR	E	1	Borrado Síncrono
RD	S	1	Lectura
EOT	S	1	Fin del Trazo
SOP	S	1	Inicio del Perfil
RDY	S	1	Ready
EOP	S	1	Fin del Perfil
Dout	S	36	Dato de Salida

El formato de datos de entrada (Deltas): 64.192

El formato de datos de salida: 18.18

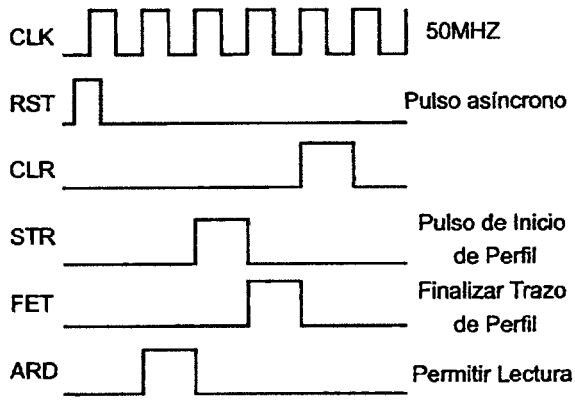
Nótese que el formato de las Deltas es 64.192, es decir, el ancho de palabra del dato de entrada es de 256 bits, sin embargo, el bus de entrada para las Deltas es de 32 bits, la finalidad principal del cambio en el tamaño del bus es para la reducción de los recursos necesarios a ocupar en el FPGA.

Por lo tanto es necesario seccionar la Delta original en 8 partes de 32 bits y posteriormente el generador de perfiles se encargará de unir las ocho secciones mediante una memoria RAM para su procesamiento.

Utilización de recursos en un FPGA xc3s200 con 200 000 compuertas disponibles

Slices:	10%
Slice Flip Flops:	03%
GCLKs:	12%
Block RAMs	16%

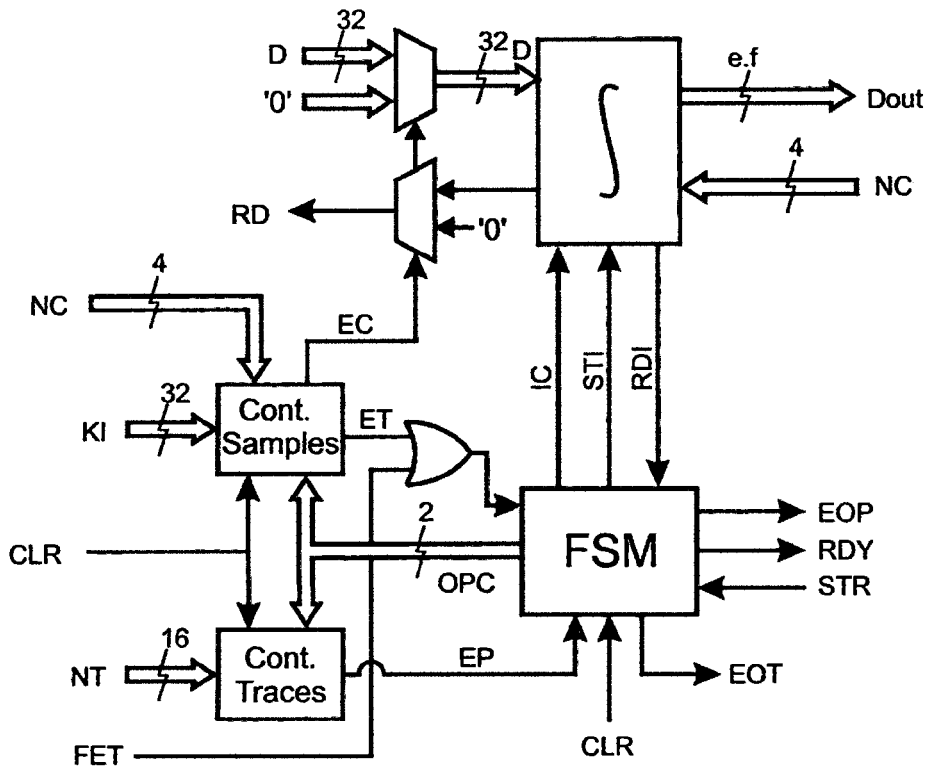
Activación de las señales de control:



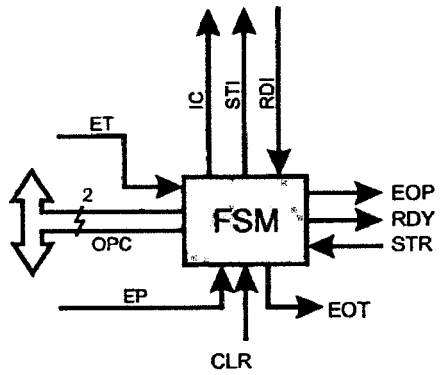
Tiempo de procesamiento por muestra: 12.003 ns
Frecuencia máxima: 103.477MHz

La señal NC indica el grado del polinomio *(n)* y está configurada para obtener el perfil de velocidad de un polinomio, para reconstruir el polinomio en posición es necesario introducir el grado del polinomio mas 1, es decir, *n+1*.

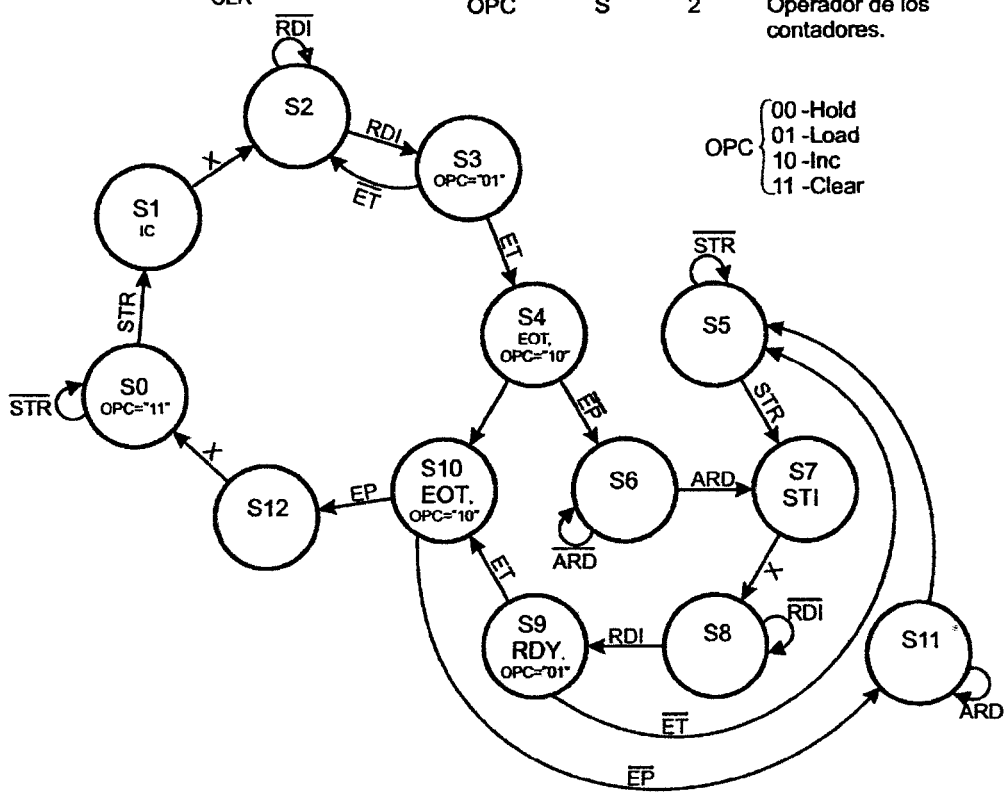
Estructura digital del generador de perfiles.



El formato de salida de Dout tiene un ancho de 36 bits, teniendo la posibilidad de ajustarlos mediante las variables e.f (e: parte entera, f:parte fraccionaria).

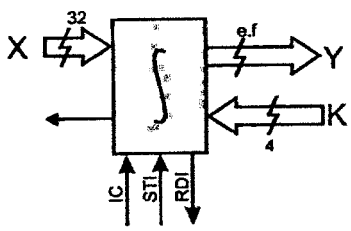


NOMBRE	TIPO	ANCHO	DESCRIPCIÓN
CLK	E	1	Reloj 50 Mhz
RST	E	1	Reset asíncrono
STR	E	1	Señal de inicio
CLR	E	1	Borrado síncrono
RDI	E	1	Lectura de la integral
ET	E	1	Contador de muestras
EP	E	1	Contador de trazos
RDY	S	1	Ready
EOT	S	1	Fin del trazo
EOP	S	1	Fin del perfil
IC	S	1	
STI	S	1	Inicio de la integral
OPC	S	2	Operador de los contadores.

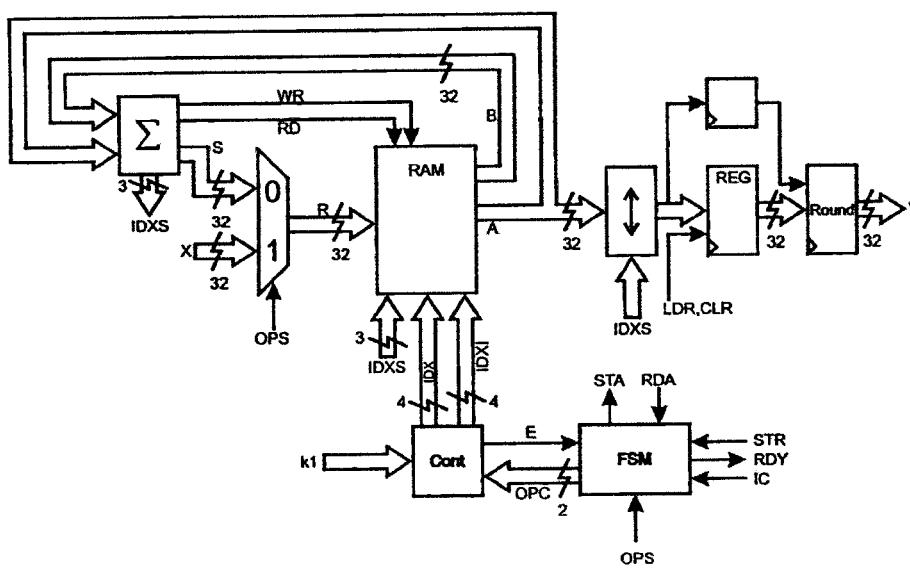


OPC { 00 -Hold; 01 -Load; 10 -Inc; 11 -Clear

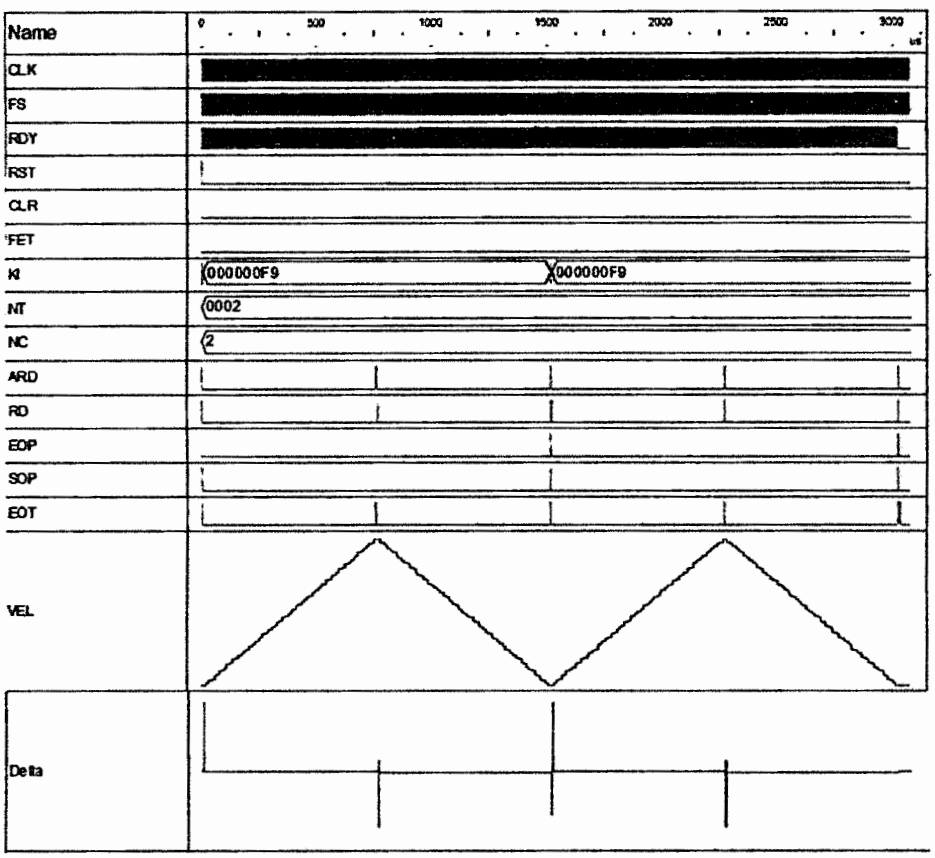
Estructura digital de la integral.



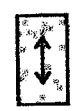
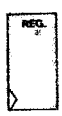
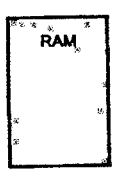
NOMBRE	TIPO	ANCHO	DESCRIPCIÓN
CLK	E	1	Reloj 50 Mhz
RST	E	1	Reset asíncrono
CLR	E	1	Borrado síncrono
IC	E	1	
STI	E	1	Inicio de la integral
K	E	4	No. de integraciones
X	E	32	Dato de entrada
RDY	S	1	Ready
RDI	S	1	Lectura de los datos
Y	S	32	Dato de salida



SIMULACION DEL GENERADOR DE PERFILES.



Glosario de módulos

Módulo	Descripción	Módulo	Descripción
	Integral		Modulo de redondeo
	Multiplexor		Módulo de ajuste
	Registro		Contador de muestras
	RAM		Contador
	Sumador		Maquina de Estados Finitos
	Contador de trazos	**Señal**	**Descripción**
			Señal de n bits
			Señal de un bit