ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

INGENIERÍA TÉCNICA EN INFORMÁTICA DE SISTEMAS

Curso Académico 2010/2011

Proyecto Fin de Carrera

Diseño e implementación en FPGA del control numérico computarizado para fresadora de tres ejes: control manual e interpretación de coordenadas en dos dimensiones.

Autor: Alfonso de Lara Rubio
Tutor: Felipe Machado Sánchez
Cotutor: Susana Borromeo López

Agradecimientos

Me siento profundamente satisfecho y felíz de haber terminado esta etapa de mi vida de la forma en la que lo he hecho. Es muy típico decirlo, pero no se me ocurre una mejor forma de decir que ha sido posible gracias a ellos, a la presencia de esas personas que han estado a mi lado durante todo este tiempo. Me gustaría agradecerles todo lo que han hecho por mí sin importar cómo ni cuando, haciendo especial mención a:

Mis padres, por su dedicación, cariño e incondicional apoyo.
Vero, por estar en mi vida y hacerme feliz.
Mis tíos Paqui, Alberto y Leo, por ser mis otros padres.
Mi tía Mari, nunca te olvidaré.
Mis primos, por darme tan buenos ratos.
Al resto de mi familia, por ser como son.
Mis compañeros de clase:
Sergio, ojalá y todos fuesen igual de malos que tú.
Franchu, he olvidado lo que te iba a decir.
Rober, por compartir la carga conmigo.
Héctor, por compartir su mala suerte.
Antoñito, por ser tan sabio.
Piedad, por ser tan chica.
Luisete, Jose, Javi y David, sin ellos la universidad habría sido mucho más aburrida.
Mi tutor Felipe, por darme un empujón para alcanzar esta meta.
Los profes del DTE Susana, Joaquín y Cristina, por echarme un buen cable.
Víctor, por darme unas ideas para la fresadora.

Por último, dar gracias de todo corazón a mi abuela Ana. Por dotar de significado a los términos: ejemplo, perseverancia, tesón, alegría, bondad y amor. Y por ser la mejor persona que haya conocido.

Resumen

Una fresadora es una máquina herramienta[1] utilizada para realizar mecanizados[2]. El Control Numérico Computarizado (CNC) es la técnica por la que se dirige el posicionamiento de un dispositivo mecánico móvil mediante órdenes en tiempo real basadas en un modelo numérico. La aplicación de sistemas de control numérico por computadora en las máquinas-herramienta permite aumentar la productividad[3] y posibilita realizar operaciones con un elevado grado de precisión dimensional. Sin embargo, la inclusión de este tipo de control implica aumentar considerablemente el coste de la maquinaria e incluir equipos externos donde realizar ese control.

Por otra parte, los dispositivos de lógica programable, y en particular las FPGAs (*Field Programmable Gate Array*), están sustituyendo a los microprocesadores de propósito general y a los ASICs (*Application Specific Integrated Circuit*) en los sistemas de control.

En este proyecto fin de carrera, mediante el uso de dispositivos de lógica programable, se ha conseguido implementar el control numérico computarizado para una fresadora de tres ejes. Se ha implementado un diseño en VHDL, modular y configurable que permite su uso en gran número de aplicaciones para todo aquello relacionado con trazado en dos dimensiones (impresoras, plotters, etc). Un ejemplo de ello ha sido la integración de este proyecto con el trabajo realizado en el proyecto fin de carrera de Héctor Gálvez Barrios para interpretar modelos de dos dimensiones partiendo de una imagen.

Con la intención de validar el control implementado sobre la FPGA se ha construido un prototipo funcional de fresadora de tres ejes. Para ello se ha realizado tanto la parte mecánica, diseño y fabricación, como la electrónica de control de la misma.

Índice

Índice de figuras

Capítulo 1: Introducción

La escultura es la más antigua de las técnicas y un claro ejemplo de modelado sobre un material base. En sus inicios los escultores realizaban sus obras en materiales como madera, piedra o hueso empleando para ello rudimentarias herramientas y técnicas. A lo largo del tiempo tanto técnicas como herramientas han sufrido una serie de evoluciones, llegando en algunos sectores de la industria a eliminar por completo el factor humano, es decir, la artesanía manual ha sido sustituida por la producción en serie mediante el empleo de maquinaria, capaz de reproducir modelos de una forma más precisa, rápida y eficiente. El desarrollo tecnológico ha desembocado en la aparición numerosas técnicas de producción pero nosotros nos vamos a centrar en la más parecida a la escultura, eso si, mediante una máquina y que se conoce con el nombre de mecanizado.

En el marco actual, los proyectos que impliquen el diseño y la proyección de modelos físicos requieren de un prototipo, o en su defecto una réplica a escala, en determinadas fases de su desarrollo. Por ende, se hace necesaria la producción de las partes específicas que no sean comerciales y que formen parte del proyecto. Para obtenerlas o bien se opta por contratar a terceros (empresas especializadas) o se pueden llevar a cabo mediante una producción propia. En los últimos tiempos cada vez se recurre más a la segunda opción debido al uso de maquinaria lo más genérica posible que sea capaz de producir diseños personalizados que se ajusten a las necesidades de cada proyecto.

Es en este punto donde el mecanizado se convierte en una buena opción para la producción a baja escala. Las ventajas que ofrece son:

a) El abaratamiento de costes: ya que para producciones muy pequeñas es notable-mente más barato que la producción por terceros.

b) La confidencialidad de los modelos: los cuales han llegado en muchos casos a ser filtrados por la competencia pagando a las empresas productoras por los planos.

Otro aspecto que diferencia el empleo del mecanizado de una producción masiva en serie es que requiere menor cantidad de maquinaria. En muchos casos basta con tener una herramienta de mecanizado, lo cuál también puede ser una ventaja, sobre todo para pequeñas y medianas empresas. La herramienta de mecanizado más extendida es el torno, pero la más polivalente es la fresadora.

Antiguamente este tipo de máquinas eran manuales y la calidad de la producción dependía de personal especializado en su uso (torneros o matriceros). Pero la modernización de las mismas ha desembocado en su automatización, siendo posible su manejo a través de un equipo (computadora), lo que nos permite una mayor precisión y una casi completa eliminación de las diferencias a la hora de producir copias de un mismo modelo.

Hay distintos modelos de fresadoras en el mercado actual. Las de mayor coste suelen llevar integrado un equipo dedicado exclusivamente para su manejo. Por otro lado se encuentra la opción más económica que requiere el uso de un ordenador auxiliar, de este modo la máquina no es por sí sola independiente.

El empleo de dispositivos de lógica programable, y en particular las FPGAs (*Field Programmable Gate Array*), ha aumentado considerablemente para su uso como elemento de control sustituyendo a los microprocesadores de propósito general y a los ASICs (*Application Specific Integrated Circuit*).

En este proyecto fin de carrera se propone el uso de FPGAs como elemento de control para eliminar la dependencia del PC externo, abaratar costes y reducir el tamaño del equipo de control auxiliar.

1.1 El Control Numérico Computarizado y la Fresadora

El Control Numérico Computarizado (CNC) es la técnica por la que se dirige el posicionamiento de un dispositivo mecánico móvil mediante órdenes en tiempo real basadas en un modelo numérico.

Existen diversos tipos de máquinas que pueden ser comandadas por esta metodología pero nos vamos a centrar en el manejo de una fresadora de tres ejes. La fresadora se caracteriza principalmente porque el movimiento no se aplica sobre el material a moldear sino que en este caso la parte móvil es la herramienta que moldea. Como mínimo las fresadoras poseen tres ejes de movimiento, pero esto no significa que no existan ejemplares con mayor número de ejes. De hecho, cuanto mayor sea su número, mayores y más complejos serán los modelos que pueda reproducir la máquina.

Al igual que en el modelado, a la hora de mecanizar se precisa establecer un sistema de coordenadas. Dicho sistema se tratará de compatibilizar al máximo con el del modelo para ser lo más fiel posible a la hora de ser producido.

El sistema de coordenadas a su vez se verá limitado por la componente mecánica. Claro está que cuanto mayor sea la resolución de la componente mecánica, más precisos podrán ser los modelos obtenidos.

En nuestro caso, trataremos con un prototipo de fresadora de tres ejes completamente funcionales. Existen fresadoras de tres ejes cuyo eje Z tiene dos posibles movimientos, es decir, solo hacen posible el corte o fresado en dos dimensiones. Un buen ejemplo de ello es la microfresadora que posee el Departamento de Tecnología Electrónica (DTE) para crear circuitos impresos, una LPKF ProtoMat C60[4].

1.2 El lenguaje de descripción hardware VHDL

Los lenguajes de descripción hardware, del inglés *Hardware Description Languages* (HDL), tienen sus orígenes en los años 70 enfocados al diseño de circuitos digitales mediante herramientas de CAD (*Computer Aided Design*) electrónico. Los

primeros lenguajes no tuvieron apenas impacto en la industria, ni tampoco a nivel académico.

A mediados de los 80 aparecieron los lenguajes VHDL[5][6][7][8] y Verilog[9] que si tuvieron un importante auge, imponiéndose como herramientas de desarrollo electrónico. Actualmente ambos lenguajes lideran su sector, desplazando cada vez más al resto de lenguajes todavía soportados por algunas herramientas de CAD.

Las siglas VHDL son el resultado de la conjunción de VHSIC, que corresponden a las siglas en inglés de Circuitos Integrados de Muy Alta Velocidad (*Very High Speed Integrated Circuit*) y las anteriormente mencionadas HDL. Este lenguaje posee una sintaxis que se asemeja a la de algunos lenguajes de programación de alto nivel como ADA. Su enfoque hacia la especificación y documentación de sistemas digitales hizo que su semántica estuviera orientada al modelado de hardware. Las cualidades más destacables son las siguientes:

- ***Es un lenguaje formal y está normalizado***: que sea formal elimina la ambigüedad a la hora de describir mediante este la estructura o el funcionamiento de un circuito. Al estar normalizado brinda la posibilidad de ser compatible con cualquier herramienta que se ajuste a su norma oficial.
- ***Altamente descriptivo***: VHDL permite trabajar con numerosas metodologías de diseño, no es dependiente de la tecnología y la descripción de hardware puede poseer distintos niveles de abstracción.

1.3 FPGAs para prototipado

Las FPGAs son dispositivos de lógica programable. Sus siglas hacen referencia a Array de Puertas Lógicas Programables (del inglés *Field Programmable Gate Array*). Están formadas por cadenas bidimensionales de bloques lógicos y *flip-flops* con una serie de interconexiones que los hacen eléctricamente programables (ver Figura 1).

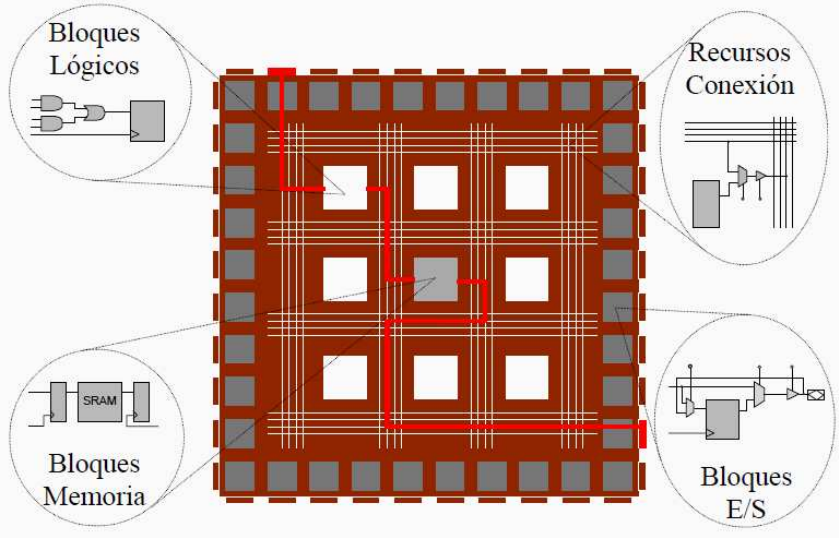

Figura 1: Estructura simplificada de la arquitectura interna de una FPGA

Estas interconexiones funcionan a modo de interruptores que se pueden programar mediante señales eléctricas y precisamente esto lo que las diferencian de los Circuitos Integrados. Los bloques lógicos de una FPGA pueden ser programados para dar una funcionalidad simple como podría ser la de un transistor o tan compleja como la de un microprocesador. Para que brinden funcionalidades complejas basta con combinar los bloques lógicos necesarios. Se pueden usar para implementar también funciones lógicas secuenciales o diferentes combinaciones de circuitos combinacionales.

Las FPGAs se introdujeron como alternativa a los ASICs para implementar sistemas complejos en un solo chip y para ofrecer flexibilidad de reprogramación al usuario que no ofrecían estos. Otra ventaja importante es que por medio de herramientas de diseño asistido por ordenador (CAD) podrían implementarse circuitos en un corto periodo de tiempo lo cual lo hace una herramienta tremendamente útil para el diseño y prueba de prototipos electrónicos. Con ello se eliminaban los procesos de *layout* físicos hasta que no se tenía una versión definitiva del producto ahorrando costes tanto en el proceso como en el tiempo de desarrollo reduciendo el *Timetomarket*.

En el apartado 3.5 se describe en detalle la metodología de diseño con FPGAs.

Capítulo 2: Objetivos

En el mercado actual un equipo profesional completo de fresado por CNC tiene un elevado coste, sin embargo, existen opciones más económicas que cubren perfectamente las necesidades de las pequeñas y medianas empresas. Estas opciones se basan en la obtención de maquinaria de menores prestaciones y por ello menor equipación. En estos casos se suprimen de la máquina elementos como la electrónica y el ordenador dedicado a su manejo, dando libertad al usuario final a la hora de elegir estas dos partes del equipo entre la gran variedad que ofrece actualmente el mercado. Dado que ambos elementos suponen un coste adicional significativo hemos indagado en la opción de incorporar el dispositivo de control sin elevar en exceso el coste de dichas alternativas más baratas.

El objetivo principal de este proyecto es realizar el Control Numérico Computarizado para una fresadora de tres ejes mediante el uso de dispositivos de lógica programable, en concreto FPGAs.

Para lograr el objetivo anterior se han planteado una serie de objetivos secundarios:

1. Desarrollo modular del sistema propuesto.
2. Diseño de un prototipo funcional de Fresadora para poder validar el Control Numérico que se va a desarrollar en este proyecto.
3. Minimizar el coste del prototipo como consecuencia del bajo presupuesto disponible.
4. Integración del Control Numérico con otro sistema generador de órdenes independiente. Incluida la validación de la misma.

Capítulo 3: Materiales y metodologías empleados

El proyecto se ha dividido en varias fases diferenciadas:

1) Desarrollo del prototipo de Fresadora de tres ejes: como ya se ha mencionado, dada la imposibilidad de adquirir una fresadora y para validar el trabajo de este proyecto fin de carrera se ha propuesto como objetivo adicional el desarrollo de un prototipo. Este desarrollo se ha dividido en:

- Diseño y construcción de la parte mecánica**.**
- Diseño de la electrónica de control de los motores**.**

2) Implementación del Control Numérico sobre una FPGA: mediante un diseño modular y escalable.

3) **Integración con el Módulo de Generación de Coordenadas:** incluyendo su posterior validación.

3.1 Entornos de desarrollo

Entre los entornos de desarrollo vamos a distinguir dos grupos dependiendo del objetivo para el que se han empleado:

- Diseño digital: todo lo que hacer referencia a la implementación, simulación y programación de la FPGA. A este grupo pertenecen Xilinx ISE[10], ModelSIM[11] y Digilent Adept[12].
- Diseño electrónico: software empleado para implementar el control electrónico de los motores que mueven nuestro prototipo de Fresadora. Grupo compuesto por

el entorno Kicad[13] y el paquete de software LPKF necesario para trazar circuitos con la microfresadora del LabTel[14], modelo ProtoMat C60[15].

3.1.1 Xilinx ISE

ISE (*Integrated Software Environment*) de Xilinxes un entorno informático compuesto por un conjunto de herramientas que asisten en el proceso de diseño, simulación, síntesis del resultado y configuración del hardware. Permite diseñar circuitos digitales por medio de esquemas lógicos, máquinas de estados o bien utilizando lenguajes de descripción de hardware como por ejemplo VHDL o Verilog.

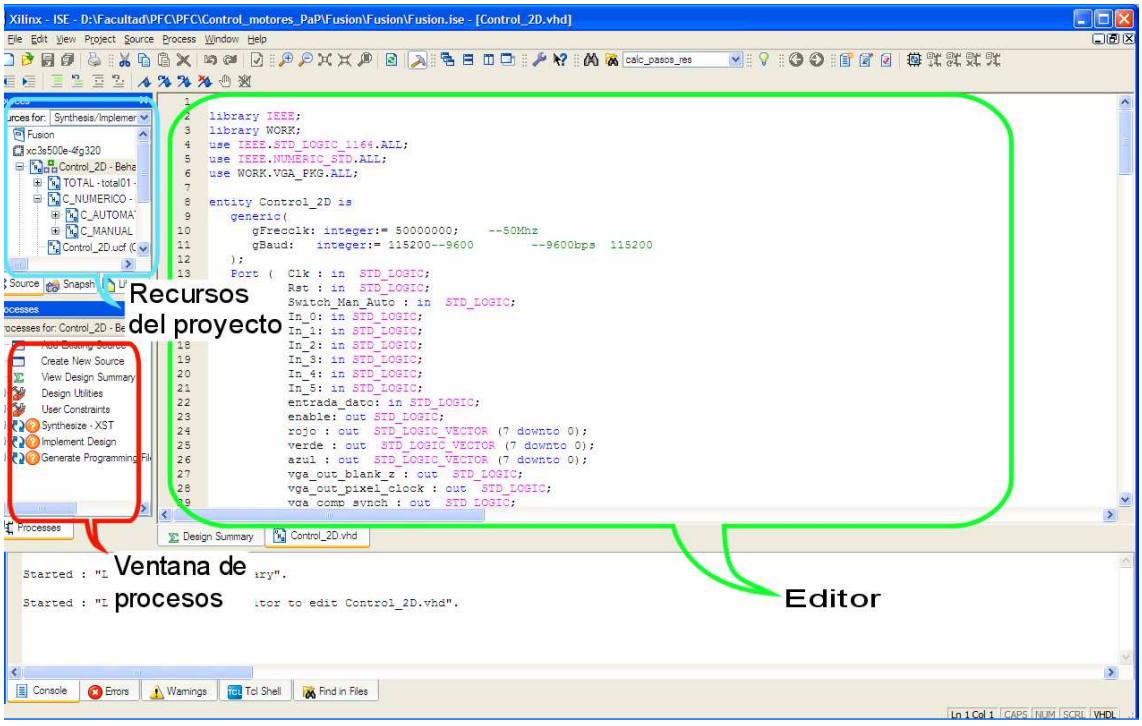

Figura 2: Captura de Xilinx ISE

La herramienta se divide en varias ventanas. La mayor y a la vez principal es el editor. Podemos ver a la izquierda otro par de ventanas, una primera de recursos del proyecto y otra de procesos asociados a él (ver Figura 2). En su entorno hemos optado por desarrollar la captura de los diferentes diseños para los módulos que componen nuestro proyecto empleando el lenguaje de tipo descriptivo VHDL. A su vez se han creado con esta herramienta los ficheros de test o estímulos, también llamados *Testbench*, para comprobar en simulaciones el funcionamiento de los circuitos.

Xilinx ISE también cuenta con una herramienta de simulación que analiza el comportamiento de los circuitos a nivel funcional. A este nivel no se tienen en cuenta los retardos provocados por el hardware así que este tipo de simulación no nos conviene. Para realizar las simulaciones funcionales hemos empleado la herramienta ModelSim que se expone a continuación.

3.1.2 ModelSim

ModelSim de Mentor Graphics es una herramienta software que nos permite editar, compilar, simular y depurar diseños de sistemas digitales descritos mediante lenguajes como VHDL y Verilog. También nos brinda la posibilidad de realizar diseños mixtos, es decir, con bloques realizados en ambos lenguajes.

Su interfaz gráfica de usuario (GUI: *Graphical User Interface*) de fácil manejo nos permite de un modo rápido identificar y depurar los diferentes errores que puedan surgir en el funcionamiento de los circuitos. En esta GUI nos muestra de forma gráfica una interpretación de todas las señales que formen parte de nuestros diseños, pudiendo también en ella modificar y recompilar insitu el código para poder volver a simularlo. No solo facilita la depuración, sino que a la vez agiliza la detección y depuración de errores puesto que no hace falta estar generando y volcando los modelos a la FPGA para llevar a cabo las distintas pruebas.

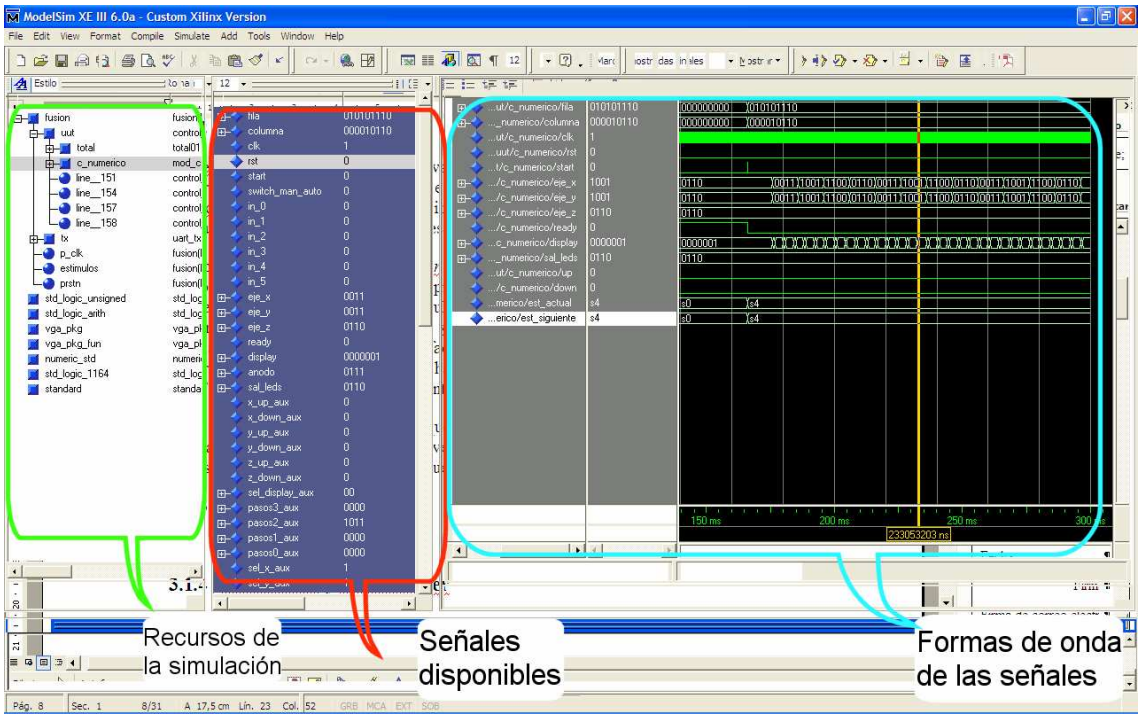

Figura 3: Simulación de nuestro diseño con ModelSim

Para aprovechar las posibilidades que nos brinda esta herramienta, se han simulado previamente todos los bloques y una vez depurados se implementan en la FPGA de la placa de desarrollo. La Figura 3 es un ejemplo de simulación de nuestro diseño. En ella se puede ver la pantalla de formas de onda donde aparecen todas las señales relacionadas con nuestro diseño y su evolución temporal.

3.1.3 Digilent Adept

Software dedicado a la programación y configuración de las FPGAs fabricadas por Digilent. Capaz de transferir datos desde y hacia las FPGA. Permite consultar registros específicos. Para programar únicamente necesita el fichero generado por Xilinx ISE de extensión *.bit*. Para reprogramar la memoria se usará el tipo de archivo de extensión *.mcs*. Su interfaz puede verse en la Figura 4.

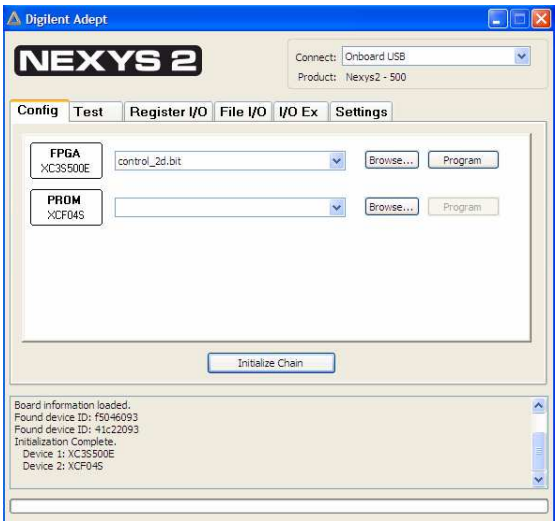

Figura 4: Captura de Adept

Hicimos uso de dos modos de programar con *Adept*. En las pruebas se usó el modo de programación directo sobre la memoria *flash* que en caso de hacer modificaciones en el código se sobrescribe de forma rápida. Esta forma no hace persistente el código en la placa y si esta se apaga se tendrá que volver a programar.

El otro modo consiste en programar la EEPROM (*Electrically-Erasable Programmable Read Only Memory*) para que el código persista y quede almacenado. Solo se hizo uso de este modo cuando alguna parte del proyecto estaba en versiones avanzadas. Ambas programaciones se realizan a través del puerto *miniUSB* integrado en la placa de desarrollo Nexys II.

3.1.4 KiCAD

KiCad es un programa de código libre (GPL) para la creación de esquemas electrónicos y circuitos impresos. La suite Kicad está compuesta de un conjunto de cuatro programas y un gestor de proyectos:

- **EeSchema**: programa editor de esquemáticos. Posee una librería de componentes predefinidos formada por los componentes más comunes. Por si estuviéramos interesados en el empleo de un componente que no se encuentra disponible en su librería, proporciona una herramienta muy sencilla para crear nuestros propios componentes.

- **PcbNew**: para la realización de circuitos impresos. Al igual que en EeSchema la librería podría ser más completa, pero como también incluye una herramienta de edición de componentes y huellas se ha podido suplir esta carencia. Este programa también nos permite, entre otras funciones, visualizar los circuitos creados en tres dimensiones.
- **Gerbview**: visualización de documentos generados en formato GERBER (documentos de fototrazado).
- **Cvpcb**: utilidad de asociación de las huellas físicas a los componentes electrónicos utilizados en el esquemático.

En el apartado 4.2 se muestran diversas capturas de este software. Todas ellas corresponden a la electrónica de control de los motores de la Fresadora. Tanto los esquemáticos como el diseño del circuito PCB se han extraído de Kicad.

3.1.5 LPKF CircuitCam, BoardMaster y ProtoMat C60

CircuitCam y BoardMaster son los dos programas que componen el paquete software del fabricante de microfresadoras para circuitos impresos LPKF

- ***CircuitCam***, mostrado en la Figura 5, es un programa de importación de ficheros de fototrazado al formato del tipo conveniente para la su posterior trazado. Permite editar los ficheros y añadir los trazos de corte del borde de las placas que compondrán nuestros circuitos impresos. Desde esta herramienta se obtienen ficheros con extensión *.LMD* a partir de ficheros de fototrazado que en nuestro caso son de tipo GerberX.

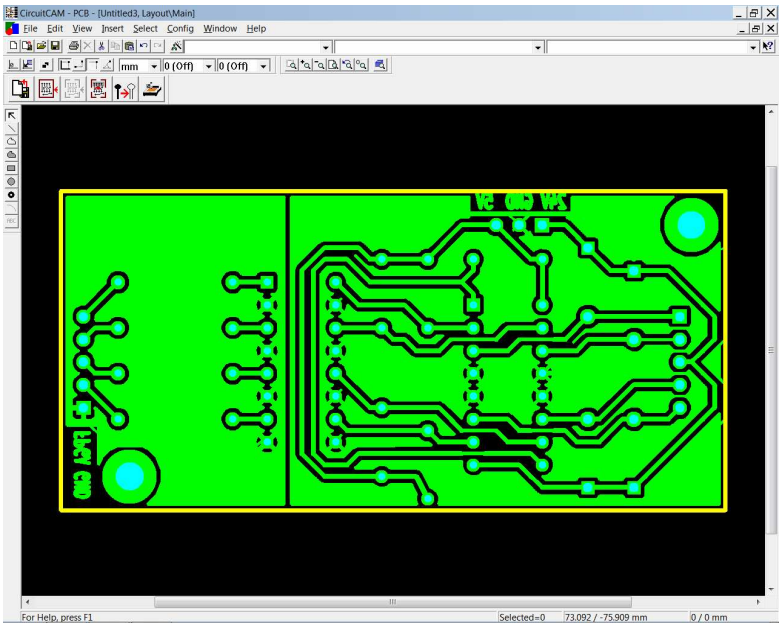

Figura 5: Captura de CircuitCam

- ***BoardMaster*** es el software encargado de trazar a partir del fichero de extensión *.LMD* nuestros circuitos impresos por medio de la microfresadora. A través de su interfaz, mostrada en la Figura 6, posicionaremos nuestros modelos en las planchas con capas de cobre y seremos guiados por las distintas etapas del trazado para el correcto funcionamiento de la ProtoMat C60.

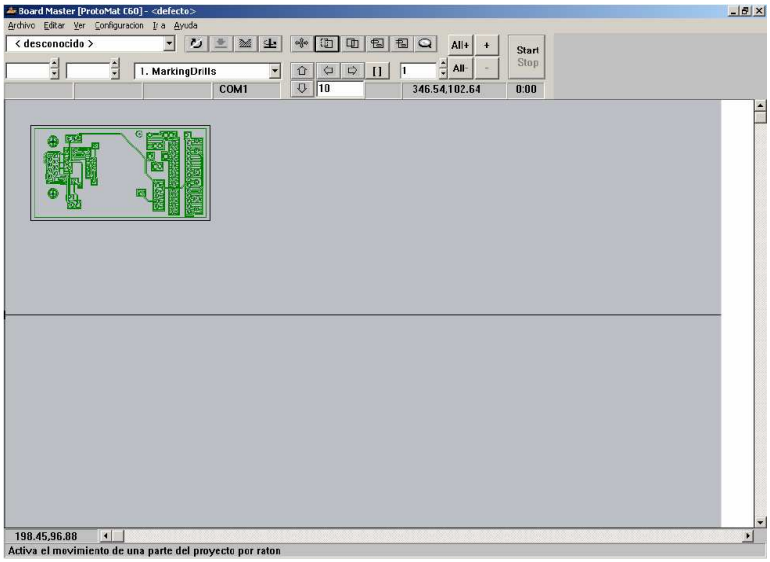

Figura 6: Captura de BoardMaster

El LabTel de la URJC cuenta con una LPKF Protomat C60, como la de la Figura 7, en la que se han realizado los distintos circuitos necesarios para el desarrollo del proyecto.

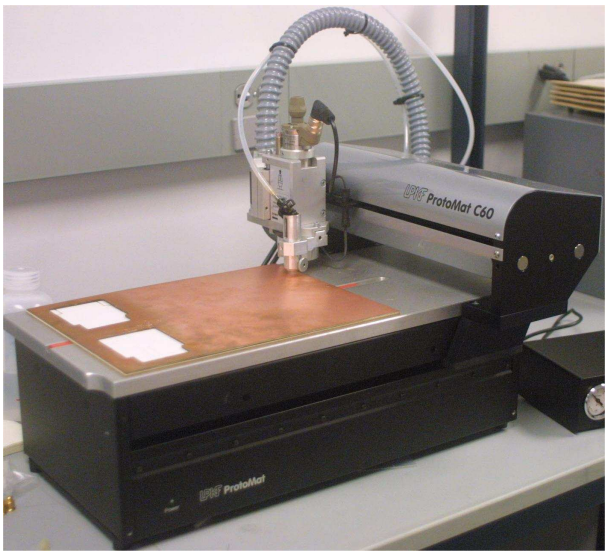

Figura 7: LPKF ProtoMat C60

3.2 Digilent Nexys II

La tarjeta de desarrollo Nexys II comercializada por Digilent es una completa plataforma de desarrollo de circuitos basada la FPGA del fabricante Xilinx modelo *Spartan-3E XC3S500E ó XC3S1200E* con encapsulado FG320. Están dotadas respectivamente de 500.000 y 1.200.000 puertas lógicas equivalentes.

Dispone de numerosos periféricos como: 8 diodos LED, 4 pulsadores, 8 interruptores, un puerto PS/2, un puerto VGA, un puerto serie, un display compuesto de 4 LED´s de 7 segmentos y 4 puertos de expansión mediante conectores PMOD (ver Figura 8). Todas las señales accesibles al usuario están protegidas en caso de cortocircuitado.

Su reloj es de 50MHz pero posee un zócalo para poder albergar un reloj de frecuencia diferente. Lleva incorporadas memorias RAM (16 MB) y ROM.

La Nexys II es compatible con todas las versiones del paquede de herramientas Xilinx ISE. Sus numerosas posibilidades junto con su elevada relación calidad/precio nos llevó a elegirla como dispositivo de control.

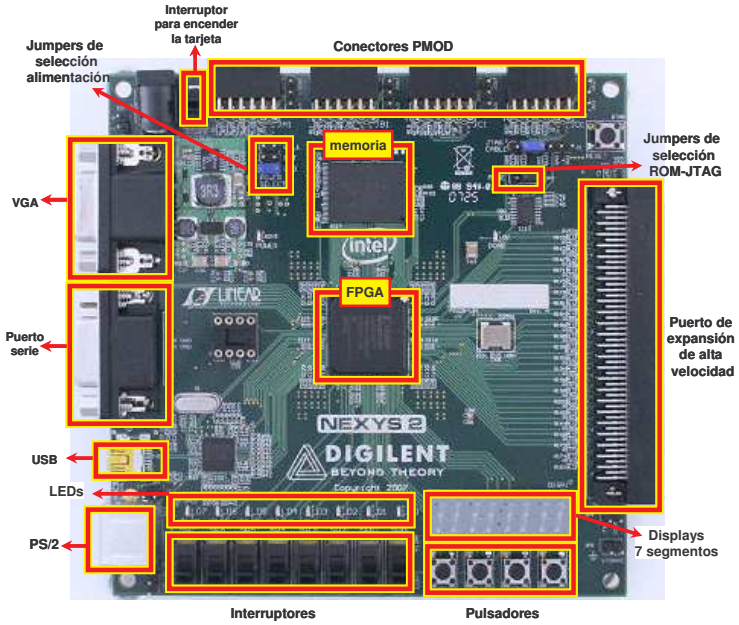

Figura 8: Localización de componentes en Digilent Nexys II

3.3 Prototipo desarrollado de Fresadora de tres ejes

Como ya se ha comentado, debido al alto coste que supone la compra de una Fresadora, como parte del PFC se ha fabricado un prototipo que nos permita validar el control desarrollado.

Aunque ha supuesto un esfuerzo añadido al proyecto, no se ha considerado oportuno profundizar mucho en la construcción de la misma ya que se quiere focalizar el proyecto hacia la parte de diseño e implementación del Control Numérico.

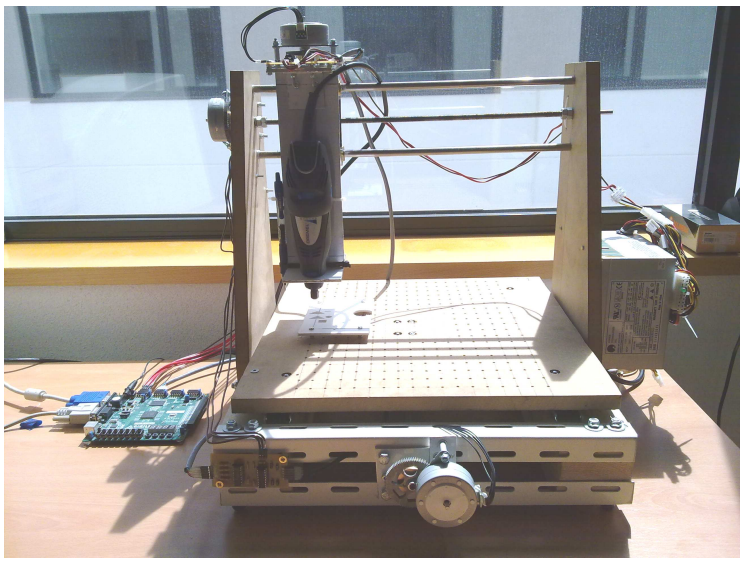

Figura 9: Prototipo de Fresadora de 3 ejes fabricado

El prototipo obtenido que se muestra en la Figura 9 posee las siguientes especificaciones técnicas:

- Tamaño de la mesa: 384 x 400 mm (útiles 354 x 400 mm).
- Motores (X - Y - Z): 3 x PM55L-048.
- Mandril/Fresadora: Dremel 300.
- Resolución: hasta 0,00434 mm.
- Velocidad max.: 26,4 mm/min.

- Transmisión (X - Y - Z): vara de acero roscada M8 de paso 1,25 mm. Dispone de sistema propio *antibacklash* y reductora al piñón de ataque del motor de relación $T = 96/16 = 6$.
- Guías lineales ejes X - Z: metálicas de 9,5 mm con casquillo de bronce autolubricado.
- Guía lineal eje Y: dos patines con rodamientos de bolas de apoyo lateral.

Un aspecto importante a señalar es que el 40% de los materiales que componen el prototipo son reciclados. En la Figura 9 se aprecia el prototipo acabado.

3.4 Motores paso a paso

Para el movimiento de los husos[1] de nuestro prototipo se requieren unos motores tipo *Paso-a-Paso*. Estos motores suelen ir equipados en mecanismos de precisión en los que se requiere el control del giro en alguno de sus elementos como pueden ser impresoras, maquinaria de oftalmología para rotación de lentes, microscopios electrónicos, etc.

Las características principales de este tipo de motores son las siguientes:

- La posibilidad de estacionamiento en determinados puntos de su giro dependiendo de la amplitud del paso, es decir, su resolución.
- El aplique de par en situación estacionaria.
- Alta precisión en el posicionamiento con un error que oscila entre un 3 y un 5% de la longitud del paso y el cual no es acumulable entre un paso y el siguiente.
- Posee numerosas ventajas sobre los motores convencionales: los únicos elementos de rozamiento son los rodamientos (no posee escobillas), el coste del control es menor, se le pueden aplicar muy bajas velocidades de giro sin pérdida del par,

[1] Parte móvil de la transmisión que transforma la rotación de los motores en movimiento longitudinal. En nuestro prototipo son varas roscadas.

poseen una mayor amplitud de velocidades de giro efectivas y poseen una mejor respuesta a las señales que se le envíen.

De entre los diferentes tipos de motores paso a paso los empleados en el proyecto son del tipo unipolar de imanes permanentes, en concreto el mostrado en la Figura 10 que es un NMB modelo PM55L-048[16]. A continuación se expone un pequeño resumen del funcionamiento de este tipo de motores.

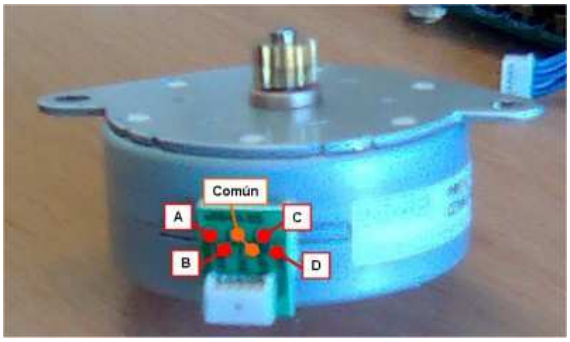

Figura 10: Motor Paso-a-Paso NMB PM55L-048 empleado en nuestro prototipo

El motor paso a paso unipolar está compuesto básicamente por más de un bobinado, en nuestro caso son cuatro, que se encuentran distribuidos en el estator, como se muestra en la Figura 11. El rotor en este caso difiere del de la Figura 11 puesto que posee un cuerpo formado por varios imanes permanentes cuya polaridad se alterna de uno en uno.

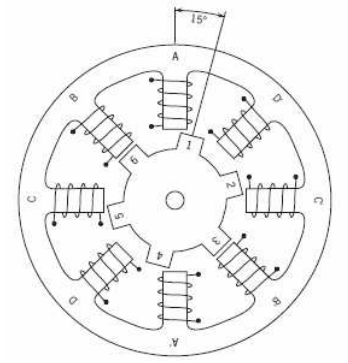

Figura 11: Sección de motor Paso-a-Paso de reluctancia variable

El funcionamiento del mismo es bastante sencillo, por medio del establecimiento de la corriente en determinadas bobinas, el cuerpo del estator se magnetiza. El movimiento sucederá cuando los polos del rotor sean atraídos por el cuerpo magnetizado del estator. El rotor se posicionará en función de la atracción al más cercano de los polos de igual signo. En el caso de la figura será por la proximidad al diente más cercano del rotor.

Existen distintos modos de funcionamiento para los motores paso a paso:

- De paso completo con una fase: se emplean los pasos completos pero mediante la polarización de una bobina en cada paso. Esto conlleva una pérdida del 50% del par del motor y por dicha pérdida de par un giro más suavizado del rotor (los pasos son menos bruscos) sobre todo a bajas velocidades.
- De paso completo: en esta modalidad se emplean pasos completos pero se diferencia de la anterior en que siempre estarán polarizadas dos bobinas. En este modo se emplea el 100% del par del motor.
- De medio paso: en este caso las variaciones de movimiento tienen una resolución de medio paso. Esto se consigue polarizando alternativamente 1 y 2 bobinas. No se obtiene el total del par disponible pero si que se consigue una resolución mayor (el número de pasos se duplica).
- De micropasos: mediante variaciones constantes de las corrientes del motor.

Para nuestra fresadora hemos optado por el modo de paso completo. El motivo principal es el poder desarrollar el mayor par de los motores. No consideramos las opciones de medio paso ni la de micropasos porque la resolución obtenida mediante las reductoras es más que suficiente, el par sería mucho menor ya que en estos modos se llega a tener en determinados micropasos una única bobina activa y el control de los motores pasaría a ser mucho más complejo.

Tabla 1: Secuencia de entradas para los motores Paso-a-Paso según el fabricante NMB

CW ↓	COLOR / STEP	BLACK	ORANGE	BROWN	YELLOW	↑ CCW
	1	−	+	+	−	
	2	−	−	+	+	
	3	+	−	−	+	
	4	+	+	−	−	

Siguiendo las especificaciones del fabricante en la Tabla 1 se muestra la secuencia a seguir para el funcionamiento de los motores en modo de paso completo.

Nuestros motores poseen cuatro cables correspondientes a la masa de cada una de las bobinas que lo componen. A su vez, disponen de un quinto cable de conexión común a todas las bobinas (Vcc). El esquema eléctrico del mismo se muestra en la Figura 12.

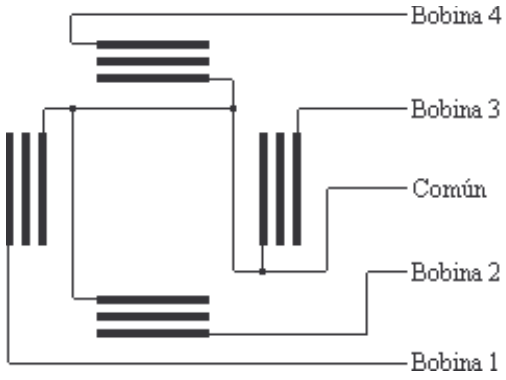

Figura 12: Esquema electrónico de motor Paso-a-Paso

Según el fabricante de alguno de los motores disponibles pudimos observar montajes típicos para el control de los mismos. Los más sencillos y empleados consisten en controlar mediante transistores (típicamente transistores Darlington) las corrientes a cada una de las bobinas.

Los motores paso a paso empleados en la máquina trabajan a un voltaje mayor que el de la FPGA. Por esta diferencia de voltaje ha sido necesario un aislamiento de la electrónica de los motores con respecto a nuestra Nexys II con el objetivo de preservarla de posibles picos de tensión que pudiesen dañarla. Los dispositivos que se han

empleado con este fin son los optoacopladores. En el apartado 4.2 se presenta el diseño implementado para el control de los motores.

3.5 Metodología de diseño con FPGAs

Para llevar a cabo el diseño de los circuitos digitales implementados sobre la FPGA se ha empleado el paquete informático anteriormente mencionado ISE® de XILINX Inc. (Ver 3.1.1).

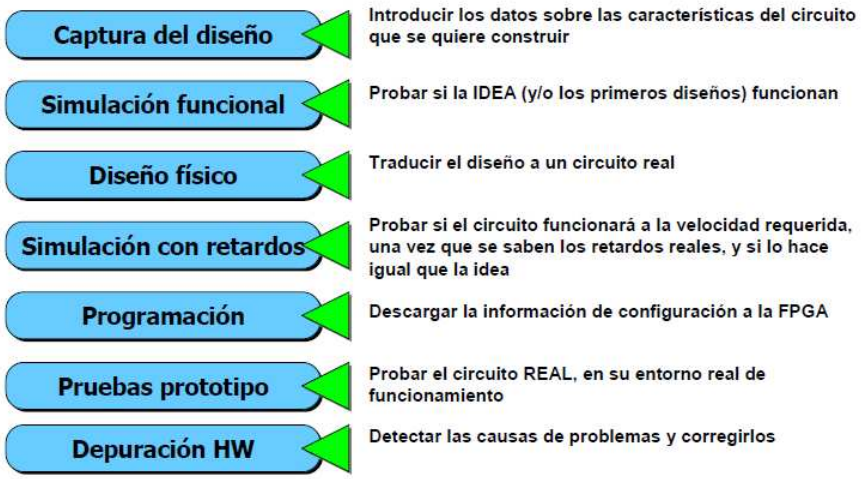

Figura 13: Metodología de diseño con FPGAs

Al ver la Figura 13 se puede observar la metodología empleada en el diseño y desarrollo de los sistemas digitales basados en lógica programable. Para llevarla a cabo se sigue una serie de pautas, las cuales se mencionan a continuación:

a) Captura del diseño: Se introduce el diseño que se quiere simular. Hay varias posibilidades para la captura de los diseños como son: esquemáticos, utilizando lenguajes de descripción de hardware, ecuaciones algebraicas o editores de diagramas de estados (ver Figura 14). En nuestro caso utilizaremos el lenguaje de tipo descriptivo VHDL.

b) Simulación funcional: Se crea un banco de prueba o *testbench* para comprobar el funcionamiento del circuito. Se simula dicho funcionamiento mediante el software de simulación de circuitos digitales ModelSim XE. Se comprueba que el funcionamiento del circuito sea el esperado. Para ello se debe

realizar un análisis de las formas de onda de las señales obtenidas. En caso de obtener unos resultados no esperados, se pasará a revisar el diseño empezando por la especificación de partida para así localizar el fallo, retocando así el esquema y repitiendo el ciclo de diseño.

c) Implementación del diseño en la FPGA: En nuestro caso se va a emplear una FPGA de Xilinx, en concreto el modelo Spartan3E-500. Con la herramienta de Xilinx tenemos cubiertas las etapas que van desde la captura de nuestro diseño hasta la programación del dispositivo de lógica programable. Por otro lado la herramienta tiene automatizadas las etapas comprendidas entre la captura y la programación de la FPGA.

d) Validación: mediante pruebas sobre el prototipo.

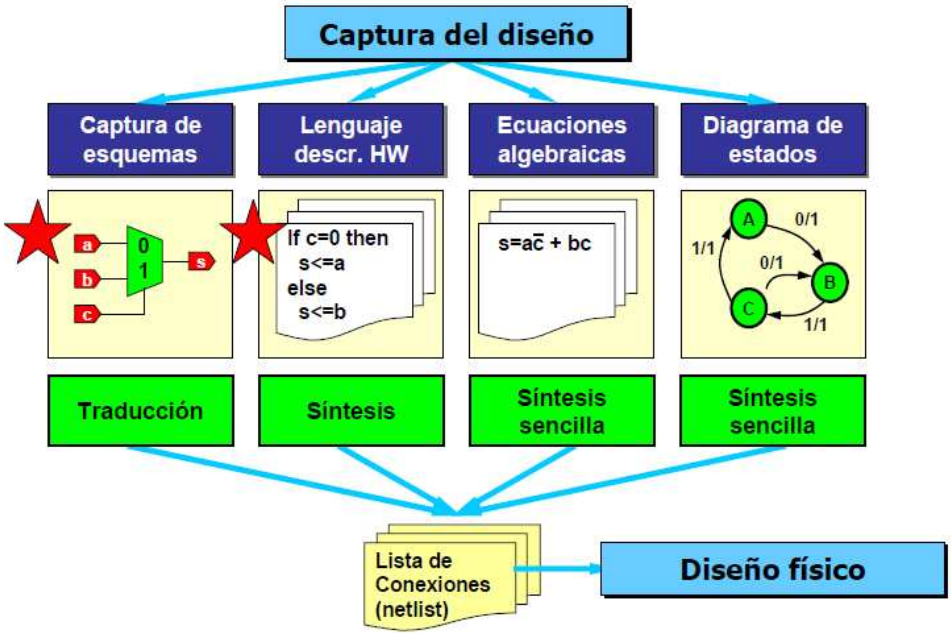

Figura 14: Métodos para la captura de diseños mediante FPGAs

Capítulo 4: Diseño e implementación

Como ya se ha explicado en los capítulos anteriores, el desarrollo del proyecto se ha dividido en dos partes: implementación del control numérico sobre una FPGA y el diseño y trazado de la electrónica de control de la fresadora.

4.1 Implementación del control numérico sobre una FPGA

La implementación sobre la FPGA se ha realizado de una forma modular. Esta técnica se basa en dividir un diseño complejo en otros de menor tamaño. De esta forma se facilita la reutilización de los módulos.

Nuestra implementación dispone de dos modos de funcionamiento o control de la fresadora:

I. **Modo manual**: a través de este modo el usuario fijará el origen desde el que partirá la fresadora.

II. **Modo Automático**: este modo no obedece al control manual. Llevará a cabo de forma automatizada el posicionamiento de la máquina mediante sus señales digitales de control.

Para una mejor comprensión de la totalidad del Control Numérico desarrollado procedemos a explicar su implementación modular haciendo uso de un estudio "*top-down*", es decir, partiendo del módulo de mayor jerarquía hasta llegar a los de menor. Los módulos desarrollados en este apartado son todos aquellos que hemos considerado de relevancia, omitiendo los módulos que corresponden a componentes de un carácter más común como pueden ser los multiplexores, comparadores, etc.

4.1.1 Módulo Principal

Se han diferenciado dos partes en nuestro módulo principal, la automática y la manual. Cada una de ellas asociada a uno de los modos de funcionamiento explicados en los párrafos anteriores.

En nuestro Módulo Principal hemos incluido también dos multiplexores que funcionan como selectores de salida. El primer multiplexor es para las salidas de control de los motores que van directamente a la fresadora. El segundo controla las señales de salida de la interfaz de usuario. Ambos multiplexores se rigen por la entrada de selección de modo llamada *Switch.*

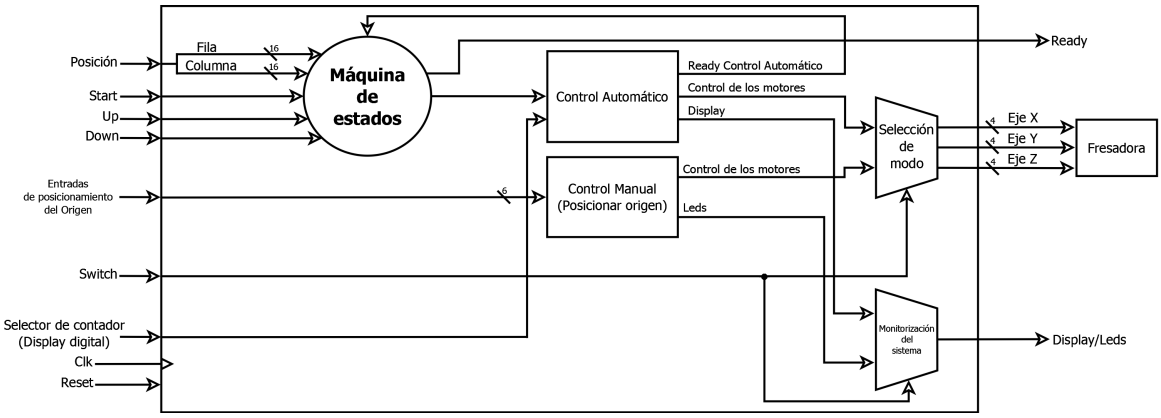

Figura 15: Diagrama de bloques del Módulo Principal

La Figura 15 nos muestra el diagrama de bloques del Módulo Principal. A continuación se explican en detalle cada una de las entradas y salidas para cada modo de funcionamiento y aquellas que son independientes de los mismos.

Para el ***modo automático,*** que incluye la máquina de estados y el bloque denominado "control automático", se dispone de las siguientes entradas y salidas:

1) Posición: en realidad es una entrada doble, de dos señales de 16 bits correspondientes a la posición del eje X (Fila) y del eje Y (Columna) en la que queramos que se posicione nuestra máquina.

2) Up/Down: dos entradas de un bit cada una. Como el Control Numérico es de dos dimensiones estas entradas controlarán la subida y bajada de la herramienta de corte instalada en el eje Z de la fresadora. Tanto el movimiento

de subida como el de bajada son del mismo número de pasos de su eje, fijados por medio de una constante en un nivel inferior de la implementación que se detallará a posteriori.

3) Start: entrada de un bit que ordena el posicionamiento de la máquina.

4) Selector de contador: entrada de dos bits empleada para seleccionar el valor que deseamos que se muestre en el display de la FPGA. Hay cuatro posibles valores: el contador principal de pasos del control automático y los tres contadores correspondientes a cada uno de los ejes (X/Y/Z). De este modo se puede saber en tiempo real que ejes se encuentran en movimiento y en que posición.

5) Ready Control Automático: salida de un bit que informa de que el control automático esta listo para recibir una orden de posicionado. Si la máquina se encuentra en movimiento estará a 0 y en caso contrario se mantendrá un 1 en esta salida.

6) Display: salida para el display digital.

7) Eje X/Eje Y/Eje Z: son salidas de 4 bits para dar las órdenes de movimiento a cada uno de los ejes.

El ***modo manual*** dispondrá de las siguientes entradas y salidas:

1) Entradas de posicionamiento del Origen: el origen de coordenadas será inicialmente determinado por el control manual. Se ha designado para simplificar el modelo una entrada de 6 bits que determina la posición de partida de la fresadora. A cada eje le corresponde un par de bits de esta entrada, los cuales sirven para mover el eje en un sentido u otro.

2) Leds: salida de 4 bits que controla el encendido de cuatro led´s de la FPGA. Sirve de indicador para saber que se esta mandando orden de movimiento a alguno de los ejes. Si no se está moviendo ningún eje indicará el estado en el que se encuentra la salida de 4 bits correspondiente al eje X por defecto.

3) Eje X/Eje Y/Eje Z: idénticas a las del modo automático para poder controlar también los tres ejes.

Las entradas y salidas comunes a ambos modos de funcionamiento son las siguientes:

A) **Switch**: entrada de un bit para la selección del modo de funcionamiento. También se seleccionará con ella la salida de monitorización del módulo.

B) **Clk**: dado que nuestro módulo es síncrono tendrá una entrada que corresponde al reloj de la FPGA. Esta entrada se irá propagando a los módulos de inferior nivel que también sean síncronos puesto que es necesario que se comparta la misma señal de reloj.

C) **Reset**: entrada para resetear el módulo. De idéntica forma a la señal Clk, se propagará al resto de módulos.

Clasificaremos los movimientos que se ordenan a la fresadora en tres tipos:

I. De posicionamiento: son aquellos movimientos en los que la fresa no está actuando. Al efectuarlos se tomará el camino más corto. La posibilidad más compleja será un movimiento diagonal de los ejes X e Y simultáneamente seguidos de otro individual de uno de los dos ejes.

II. De fresado: movimientos con la fresa actuando sobre el material a desgastar. Constan de un único paso en una de las 8 direcciones posibles.

III. De bajada/subida: movimientos de la herramienta de corte para penetrar o salir del material a fresar.

La máquina de estados asociada al módulo principal llevará tanto la gestión de los elementos de memoria requeridos como la ejecución de las órdenes del módulo automático. Resumiendo de una forma sencilla se puede decir que ordenará al módulo automático “movimientos simples” que compongan “movimientos complejos”. Para nosotros un “movimiento simple” será aquel en el que se tenga que mover uno o dos ejes un número de pasos fijo, es decir, una línea, ya sea ésta en diagonal (dos ejes), vertical u horizontal. De este modo nuestra máquina de estados se encargará de descomponer un “movimiento complejo” en dos dimensiones en varios “movimientos simples”. A lo anterior hay que añadir la obtención de la salida *Ready* que indicará que el Control Numérico está listo para llevar a cabo otro movimiento.

Para este módulo es necesaria la gestión de determinados elementos de memoria que almacenen los datos requeridos, como por ejemplo la posición actual de la máquina o la posición a la que queremos ir una vez se haya dado la señal de comienzo. Para una fácil comprensión de la máquina de estados indicaremos algunas de las variables almacenadas sobre los citados elementos de memoria:

►N_Pasos_X: variable que nos indica el número de pasos que tendrá que desplazarse en según el eje X. Se calcula para cada movimiento teniendo en cuenta la posición actual y la posición a la que se quiere ir.
►N_Pasos_Y: idéntica a la anterior pero respecto al eje Y.

Existen más variables gestionadas por la máquina de estados pero que no nos son relevantes a la hora de explicar la máquina de estados cuyo diagrama se puede consultar en la Figura 16.

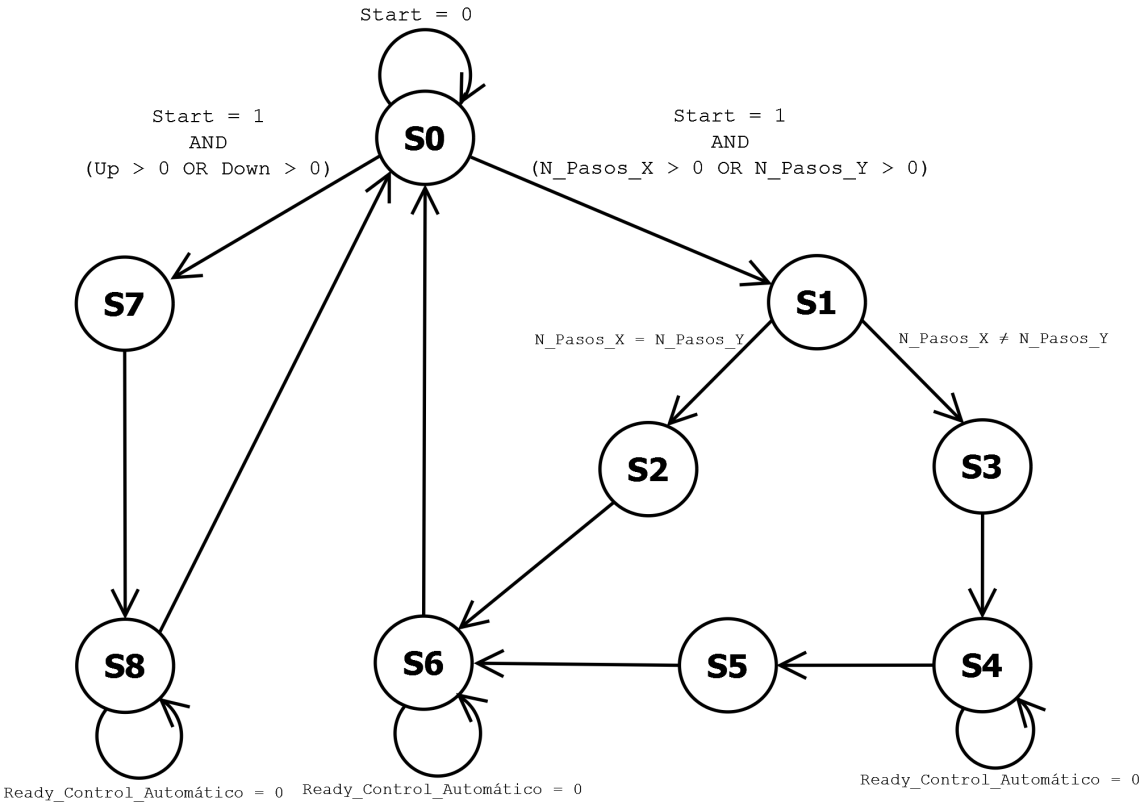

Figura 16: Diagrama para la máquina de estados del Módulo Principal

Para nuestra máquina de estados son esenciales las entradas *Start*, *Ready*_Control_Automático, *UP/DOWN* (si el movimiento es de subida o bajada) y

Posición (para el movimiento de posicionamiento en el plano de dos dimensiones). Las órdenes correspondientes al módulo del Control Automático se detallarán en el apartado 4.1.2.

Ahora explicaremos uno a uno cada estado de la máquina de estados:

- **S0:** es un estado de espera. Mientras la señal Start esté a 0 nos mantendremos en este estado. Se optimizó el cambio de estado añadiendo a la condición que tenía que haber en las variables calculadas un número de pasos a realizar distintos de cero. Sin este cambio se podía ordenar un movimiento a la posición actual con lo que no se producía ninguna orden a la fresadora pero sí se llevaban a cabo todas las transiciones entre estados.

  Hemos aprovechado la espera del estado S0 para realizar los cálculos sobre las variables que almacenarán el número de pasos a dar por la fresadora, las ya mencionadas N_Pasos_X y N_Pasos_Y y se calculan teniendo en cuenta el origen fijado por nosotros (será nuestra posición (0, 0)) y la posición de posicionamiento (compuesta de Fila y Columna).

- **S1:** es un estado de decisión. En él una vez se obtienen los pasos a dar por cada eje se decide entre dos opciones de trazado. La primera opción corresponde a la situación en la que tenemos el mismo número de pasos a dar por ambos ejes y para ello pasamos al estado S2. La segunda corresponde al caso en el que hay un número distinto de pasos a dar para cada eje para lo que iremos a S3.

  En este estado nos hemos visto obligados a realizar una espera inicial. El motivo es el tiempo que se requiere para llevar a cabo los cálculos necesarios para obtener la cantidad de pasos a realizar. La velocidad del reloj de la FPGA hace que para nosotros este retardo sea despreciable (del orden de 20 ciclos de reloj que equivalen a 0,4 μS).

- **S2:** pasaremos en este punto a dar al control automático la orden correspondiente para ejecutar un movimiento del mismo número de pasos para

los ejes X e Y. Una vez ejecutada la orden se transita al estado de espera de fin de movimiento S6.

- **S3:** es el estado más complejo de todos. En él se ha llevado a cabo una solución para simplificar la máquina de estados. Estudiando el funcionamiento de la fresadora entendimos que existen dos situaciones de funcionamiento: la situación en la que se está fresando, es decir, con la herramienta de corte bajada por medio de una orden con la señal de entrada DOWN y la situación en la que la herramienta de corte está "*al aire*", o lo que es lo mismo sin haber profundizado en el material a cortar o habiendo realizado previamente una orden de levantamiento de la herramienta con la señal UP.

De esta manera, con la herramienta "*al aire*" podremos ordenar un movimiento de un número de pasos en teoría ilimitado. En el supuesto de tener la herramienta en posición de corte debemos realizar órdenes teniendo en cuenta el modelo a fresar. Por ello solo se realizarán movimientos que tengan por longitud un paso[2]. En el caso contrario, como con la herramienta "*al aire*" no se tienen restricciones a la hora de posicionarla hemos elegido trazar el camino más corto.

A continuación vamos a explicar cómo trazar la diagonal de un *movimiento complejo*. Partimos de la base en la que siempre que se ordene un movimiento de distinto número de pasos para cada eje se trazará por medio de una diagonal de dos ejes seguida de una recta de un solo eje (camino más corto). En el caso de que se tenga una orden de movimiento de un solo eje se tomará este modo de trazado a través del estado S3. Es factible ya que el movimiento cumplirá la condición → N_Pasos_X ≠ N_Pasos_Y. El movimiento de un solo eje implica que uno de ellos tenga un valor igual a 0 pasos. Como el número de pasos para trazar la diagonal del movimiento compuesto es igual al de menor módulo de los

[2] Nuestra implementación también aceptaría ordenes de fresado de más de un paso, siempre que estas fueran en una misma dirección y sentido. Se planteó la implementación de un modelado vectorial aunque se desestimó ya que para que tuviera sentido la fresadora debería ser al menos igual de rápida que la electrónica. Como es prácticamente imposible se optó por implementar este modelo que además de ser más sencillo es igualmente válido.

ejes, resultará que el número de pasos para nuestra diagonal será 0. De esta forma no se trazará diagonal alguna y después se ordenará un movimiento rectilíneo de un único eje. S3 siempre transita al estado S4.

- **S4:** estado de finalización de espera del movimiento de S3. Tiene una espera mínima programada. Se ha reutilizado todos los recursos empleados en la espera inicial que se realizaba en S1. Con la espera inicial terminada dependerá de la señal de salida Ready_Control_Automático. Si se recibe un 1 pasaremos a S5. En caso contrario permaneceremos en S4.
- **S5:** hemos programado este estado para que lleve a cabo el movimiento rectilíneo resultante de descomponer un *movimiento complejo*. Se ejecuta en él la orden de trazar en el eje al que pertenecen un mayor número de pasos la diferencia de pasos entre el eje con mayor número y el de menor. La diferencia del número de pasos se realiza a la par que las operaciones necesarias en el resto de estados. Se calcula por lo tanto en S0 y S1. Por ello, este estado carece de espera inicial.
- **S6:** estado de espera para el movimiento de S5 y S2. Con la finalización de la espera siempre nos conduce a S0.
- **S7:** realiza los movimientos de bajada y subida. El número de pasos a realizar en ambos movimientos es siempre el mismo y está fijado en una constante. Tampoco posee espera inicial. De aquí siempre transitamos a S8.
- **S8:** estado de espera para el movimiento de S7. Con la finalización de la espera siempre nos conduce a S0.

4.1.2 Módulo para el Control Automático

Se puede definir el Control Automático como la unión de una serie de módulos (diseño estructural). El diagrama de bloques en el que se especifican las entradas y las salidas, así como las señales intermedias, se puede ver en la Figura 17.

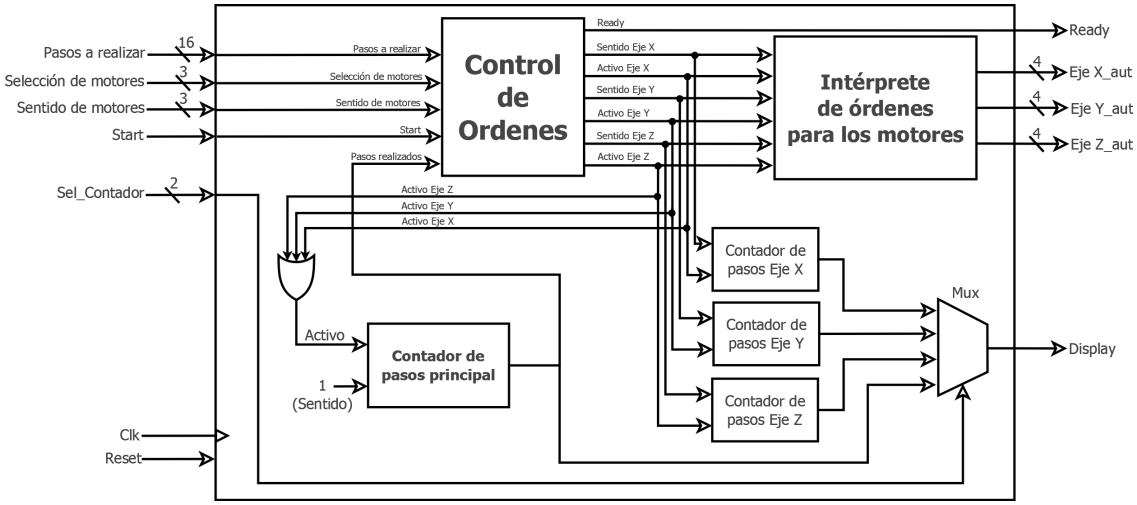

Figura 17: Diagrama de bloques del Módulo de Control Automático

Una vez llegado al punto de la implementación en el que se poseían tanto los contadores de pasos como el *Intérprete de órdenes para los motores* decidimos implementar un módulo que generara las entradas para el intérprete teniendo en cuenta tanto el avance de cada eje como las entradas que recibe el Control Automático. Fue denominado *Control de Órdenes*.

El avance se registra mediante el *Contador de Pasos Principal*. Este contador es de sentido único, es decir, siempre cuenta de forma ascendente y por ello su señal de entrada *Sentido* ha sido fijada por una constante de un bit de valor 1 (el que corresponde al sentido de cuenta ascendente para el módulo *Contador de Pasos*). Así que ya se poseían todos los elementos para poder implementar un módulo de mayor nivel que nos daba abstracción sobre ellos.

También se ha incluido una parte de monitorización, a través del *display* con el que cuenta la placa Nexys II, en tiempo real del número de pasos que haya recorrido cada eje. Para este fin bastó con replicar el módulo del contador principal asignando un

contador a cada eje. El hecho de poseer la misma señal de reloj que el contador principal implica que estén sincronizados con él y para estar asociados a cada eje tuvimos que proporcionarles las salidas pertinentes que generaba el *Control de Órdenes*. Para gestionar qué contador mostraba el display se implemento un multiplexor 4 a 1 que recogía las salidas de los tres contadores de los ejes y la del contador principal.

4.1.3 Módulo de Control de Órdenes

Como se ha indicado anteriormente, el Módulo de Control de Órdenes es el encargado de gestionar las señales de entrada del *Módulo Intérprete* para los motores y esta formado por un comparador de 16 bits, un divisor de frecuencia y una máquina de estados (consultar Figura 18).

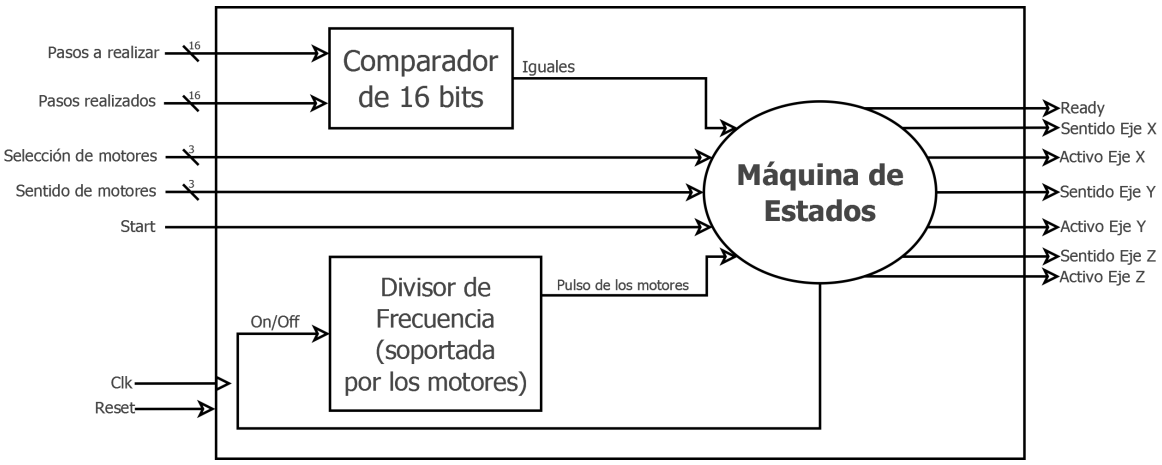

Figura 18: Diagrama de bloques para el Módulo de Control de Órdenes

La gestión de las señales que se producen en este módulo depende directamente del número de pasos realizados en el estado actual de un movimiento. Nuestro Módulo de Control recibirá, por una de sus entradas, en tiempo real este dato proveniente del *Contador Principal*. Se ha implementado en su interior un comparador de 16 bits para generar una señal de un bit que se transmitirá a la máquina de estados. Es la señal de control de parada del movimiento.

Tanto las señales relativas a la selección y sentido de los motores, como la señal de inicio *Start* se transmiten directamente a la máquina de estados.

Es importante destacar que este módulo también va a transmitir el pulso que provoca el cambio de paso de los motores implicados en el movimiento. Para este cometido es necesario un divisor de frecuencia que convierta la señal del reloj en otra señal periódica con una frecuencia que sea soportada por los motores para efectuar el cambio de los pasos. Si se supera esa frecuencia los motores no funcionan correctamente. Es posible variar tambien la velocidad de los mismos teniendo control sobre la frecuencia. Se hizo un estudio previo para ver la velocidad máxima soportada y se fijó como constante.

El funcionamiento de la máquina de estados es muy simple. Está compuesta por tres estados como muestra la Figura 19.

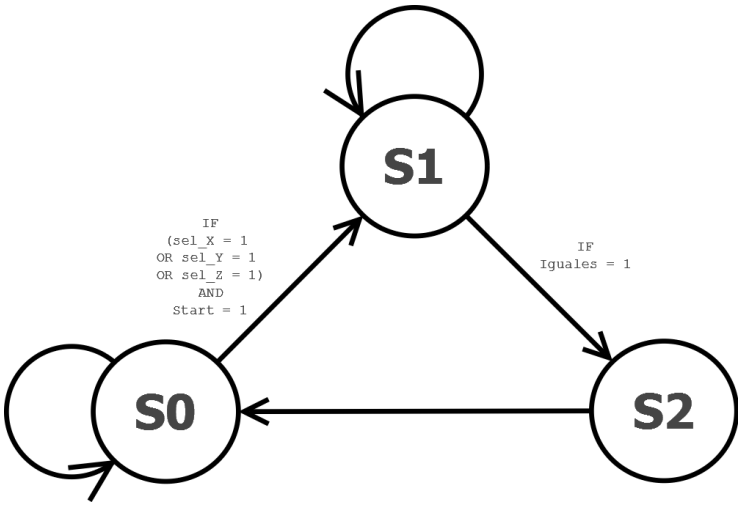

Figura 19: Máquina de estados del Control de Órdenes

A continuación se describe la funcionalidad de la máquina de estados del *Control de Órdenes*:

- **S0**: estado de reposo. Se mantendrá en este estado mientras no reciba ninguna orden. Se considera la llegada de la orden cuando al menos uno de los bits de selección de los motores junto con la señal de inicio *Start* estén a 1. El siguiente estado será S1.
- **S1**: inicialmente habrá una traducción instantánea de las señales de entrada a las salidas que se pasan al intérprete de órdenes. Mientras tanto, la máquina de estados estará pendiente de que se produzca un cambio en la señal

de entrada Iguales. Mientras se mantenga a 0 significará que no se ha alcanzado el número de pasos deseado, por lo que nos mantendremos en S1. Cuando el número de pasos se haya alcanzado Iguales pasará a ser 1 y la máquina de estados transitará a S2.

- **S2**: estado en el que se resetean las señales de salida para el intérprete de órdenes y el módulo divisor de frecuencia. Solo realizará transición a S0.

4.1.4 Módulo del Intérprete de Órdenes de los motores

La funcionalidad de este módulo es incorporar en uno solo los tres *Drivers* de los motores. Recibe las señales de activación y sentido de cada eje y se las distribuye de forma directa (ver Figura 20).

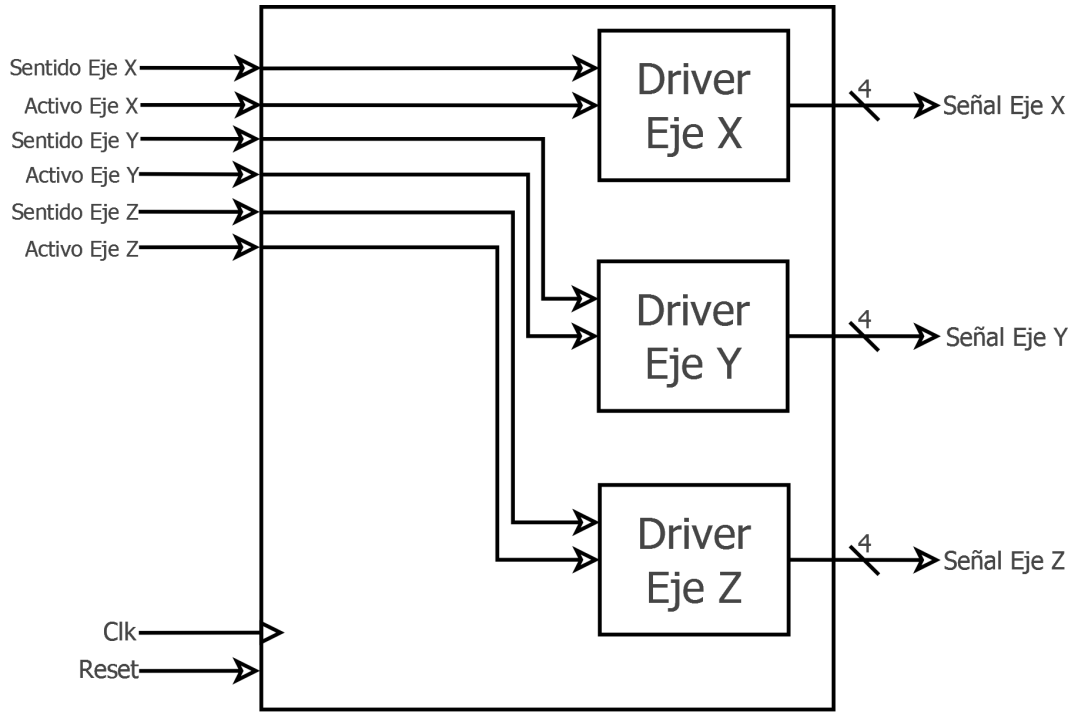

Figura 20: Diagrama de bloques para el Intérprete de Órdenes de los motores

4.1.5 Módulo para el Driver de los motores

Módulo encargado de generar la salida para cada motor. El diagrama de bloques correspondiente es el de la Figura 21. Estos módulos requieren de una máquina de estados para alternar la señal de control de los motores entre las cuatro posibilidades de la secuencia establecida por el fabricante de los motores y que se encuentra especificada en la Tabla 1.

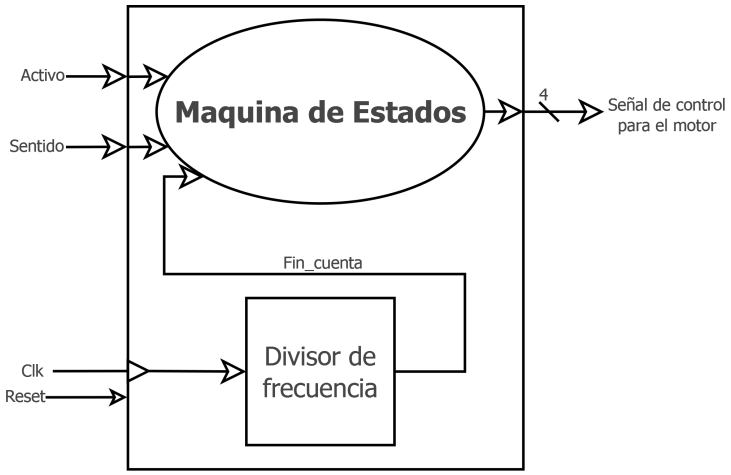

Figura 21: Diagrama de bloques del Driver de los motores

En el momento de la implementación se incluyó un divisor de frecuencia idéntico al visto en el módulo de *Control de Órdenes*. Su objetivo es evitar el incorrecto funcionamiento de los motores. Para ello se fijó la frecuencia máxima de cambio de paso soportada por estos.

La máquina de estados, mostrada en la Figura 22, solo contemplará los cambios en las variables de entrada *Activo* y *Sentido* teniendo como señal de reloj la resultante del Divisor de Frecuencia. Así si se recibe una señal de entrada a una frecuencia mayor el *Driver* la ignorará y no habrá variación alguna en la señal de control del motor.

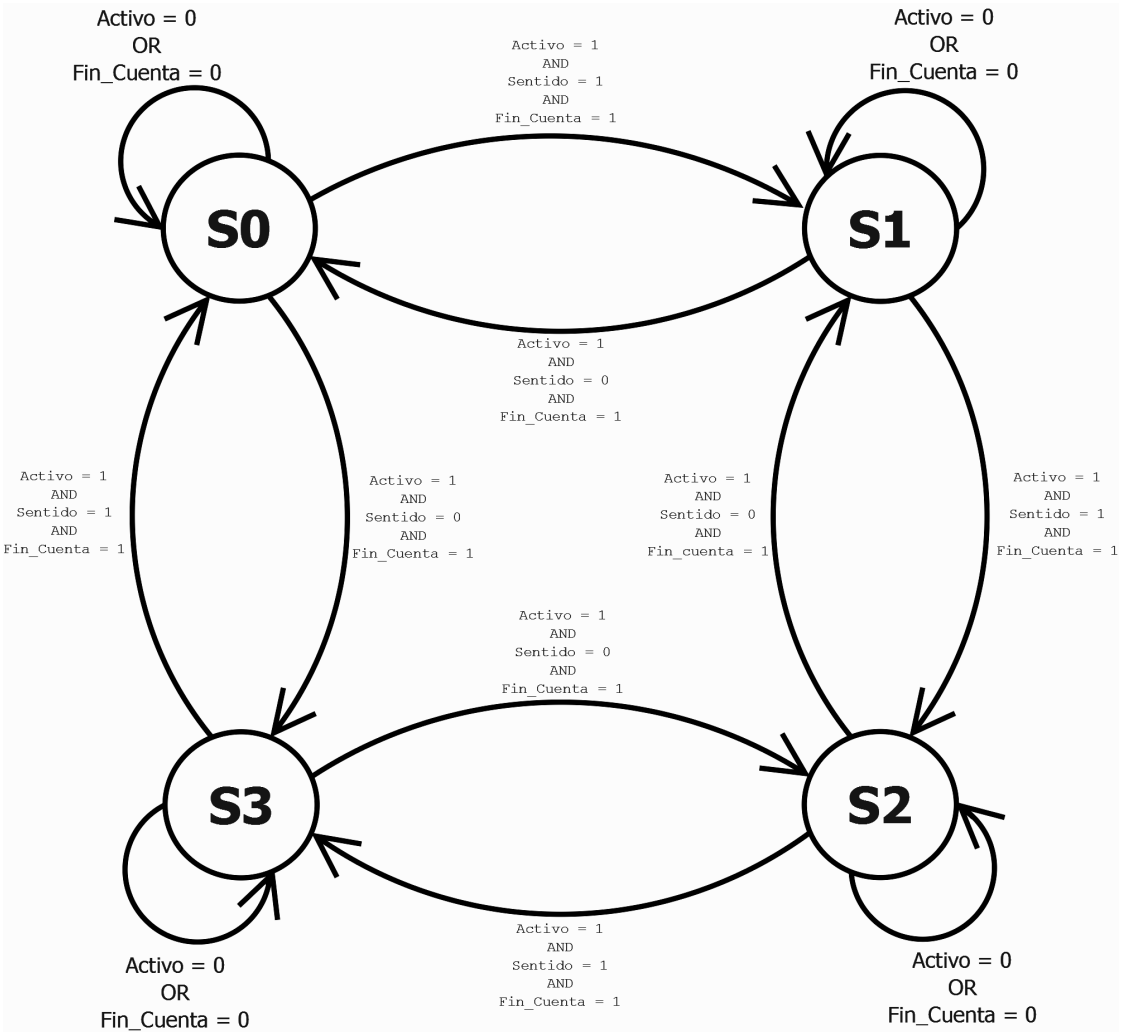

Figura 22: Diagrama para la máquina de estados del Driver de los motores

Todos los estados son iguales. Tienen las mismas transiciones y sus condiciones son idénticas. Cada estado tiene asignada una de las salidas de la secuencia de la Tabla 1 fijandola a la *Señal de control para el motor*.

Las señales *Activo* y *Fin_Cuenta* activarán la transición entre los estados. Bastará con que una de ellas esté a cero para que no se cambie de estado. La señal *Sentido* es la que controla el giro de los motores, o lo que es lo mismo, el sentido de circulación para las transiciones entre estados.

4.1.6 Módulo para el Contador de Pasos de 16 bits

Teniendo en cuenta que todo Control Numérico Computarizado se basa en poder conocer en cada instante de tiempo la posición en la que se encuentra la máquina que

gestiona, necesitábamos un módulo que registrase la posición para cada uno de los ejes. Optamos por un contador de pasos que fuese capaz de contabilizar los pasos y a la vez sacar su valor de forma constante para poder ser consultado en los niveles de implementación superiores.

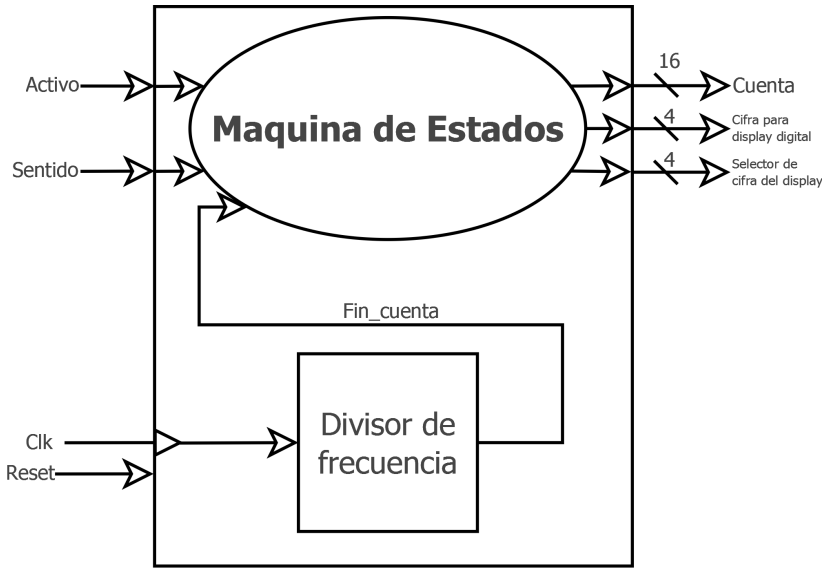

Figura 23: Diagrama de bloques del Contador de pasos

El diagrama de bloques de este modulo es el representado por la Figura 23. Posee una entrada de un bit, denominada *Sentido,* que indica el tipo de cuenta, ascendente o descendente. Su segunda entrada, además de las comunes al resto de módulos *Clk* y *Reset,* es una entrada de un bit que activa la cuenta, de nombre *Activo*.

Se pensó que también se podía dar el caso en el que por aumento de la frecuencia de los pulsos que le llegaran por la entrada *Activo* el contador desbordara el número de pasos respecto a los pasos que en realidad realizaban los motores. Lo cuál nos llevó a introducirle el mismo *Divisor de Frecuencia* que se había implementado para ser incluido en los módulos de los *Drivers* de los motores y en el llamado *Control de Órdenes*.

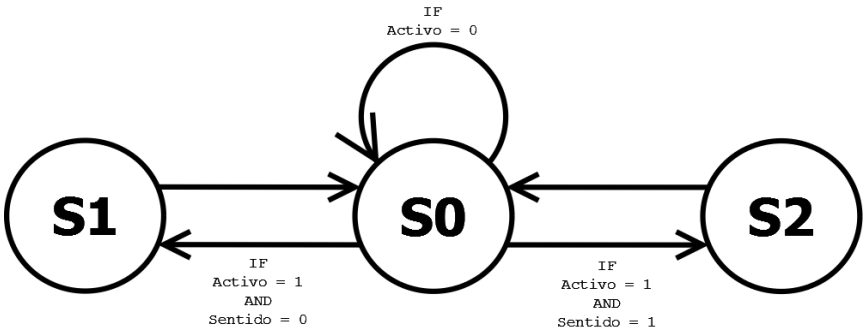

Figura 24: Máquina de estados para el Contador de pasos

La máquin de estados de la Figura 24 gestiona el conteo de pasos y en ella se han definido los tres estados siguientes:

- **S0**: estado de espera, no se transitará a ninguno de los otros dos estados mientras la señal de entrada *Activo* tenga el valor 0. cuando la señal *Activo* sea 1 se realizará el cambio de estado dependiendo del valor de la segunda señal de entrada *Sentido*.
- **S1**: se transitará del estado S0 a S1 cuando se desée que la cuenta se realice de forma descendente. El valor de la señal *Sentido* ligado a la cuenta descendente es el 0. Una vez se sitúa en S1 restará una unidad a la variable que hemos llamado *Cuenta* (que va asociada a su vez a la salida con el mismo nombre) implícita a la máquina de estados.
- **S2**: de forma contraria a lo que ocurre en S1 el cambio de estado desde S0 se produce cuando *Sentido* vale 1. Se ha asignado la cuenta ascendente a este estado por lo que aumentará el valor de *Cuenta* en una unidad.

Nuestro contador de pasos posee una implementación cíclica, es decir, al llegar al final de su resolución total pasará del mayor valor posible (FFFF en hexadecimal) al menor (0000) y viceversa dependiendo del tipo de cuenta que se esté efectuando.

4.2 Monitorización e Interfaz del sistema

4.2.1 Monitorización

Aunque el Control Numérico implementado para nuestro prototipo de fresadora funcionará la mayor parte del tiempo en su modo automático, hemos decidido incluir en la implementación la funcionalidad necesaria que nos permitiese una monitorización del sistema y que dependerá del modo de funcionamiento en el que nos encontremos.

A. Modo manual: en este modo solo nos interesa visualizar el movimiento de los ejes. Para ello se va a mostrar la secuencia de la Tabla 1 a través de cuatro leds de los ocho disponibles en la Nexys II. Como las salidas de control de los motores son de 4 bits pudimos hacer la correspondencia de cada bit de la señal a un led. Si todos se encuentran en modo estacionario los leds están asignados a los cuatro bits de la salida de control del motor del eje X (valor asignado por defecto). Y en caso de que se le ordene movimiento a uno de ellos los leds mostrarán su secuencia de la salida de control. El orden de prioridad de asignación de las tres salidas de control es X-Y-Z siendo la del eje X la más prioritaria y la del eje Z la que menos prioridad posee.

B. Modo automático: para este caso se decidió emplear los *displays* de 7 segmentos. Recordando lo explicado en el apartado 4.1.6 que hace referencia a los contadores de pasos y sabiendo que tenemos implementados en total cuatro, uno para cada eje y uno principal, se va a visualizar el conteo en tiempo real de cada uno de ellos. Las variables de conteo de 16 bits están declaradas como un tipo binario de datos. VHDL incorpora funcionalidad con capacidad para consultar un número concreto de bits de este tipo de datos. Sabiendo esto, y mediante el uso de multiplexores, se crea una variable por cada dígito del *display* compuesta por cuatro bits cada una (ver la Figura 25).

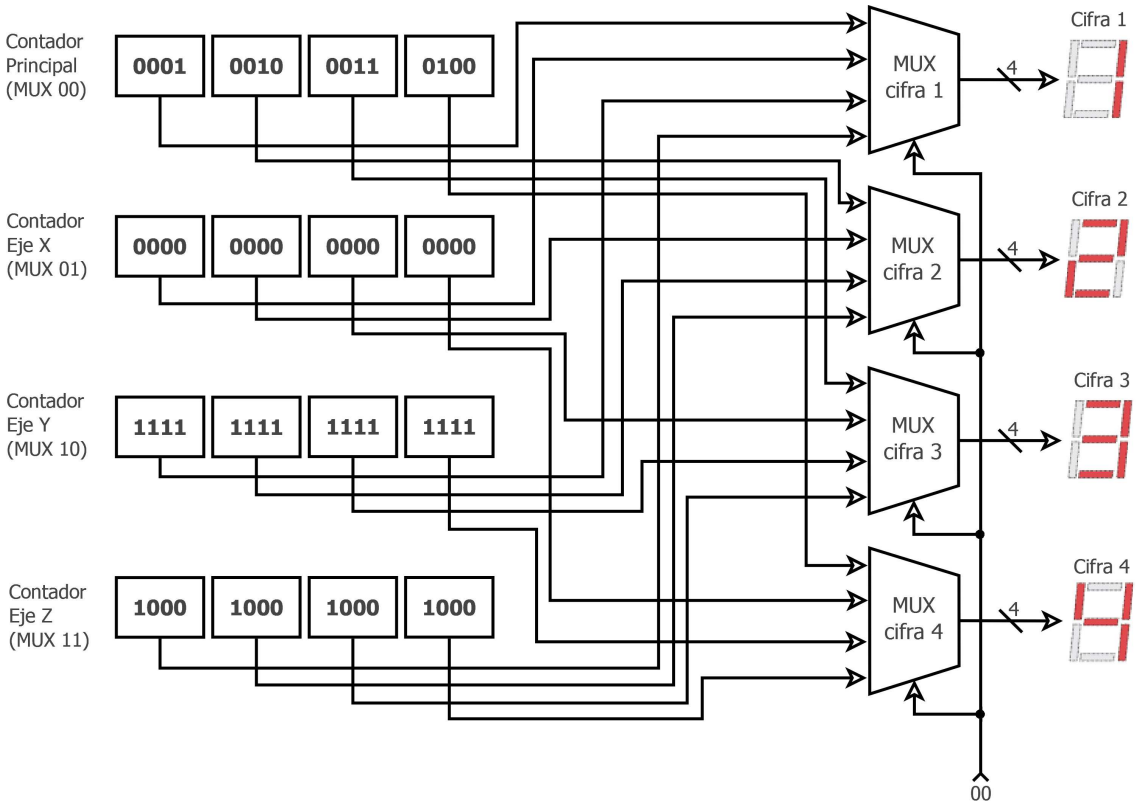

Figura 25: Esquema para la selección de señales del display digital

Por último, se ha incluido una variable más que nos informase de cuando estaba listo el Control Numérico para efectuar órdenes. Para ello asociamos la señal de salida *Ready* a uno de los led disponibles en nuestra Nexys II.

4.2.2 Interfaz

Teniendo clara la parte de la monitorización podemos exponer la parte de la interfaz también para cada modo de funcionamiento:

A. Modo manual: para el control manual de los motores se han asignado un par de interruptores para controlar cada eje, uno por sentido. Por defecto se ha dado preferencia al sentido ascendente en caso de que los dos interruptores de un motor estuvieran accionados simultáneamente.

B. Modo automático: en el modo automático la interfaz se corresponde únicamente con la monitorización del sistema para el conteo de pasos. Hemos asignado dos interruptores para codificar la selección del contador que va a ser monitorizado en el *Display* digital. Dependiendo de la combinación en binario

de dos bits que introduzcamos en estos interruptores se mostrará el conteo de pasos de uno de los cuatro contadores.

En la implementación de nuestro módulo principal existen además dos señales de entrada comunes a ambos modos que había que incluir en la interfaz. La señal de selección de modo, se ha asignado a un interruptor que decide entre uno u otro según su posición. Hay que tener en cuenta que según la implementación si pasamos de modo automático a modo manual se reiniciará el modo automático. La segunda señal que es el *Reset* del sistema implementada mediante un pulsador de los cuatro disponibles en la placa.

4.3 Control de los motores paso a paso

Una vez obtenidos los conocimientos básicos mencionados en el apartado 3.4 se llevó a cabo la realización de diferentes pruebas para controlar los motores a través de la FPGA. Una vez validado, se implementó el diseño realizado en una placa de circuito impreso (PCB).

4.3.1 Circuito de control mediante optoacopladores

Como el control se realiza mediante las salidas digitales de la FPGA, alimentada a 3,3V, el primer elemento que hemos de incluir son optoacopladores, que nos permiten desacoplar la parte digital (control) de la potencia (motores).

Un optoacoplador se compone de un diodo led y un fototransistor. El led funciona de fotoemisor de la señal a emitir y el fototransistor desempeña el papel de fotorreceptor.

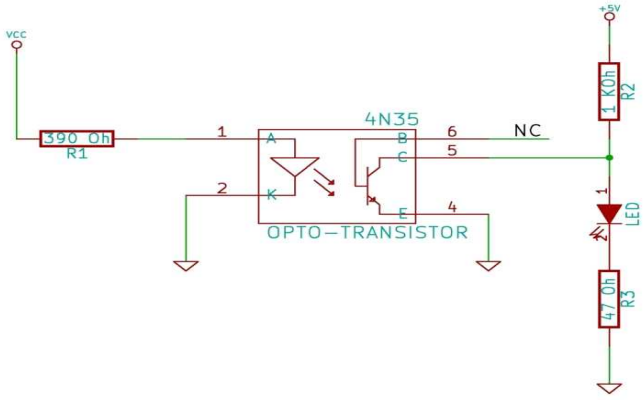

Figura 26: Esquemático del control del encendido de un led con optoaislamieto

Para familiarizarnos con el uso de optoacopladores se realizó un primer montaje, en el que se encendía y apagaba un led, utilizando como señal de control una de las salidad de la FPGA (ver Figura 26).

Posteriormente se llevó a cabo el montaje de cuatro optoacopladores para poder controlar un motor paso a paso (ver Figura 27). Cada uno de los optoacopladores controla el establecimiento de la corriente en una única bobina.

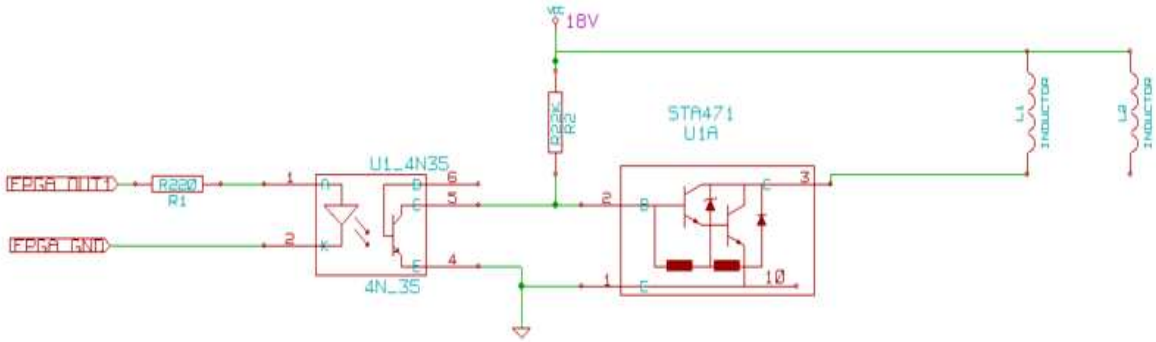

Figura 27: Esquemático para el optocontrol de una bobina mediante transistor Darlington

En el lado de potencia del esquemático (el correspondiente a la bobina) se optó por incrementar el valor de la resistencia R2 (de 1K a 22K), respecto del montaje realizado para el encendido/apagado de un led. El motivo es evitar que circulase demasiada corriente a través del fototransistor.

Las señales de control que enviaba la FPGA eran de 4 bits y debían seguir la secuencia de control correcta del motor. Con este fin, se implementó una máquina de

estados en la que cada por cada paso de la secuencia se tiene un estado. El movimiento del motor se controla por medio de dos pulsadores de la FPGA. Cada uno de estos pulsadores hará un cambio de estado en la correspondiente secuencia, según se desee provocar el giro en un sentido u otro. Su máquina de estados está detallada en la Figura 22 del apartado 4.1.5. En la Figura 28 se puede observar el montaje completo sobre un motor Mitsuni M42SP-5B[17].

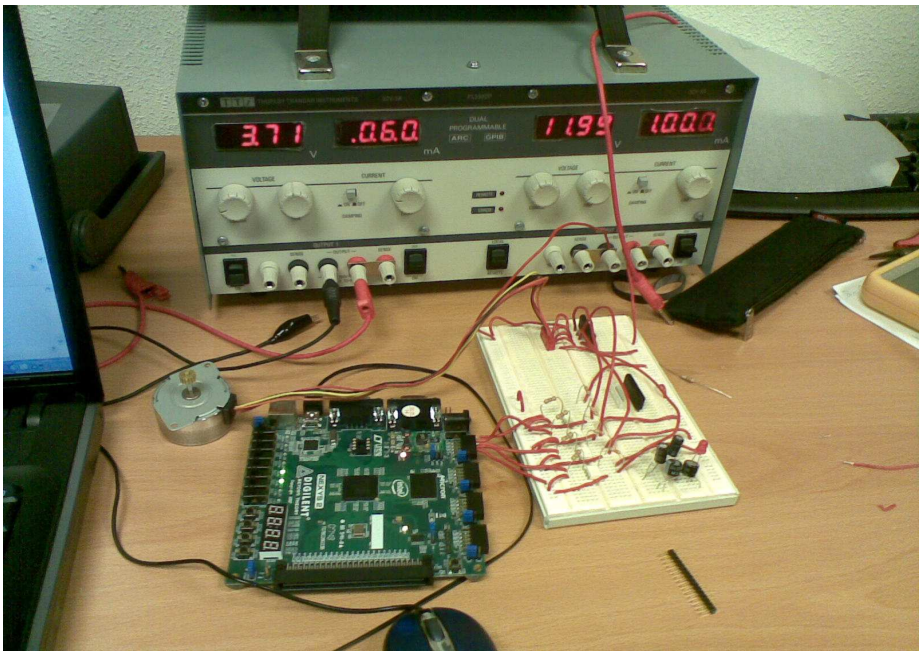

Figura 28: Imagen de prueba de control con optoaislamiento de un motor paso a paso Mitsumi M42SP-5B a través de la Nexys II

4.4 Electrónica de control

Con las pruebas pertinentes realizadas estimamos necesario el diseñar y posteriormente materializar circuitos dedicados que llevasen a cabo el control del motor asignado a cada uno de los ejes de la máquina. Se consideró que lo mejor era realizar un circuito impreso o PCB con los componentes necesarios.

Cabían dos posibilidades de implementación del circuito:

- Aunar el control de los tres motores en un único circuito impreso.
- O realizar tres circuitos independientes.

Se optó por crear tres circuitos independientes. Las ventajas que esta opción nos ofrecía eran:

1. En caso de fallo o rotura del control de un eje no habría que repetir el circuito para los otros dos.
2. El modelo del circuito se simplificaría.
3. Por el hecho de ser un circuito más simple resultaría también más compacto. Este aspecto beneficiaba el encastrado de la circuitería en la máquina.
4. El cableado entre los motores y su circuito de control correspondiente sería mucho más corto al situarse lo más próximos posible.

4.4.1 Diseño de los circuitos de control

Para el diseño de estos elementos fue necesario el software específico y de libre distribución KiCad. Como ya se ha dicho en el capítulo 3.1 este sofware nos permite diseñar nuestras propias PCB (*Printed Circuit Board*), monocapa o de doble capa, y obtener ficheros de tipo gerber para poder imprimirla en la microfresadora que posee el Departamento de Tecnología Electrónica.

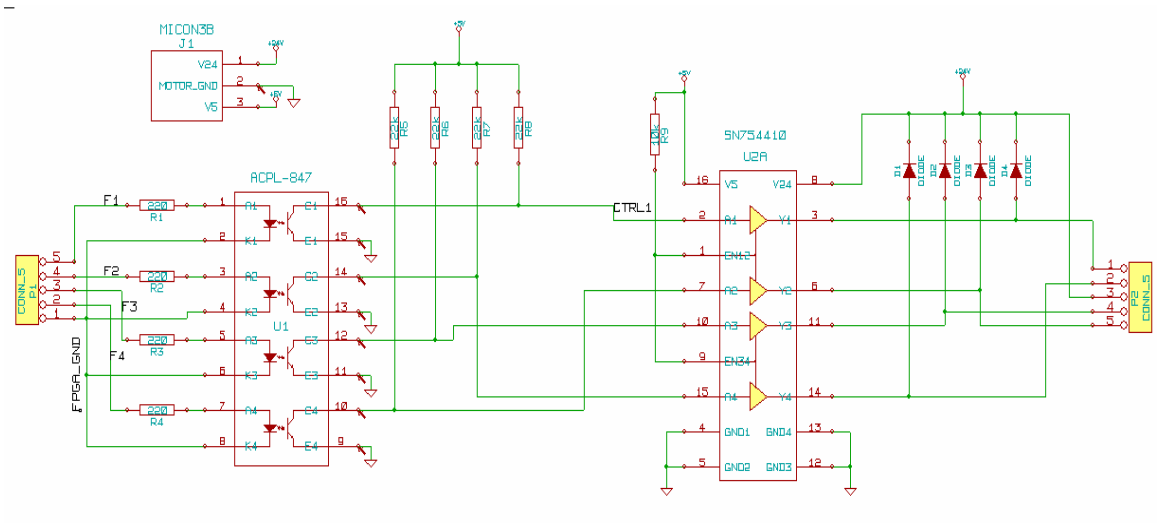

Figura 29: Esquemático de la PCB de control del motor NMB PM55L-048

Inicialmente con la herramienta de creación de esquemáticos EESquema se obtiene el esquemático de nuestra circuito de control (ver Figura 29). En él se incluyen los cuatro optoacopladores, por medio del circuito integrado ACPL-847[18] y los transistores con un SN754410[19].

En el siguiente paso asociamos los componentes físicos a los lógicos del esquemático. Pasamos después a la herramienta de diseño de PCB´s llamada Pcbnew, del paquete de KiCad, para componer el *Layout* de la PCB, o lo que es lo mismo, las pistas, disposición de componentes y conexionado de nuestra placa de circuito impreso. La versión definitiva de la PCB en Pcbnew se ve en la Figura 30.

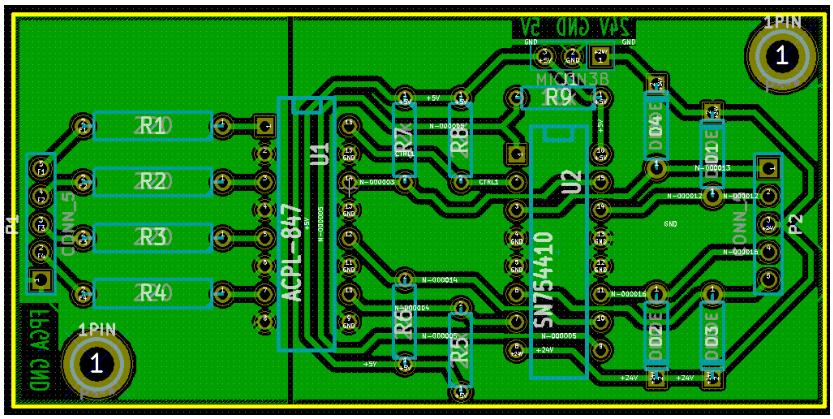

Figura 30: PCB definitiva mostrada por Pcbnew

Para facilitar una visualización física de nuestro circuito, el software posee una herramienta para levantar en 3D el diseño. La Figura 31 muestra una captura de pantalla de dicha herramienta.

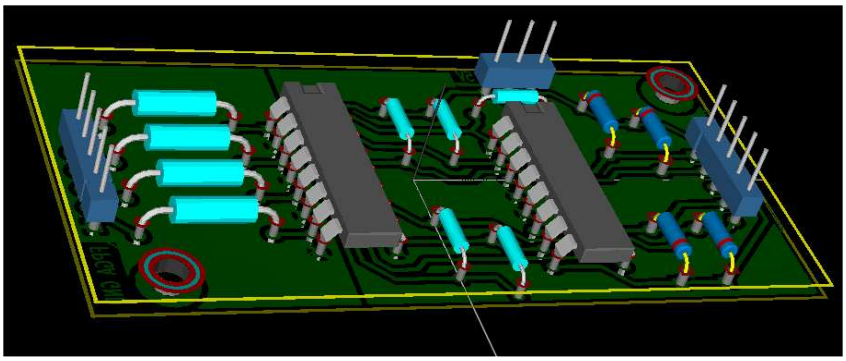

Figura 31: Imagen 3D del diseño de la PCB de control de los motores

4.4.2 Fabricación de los circuitos de control

En la Figura 32 se puede apreciar el diseño definitivo de los circuitos con los componentes ya soldados.

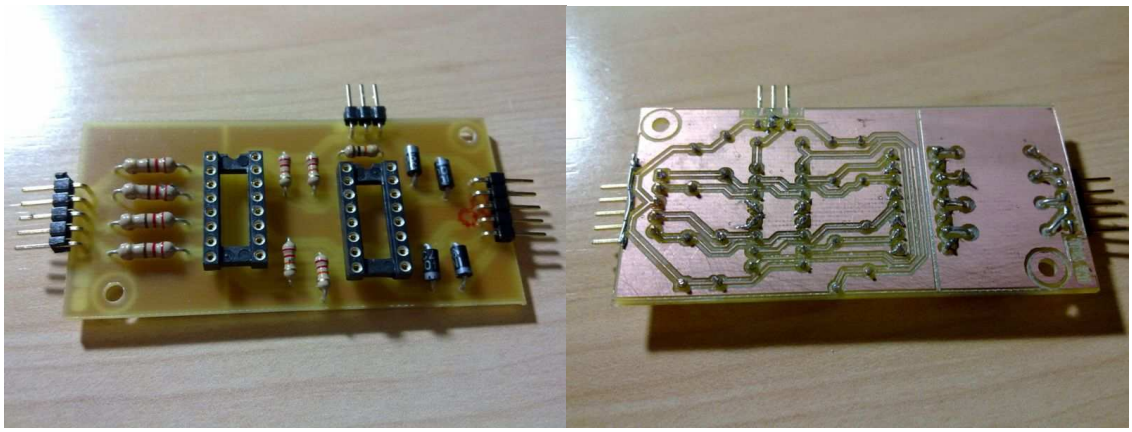

Figura 32: Caras anterior y posterior de la PCB de control de los motores terminada

Cabe destacar que entre la impresión de la primera y la última PCB se hicieron mejoras en el diseño, siendo las más significativas el incremento del grosor de determinadas pistas del lado de potencia y la inclusión de los taladros de fijación a la fresadora (situados en las esquinas superior derecha e inferior izquierda).

En lugar de soldar los integrados a la PCB se decidió incorporar zócalos de 16 pines. Con ellos, en caso de fallo o de resultar dañados los integrados, se puede llevar a cabo una rápida sustitución de los mismos, basta con extraerlos y colocar uno nuevo en su posición correcta.

Capítulo 5: Resultados y Pruebas

En este capítulo vamos a describir los hitos, que se han ido cumpliendo para conseguir los objetivos planteados en este proyecto. El primero de ellos ha sido la construcción del prototipo de fresadora, tanto la parte mecánica como la de la electrónica de control de sus motores. Con el prototipo operativo se implementó un módulo con la funcionalidad única propia del control manual de la fresadora. A continuación se diseñó el módulo relativo al modo automático de control y se fusionó en un módulo de mayor jerarquía junto con el de control manual. Y por último, se llevó a cabo la integración de nuestra implementación con la desarrollada en el proyecto fin de carrera realizado por Héctor Gálvez Barrios para interpretar modelos en dos dimensiones partiendo de una imagen.

En los siguientes apartados se describen en detalle las pruebas que se realizaron y que nos han permitido validar el correcto funcionamiento de lo desarrollado en el proyecto.

5.1 Prototipo de Fresadora y Electrónica de control

Para probar el correcto funcionamiento del prototipo únicamente se llevaron a cabo pruebas mecánicas. Estas consistieron en el accionamiento de las distintas partes mecánicas que la componen para verificar su viabilidad de trabajo.

Hubo un par de modificaciones destacables en el diseño del prototipo debidas al mal funcionamiento de determinadas partes mecánicas. Haremos una breve explicación de las mismas:

1. Soportes para los casquillos de las guías metálicas de los ejes X e Y: el material empleado para estos soportes en un principio fue la madera. Con la realización de la pertinente prueba mecánica de movimiento de los mismos sobre las guías se observó que la madera podía incluso llegar a quebrar, liberando así los casquillos y derivando en una serie de holguras sobre los carros en los que van insertados los soportes. Fueron sustituidos por otros metálicos que solventaron nuestros problemas. Los soportes metálicos de latón son cierres de fontanería para tuberías torneados para albergar los casquillos de cobre.

Figura 33: A la izquierda los soportes de madera para los casquillos de cobre de las guías y a la derecha las versiones metálicas en latón

2. Portacoronas de las transmisiones de los ejes: inicialmente fabricados en teflón se observó en su funcionamiento que debido a las flexibilidad del material no eran aptos pues no cumplían la función de anclaje a la que estaban destinados. En los momentos de mayor esfuerzo mecánico flexaban y, por lo tanto, patinaban en su alojamiento. Por ello tuvieron que repetirse en un material metálico que al no poseer flexibilidad no se producía perdida de agarre (Figura 34).

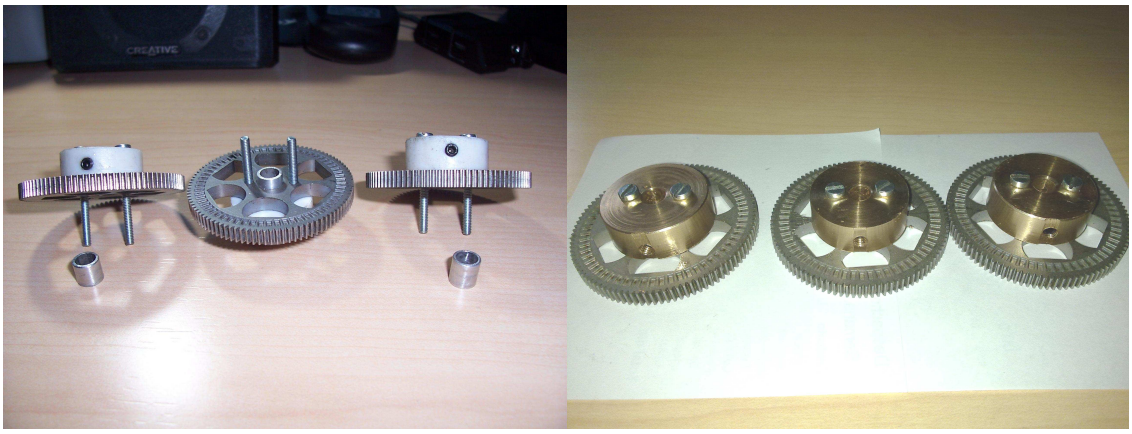

Figura 34: Portacoronas de teflón montados sobre las coronas transmisión a la izquierda. A la derecha las versiones metálicas de bronce torneado

En el caso de la electrónica de control ya se ha explicado en detalle tanto las pruebas realizadas como la implementación de los circutos en PCB (ver apartados 4.3 y 4.4).

5.2 Módulo de Control Manual

Las pruebas referentes al Módulo de Control Manual fueron las más sencillas. Simplemente se probó uno a uno cada eje, verificando que todos trabajasen en ambos sentidos. Adicionalmente se probó el funcionamiento de varios a la vez, haciendo todas las combinaciones hasta probar los tres ejes de forma simultanea.

En este punto del proyecto todavía no se había integrado una fuente de alimentación en la fresadora, siendo suministrada la energía por una fuente del laboratorio. Teniendo el Módulo de Control Manual operativo pudimos llevar a cabo las pruebas de consumo de los motores para hacer viable la inclusión de una fuente de alimentación propia para la fresadora. Se montó una fuente de alimentación de PC colocada en un lateral de la misma como se puede observar en la Figura 35.

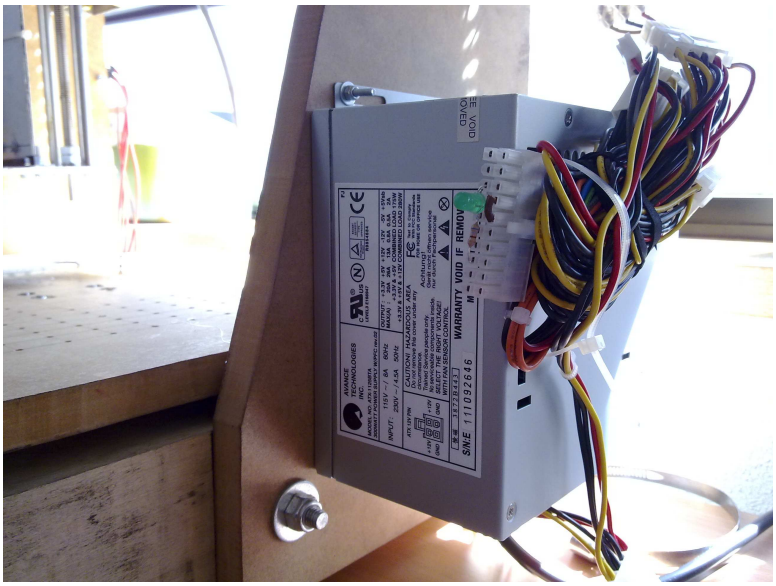

Figura 35: Fuente de alimentación de PC incorporada a un lateral de la fresadora

5.3 Módulo de Control Automático

Con el Módulo de Control Automático implementado y simulado en ModelSim se procedió a efectuar la misma prueba pero sobre la Nexys II. Para ello creamos un nuevo proyecto con Xilinx en el que incluimos el módulo a probar. Como señal de entrada externa se asignó un pulsador a la señal *Start* del Control Automático. En este proyecto generamos las órdenes de modo automático con una serie de señales que iban tomando los valores en coordenadas de cada uno de los vértices de un cuadrado. Estas variables se iban actualizando en función de la señal de salida *Ready* del Control Automático por medio de una máquina de estados.

Primero se realizó una serie de pruebas para corroborar que cada eje realizaba el número de pasos correctamente mediante la monitorización. En este punto se observó un fallo correspondiente al cálculo de las coordenadas. Este error provocaba que el control numérico ordenara un movimiento pero con coordenadas de valor (0, 0). Las consecuencias de este error eran dos: o no se producía movimiento alguno porque su posición en ese instante era igual a la de la orden o si se forzaba el movimiento (con una modificación sobre el módulo) éste no se detenía hasta dar la vuelta completa a los contadores.

Para solventarlo se incluyeron esperas en determinados estados de la máquina del control automático. Estas esperas se explican en el apartado 4.1.1 y básicamente sirven para permitir al módulo el cálculo de pasos a realizar antes de llevar a cabo el cambio de estado en la máquina de estados que ordenaba el movimiento. Sin esta espera se hacía un cambio inmediato y por ello el cálculo quedaba a cero.

Solucionado el problema del cálculo de pasos y observando mediante la monitorización que se trazaba correctamente el cuadrado, se repitió la prueba con la máquina operativa. En esta ocasión se tomaron precauciones para el trazado, poniendo en lugar de la herramienta de corte un bolígrafo que trazaba el cuadrado sobre papel.

5.4 Integración con el Módulo de Generación de Coordenadas para modelos en dos dimensiones

La integración del Control Numérico junto con el Módulo de Generación de Coordenadas (los llamaremos CN y MGC para abreviar), desarrollado en el proyecto fin de carrera realizado por Héctor Gálvez Barrios, llevó implícitas una serie de medidas para compatibilizar ambas implementaciones. Fue necesario un estudio de las características de ambos sistemas para poder adecuar las señales que comparten ambos módulos.

Dado que nuestro sistema es parametrizable esto no supuso un gran esfuerzo. Podemos recibir coordenadas con un valor máximo de resolución marcado por los 16 bits que poseen las entradas. Significa que podemos recorrer por cada eje 65.536 pasos (2^{16}). Como la imagen que envía el otro módulo tenía una resolución máxima de 400 x 400 pixels bastaba con señales de 9 bits para cubrir la imagen por completo. Ahora bien, la longitud de paso de la máquina es muy pequeña (0,00434 mm ver especificaciones técnicas en el apartado 3.3) con lo que si hacíamos un paso directo de esta señal de 9 bits a los menos significativos de nuestras señales de entrada el tamaño de lo representado por la fresadora sería demasiado pequeño. Por este motivo se decidió realizar el paso directo pero a los 9 bits más significativos.

La mayor dificultad que encontramos en la integración de los dos proyectos fue el de la sincronización, agravado por el hecho de que los resultados obtenidos en la simulación eran satisfactorios, lo que dificultó su localización. El problema apareció porque la máquina de estados del MGC era más rápida que la de nuestro CN, puesto que no tenía la necesidad de esperar para hacer sus cambios de estado y por lo tanto cambiaba sus señales de salida antes de que los registros de almacenamiento de la posición del CN hubiesen almacenado la posición de destino. Por lo tanto, no se llevaba a cabo movimiento alguno.

Al final se determinó que había que establecer una señal de sincronización entre ambos módulos y se aprovecho la señal *Ready* del CN. Esta señal no se modificaba hasta que no se realizaba el cambio de estado desde el estado inicial de espera. Como nuestro módulo ya tenía implementada una espera inicial en el estado de espera para almacenar los valores de entrada, bastó con que el MGC también esperara al cambio que se producía en la señal *Ready* para realizar su cambio de estado. Después también debía esperar a que el CN volviera a estar disponible para efectuar otra orden de movimiento, por lo que esperaba a que la señal *Ready* lo indicase de nuevo.

Solventados estos problemas la integración fue satisfactoria y se llevaron a cabo nuevas pruebas de trazado de figuras geométricas simples representadas en un fichero de imagen. Podemos ver el resultado de una de las pruebas de corte sobre material plástico en la Figura 36.

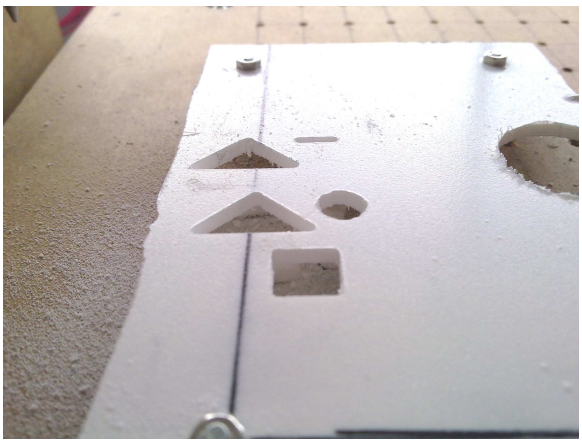

Figura 36: Resultado de prueba de corte de figuras geométricas simples sobre material plástico

Capítulo 6: Conclusiones

Mediente el uso de dispositivos de lógica programable se ha conseguido implementar el control numérico computarizado para una fresadora de tres ejes. Se han incluido dos modos de funcionamiento, manual y automático, así como la monitorización mediante el uso de displays de siete segmentos y leds de la posición de cada uno de los ejes de la fresadora.

Un aspecto importante a señalar en el desarrollo del proyecto es el diseño modular y configurable que se han implementado. Este hecho permite su uso en gran número de aplicaciones para todo aquello relacionado con trazado en dos dimensiones (impresoras, plotters, dispositivos de movimiento de precisión, etc). Un ejemplo de ello ha sido la integración de este proyecto con el trabajo realizado en el proyecto fin de carrera de Héctor Gálvez Barrios para interpretar modelos dos dimensiones partiendo de una imagen.

Por otra parte también permite la incorporación de funcionalidad adicional. Así una posible mejora sería el incluir la funcionalidad necesaria para el trazado de modelos en tres dimensiones. Incluso se podría dar el caso, ya que las FPGA nos lo permiten, de ampliar el Control Numérico para que disponga de un mayor número de ejes.

Otro aspecto a destacar es la construcción de nuestro prototipo para poder llevar a cabo la validación del Control Numérico obtenido. Resaltar que además de haber conseguido un prototipo funcional, el coste total de material no ha superado los 150 euros, aunque hay que indicar que el 40% de los materiales se ha obtenido mediante reciclado de componentes. Comparando esta cifra con el precio de un equipo de fresado comercial, el más económico que hemos podido encontrar ascendía a 1500 euros sin electrónica, consideramos que hemos superado el objetivo de minimizar los costes.

El hecho de tener un prototipo en funcionamiento nos ha facilitado la posterior validación, mediante pruebas satisfactorias, tanto de la implementación de nuestro Control Numérico como de la integración que se ha llevado a cabo con un segundo Módulo de Generación de Coordenadas partiendo de modelos representados mediante imágenes en dos dimensiones.

Bibliografía

[1] http://es.wikipedia.org/wiki/M%C3%A1quina_herramienta

[2] http://es.wikipedia.org/wiki/Mecanizado

[3] http://es.wikipedia.org/wiki/Productividad

[4] www.lpkfusa.com

[5] 2008 IEEE *Standard* VHDL *Language Referente Manual*

http://standards.ieee.org/findstds/standard/1076-2008.html

[6] Felipe Machado Sánchez, Susana Borromeo López, Cristina Rodríguez Sánchez. Diseño avanzado de circuitos digitales con VHDL v1.0 octubre 2010.

[7] Miguel Ángel Freire Rubio. Introducción al lenguaje VHDL.

[8] Universidad Rey Juan Carlos. Asignatura: Electrónica Digital II: http://gtebim.es/docencia/EDII

[9] www.verilog.com/IEEEVerilog.html

[10] www.xilinx.com

[11] www.model.com

[12] www.digilentinc.com/Products/Detail.cfm?NavPath=2,66,828&Prod=ADEPT2

[13] www.lis.inpg.fr/realise_au_lis/kicad/

[14] http://gtebim.es/investigacion/labtel

[15] www.lpkfusa.com/RapidPCB/CircuitboardPlotters/c60.htm

[16]www.eminebea.com/content/html/en/motor_list/pm_motor/pdf/pm55l048.pdf

[17]www.mitsumi.co.jp/latest/Catalog/compo/motor/m42sp5_e.html

[18] www.avagotech.com/docs/AV02-1429EN

[19] http://focus.ti.com/docs/prod/folders/print/sn754410.html