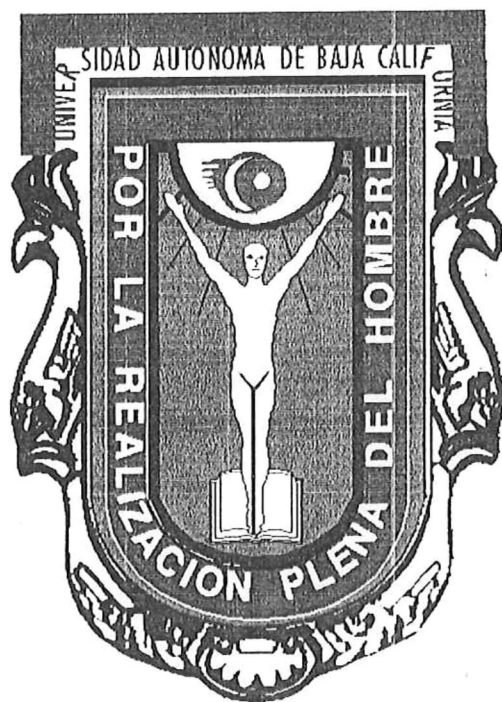


UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

FACULTAD DE INGENIERÍA



*MEDICIÓN DE LA ESTABILIDAD EN OSCILADORES
EMPLEANDO EL ESTIMADOR DE LA VARIANZA DE ALLAN*

TESIS PROFESIONAL

PARA OBTENER EL TÍTULO DE
INGENIERO EN ELECTRÓNICA
PRESENTADA POR:

MANUEL ORTIZ AGUIRRE

Ensenada, B.C. Octubre del 2002

UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

FACULTAD DE INGENIERÍA

*MEDICIÓN DE LA ESTABILIDAD EN OSCILADORES EMPLEANDO EL ESTIMADOR
DE LA VARIANZA DE ALLAN*

TESIS PROFESIONAL

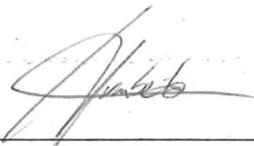
PARA OBTENER EL TÍTULO DE
INGENIERO EN ELECTRÓNICA
PRESENTADA POR:

MANUEL ORTIZ AGUIRRE

DIRECTOR DE TESIS

M.C. HUMBERTO CERVANTES DE ÁVILA

APROBADA POR:



M.C. Humberto Cervantes de Ávila
Director de tesis



Ing. Julia Sánchez Rodríguez
Sinodal Propietario



Ing. Ernesto Rocha Guerrero
Sinodal Propietario

Ensenada, B.C.

Octubre del 2002

AGRADECIMIENTOS

A DIOS por permitir (una vez más) culminar uno de mis tantos sueños.

A mi Madre por ser el pilar y el apoyo moral de todo lo que he emprendido.

A mi Padre por enseñarme que se puede lograr todo lo que se emprenda, con tan solo tener Fé en uno mismo.

A mis hermanas:

Paty gracias por ser amiga además de hermana, (comadre) tu apoyo incondicional es uno de mis mayores orgullos.

Alicia aunque peque y testaruda también te aprecio muchísimo, espero algún día asistir a tu examen profesional.

Al conjunto de personas que menciono (sin importar el orden ya que para todos tengo el mismo deseo de Gratitud):

M.C. Humberto Cervántes de Ávila por compartir sus conocimientos en el aula de clases y por ayudarme a ver realizado un sueño más.

M.C. Héctor Mejía Vázquez por ser asesor, compañero y por soportar todo mis ires y venires hasta culminar mi Tesis.

Familia Munguía Ornelas (No quiero que se escape ninguno) por adoptarme como uno más de la familia y por todo ese apoyo que he sentido desde hace ya muchos años, espero algún día poder corresponder de igual manera.

Julio C. gracias amigo por estar siempre conmigo en las buenas y en las malas, espero siempre poder contar con amigos como Tú. Gaby Elizaola e Imelda Vital por estar también incondicionalmente a través de todos estos años.

A mis compañeros por compartir conmigo tantas desveladas, exámenes, pachangas (aunque usted no lo crea !!!); por que se volvieron para mí, mi segunda familia:

Luis Sandoval y Familia por abrirme las puertas de su casa, por ser además de compañero, amigo y ayudarme en los momentos necesarios.

Eduardo Contreras y Familia por portarte igual que Sando, espero contar siempre con tu amistad y corresponder si algún día necesitas de mí.

Iván Castañedas y Familia gracias por apoyarme y acudir a mí, cuando a veces yo tampoco tengo respuestas para tus dudas y sobre todo por crearme.

A la familia Rivas Gómez (¿ No que no Comadre ?) Por apoyarme en una etapa muy difícil de mi vida (¿Recuerdas octavo?) y por los consejos gracias Angela. Hugo gracias también por apoyarme en lo de mis papeles a parte de la escuela, ese es un detalle que nunca olvidaré.

Alejandro Preciado, Fernando Carrasco, Jesús Alfaro por esos grandes viajes y congresos compartidos juntos (Compas, que tiempos aquellos).

Al resto de mi generación: Picudo, Leal, Piedra, Chilo, Martín y Olimpia por compartir muchas vivencias dentro del aula de clases, aunque algunos de ustedes no pasaran tanto tiempo como el resto en la misma.

A mis maestros (que fueron varios !!!) por compartir sus conocimientos con todos nosotros, (espero que no se hayan reservado ninguno).

A todos aquellos compañeros de diferentes generaciones (Disculpen si falta alguien): Padilla, Quiroz, Hiram, Trenado, Ruly, Choco, Yadir, Juan Carlos, Chaparro, Guero, Muraoka, Peque, Luis Rodríguez, Zague, Colorado, Magno, Escoto, Héctor Gómez, Enrique Gómez, Casimiro, Ramos, Aguilera, Fuji, Oscar Villalvazo, Olimpia Dominguez, Noemí Guzmán, Pedro Esqueda, Cheo, Francisco Andrade, René Estrada, Carlos Glez., en fin a todos aquellos que por alguna u otra razón compartimos las mismas instalaciones, por que gracias a todos ustedes y personas que faltan por mencionar, siempre existió un ambiente de camaradería en nuestra escuela, haciendo con esto más leves las pesadillas que nos daban nuestros maestros.

Por último a todas aquellas personas que faltan por mencionar, que de una u otra manera han participado en la vida de un servidor, por todo lo bueno que he recibido de ustedes, que aunque no los mencione (por falta de espacio) ustedes saben a quienes me refiero.

Gracias

INDICE

CAPITULO I

I.	Objetivo	1
I.1	Introducción al arreglo de compuertas programables	3
I.1.1	Arquitectura del arreglo de compuertas programables	4
I.2	Procesador de la Varianza de Allan	8
I.2.1	Diagrama a cuadros	9

CAPITULO II

II.	Especificaciones y consideraciones sobre el estimador	12
II.1	Consideraciones para la implementación del circuito empleando FPGA	15
II.1.1	Observabilidad de las señales de configuración de la pastilla	16
II.1.2	Observabilidad de las señales de verificación de las funciones Implementadas en el dispositivo	23
II.1.3	Disposición adecuada de terminales	23
II.1.4	Reducción de ruido de "Ground Bounce"	24
II.1.5	Consideraciones de Meta-estabilidad	24

CAPITULO III

III.	Diseño con FPGA de XILINX	25
III.1	Funcionamiento de XACT	25
III.1.1	Introducción del diseño	27
III.1.2	Capturador esquemático	27
III.1.3	Implementación del diseño	30
III.1.3.1	XDM, XMAKE	30
III.1.3.2	Archivos generados por XMAKE	32
III.1.3.3	XNFMERGE, XNFPREP, XNFMAP, MAP2LCA, PPR, MAKEBITS, MAKEPROM	33
III.1.4	Verificación del diseño	38
III.1.4.1	XDELAY, LCA2XNF, XNFBA, XDF	39

CAPITULO IV

IV.	Diseño de la tarjeta	42
IV.1	Captura esquemática	45
IV.2	Configuración del FPGA	48
IV.2.1	Simulación	49
IV.2.2	Configuración	50

CAPITULO V

V.	Pruebas y resultados	53
V.1	Verificación de datos en LATCH6	53
V.2	Verificación de alimentación, observación de fase de inicialización y limpia del FPGA	55
V.3	Fase de configuración de pastilla	56
V.4	Verificación del funcionamiento del contador y cerrojos asincrónicos	58
V.5	Verificación del contador y cerrojos sincrónicos	60
V.6	Generación del algoritmo de compensación para la cuenta capturada	62

CAPITULO VI

VI	Conclusiones	64
----	--------------	----

REFERENCIAS	65
--------------------	-----------

APÉNDICE A	A1
-------------------	-----------

APÉNDICE B	B1
-------------------	-----------

APÉNDICE C	C1
-------------------	-----------

INDICE DE TABLAS Y FIGURAS

TABLAS

4.I	Tabla de compensación de contador asincrónico	42
4.II	Distribución de terminales del FPGA	42
5.I	Terminales de análisis en LATCH6 para realización de pruebas	53
5.II	Terminales de análisis en FPGA para realización de pruebas Fase de configuración de pastilla	56
5.III	Terminales de análisis en FPGA para realización de pruebas Contador y cerrojos asincrónicos	58
5.IV	Terminales de análisis en FPGA para realización de pruebas Contador y cerrojos sincrónicos	60
5.V	Tabla de compensación para la cuenta capturada por contador asincrónico	63

FIGURAS

1.1	Arquitectura del FPGA	4
1.2	Estructura de un CLB	5
1.3	Estructura interna de IOB	6
1.4	Interconexiones	7
1.5	Diagrama a cuadros del procesador de la varianza de Allan	10
2.1	Contador asincrónico	11
2.2	Circuito de mando de captura	12
2.3	Circuito de retardo	12
2.4	Circuito acelerador de acarreo	13
2.5	Fases de configuración de un LCA	19
2.6	Secuencia de configuración	21
3.1	Metodología del diseño del sistema de desarrollo XACT	25
3.2	Pasos de diseño XILINX-VIEWLOGIC	26
3.3	Metodología de diseño utilizando VIEWLOGIC	29
3.4	Flujo de operación de XMAKE	31
3.5	Flujo de operación de PPR	36
4.1	Contador pre-escalador (circuito parcial)	43
4.2	Cerrojo (circuito parcial)	44
4.3	Circuito FPGA (Terminales de entrada y salida)	45
4.5	Contador síncrono, cerrojos y reloj del contador	46
4.6	Generador de acarreo	47
5.1	Diagrama de tiempos de LATCH6	54
5.2	Comportamiento de VCC e INIT	55
5.3	Fase de Inicialización de configuración de FPGA	57
5.4	Fase de fin de configuración de FPGA	57
5.5	Diagrama de tiempos del contador asíncrono	59
5.6	Diagrama de tiempos del contador síncrono parte baja /HC1 activa	61
5.7	Diagrama de tiempos del contador síncrono parte alta /HC2 activa	62

I. OBJETIVO

- En el desarrollo del presente proyecto de tesis se iniciará el estudio de la tecnología de FPGA's como herramienta para la innovación y simplificación en el diseño de circuitos digitales.
- Se integrarán los conocimientos sobre esta tecnología, así como los adquiridos durante la formación del profesionista para la solución de problemas y/o mejoras del diseño, aplicándolos para la integración del contador de alta velocidad del Procesador de la Varianza de Allan.

INTRODUCCIÓN

En el Laboratorio de Comunicaciones Ópticas del departamento de Física Aplicada, del Centro de Investigación Científica y Educación Superior de Ensenada (CICESE); se realizan diseños surgidos de proyectos de investigación o para apoyar el aprendizaje de los estudiantes del mismo. En ocasiones se requiere diseñar equipo electrónico para sustituir a otro, que puede ser adquirido en el mercado pero tal vez bastante costoso para alguna aplicación en específico. El presente proyecto de tesis surge de la necesidad de un instrumento, el cual es utilizado para la medición de frecuencia, que a la fecha de inicio de la misma ya se había desarrollado de manera óptima; la justificación reside en el aprendizaje de una nueva tecnología conocida como ASIC¹, que en este caso en particular es mediante el uso de FPGA² de XILINX™.

En el presente capítulo se explicará el funcionamiento interno del FPGA así como también se dará una breve descripción del procesador de la varianza de Allan.

En el capítulo 2 se mencionarán las especificaciones y consideraciones sobre el procesador de la varianza de Allan así como la descripción para la configuración del FPGA.

¹Circuitos Integrados de Aplicación Específica (Application Specific Integrated Circuit)

² Arreglo de Compuertas Programables por Campo (Field Programmable Gate Array)

CAPITULO I

En el capítulo 3 se describirá la metodología del diseño con FPGA de XILINX™, se dará una descripción del sistema de desarrollo XACT utilizado en el desarrollo del proyecto, así como también la descripción en el proceso de la implementación del diseño.

En el capítulo 4 se hablará del diseño de la tarjeta, captura esquemática, simulación y configuración del FPGA.

El capítulo 5 mostrará pruebas y resultados de sistema y finalmente el capítulo 6 mencionará las conclusiones obtenidas al desarrollo del presente proyecto.

I.1 INTRODUCCIÓN A ARREGLOS DE COMPUERTAS PROGRAMABLES [1]

El constante avance en los niveles de integración en circuitos electrónicos han mejorado muchas características de equipos, reduciendo costos, consumo de potencia y tamaño de los sistemas, incrementando el desempeño y la confiabilidad. Los niveles de incremento de integración son más evidentes en IC's³ microprocesadores y memorias. Con cada generación de procesos, la apertura de tecnología entre estos circuitos VLSI⁴ y otros IC's lógicos estándar se han ampliado. Para lograr densidades comparables a sus funciones lógicas propietarias, los diseñadores de equipo digital han sido obligados a considerar ASIC's programados de fábrica personalizados y semipersonalizados.

Los FPGA's son ASIC's de alta densidad que pueden ser configurados por el usuario. Estos combinan los beneficios de integración lógica de VLSI personalizados con el diseño, producción y las ventajas tiempo-mercado de los productos estándar. Los diseñadores definen las funciones lógicas del circuito y revisan estas funciones como sea necesario; Así los FPGA's pueden ser diseñados y verificados en pocos días, como también los cambios pueden ser realizados en tan solo unas horas, opuesto a las varias semanas que tomaría para arreglos de compuertas personalizadas. Esto puede resultar en ahorros significativos de costos disminuyendo los riesgos de cambios de diseño, reprogramación y eliminando costos de ingeniería sin repetición.

A diferencia de los arreglos de compuertas convencionales, los FPGA's no requieren costos fijos ni fabricación personalizada de fábrica. Dado que cada dispositivo es idéntico, los costos de manufactura siguen la misma curva de aprendizaje como otros productos IC's estándar de alto volumen.

³ Circuitos Integrados (Integrated Cicuits)

⁴ Integración de escala muy larga (Very Large Scale Integration)

I.1.1 ARQUITECTURA DEL ARREGLO DE COMPUERTAS PROGRAMABLES

El arreglo de celdas lógicas (LCA⁵) propiedad de XILINX, es una arquitectura similar a otros arreglos de compuertas con una matriz interior de bloques lógicos y anillos periféricos de bloques de I/O⁶ (Figura 1.1) . Las fuentes de interconexión ocupan los canales entre los renglones y columnas de bloques lógicos, y entre los bloques lógicos y los bloques de I/O.

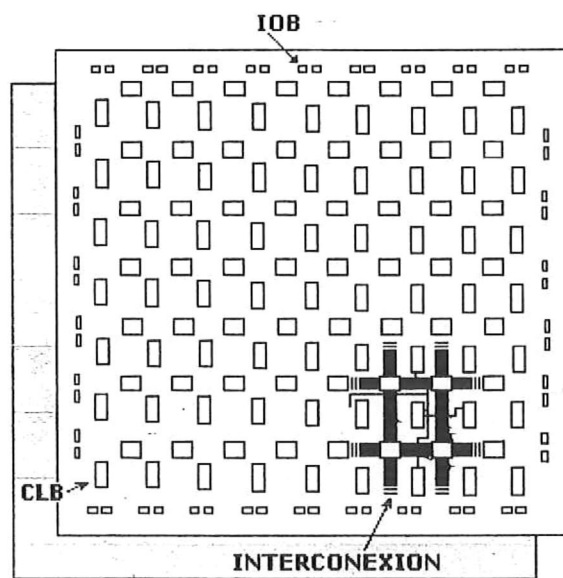


Figura 1.1. Arquitectura del FPGA

Al igual que un microprocesador, un dispositivo LCA es un dispositivo lógico manejado por programa. Las funciones de los bloques lógicos configurables y los bloques del LCA y su interconexión, son controlados por un programa de configuración, almacenado en una memoria. El programa de configuración es cargado automáticamente de una memoria externa, al encender el sistema o al ser indicado por mando; o es programado por un microprocesador como parte de un sistema de inicialización.

⁵ Arreglo de celdas lógicas (Logic Cell™ Array)

⁶ Entrada/Salida (Input/Output)

El perímetro de Bloques de Entrada/Salida (IOB) proveen una interfaz entre el arreglo lógico interno y las terminales del dispositivo. El arreglo de Bloques lógicos configurables (CLB) realiza funciones lógicas especificadas por el usuario. Los recursos de interconexión son programados para formar redes, llevando señales lógicas entre bloques de manera similar a las rutas de los circuitos impresos en circuitos de baja y mediana escala de integración.

Bloques lógicos configurables (CLB)

El centro de un dispositivo LCA es una matriz de CLB's⁷. Cada CLB (Figura 1.2) contiene registros de almacenamiento y lógica combinacional programable. La sección lógica combinacional de un bloque es capaz de implementar cualquier función booleana de entradas variables. Los registros pueden ser cargados de la lógica combinacional o directamente de la entrada de un CLB. Los registros de salidas pueden manejar la lógica combinacional directamente vía una ruta de retro-alimentación interna.

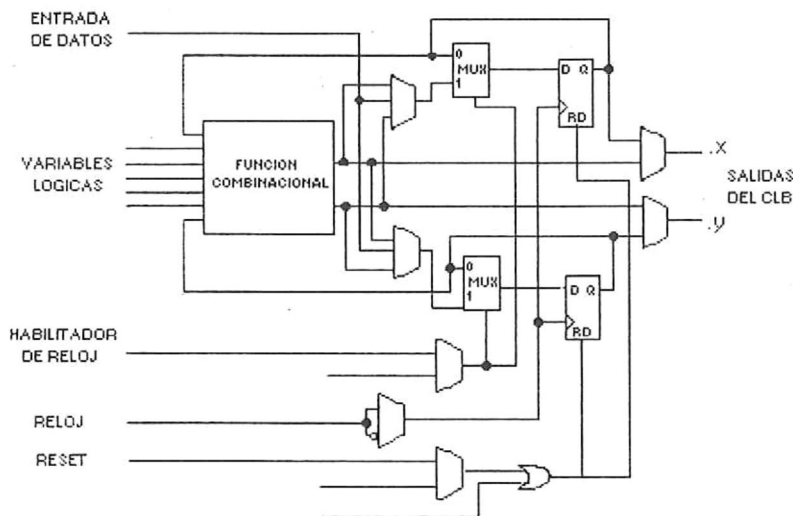


Figura 1.2. Estructura de un CLB

⁷ Bloques lógicos configurables (Configurable Logic Block)

Bloques de Entrada/Salida (IOB)

El periférico de un dispositivo LCA está hecho de IOB's⁸ programables por el usuario. Cada bloque puede ser programado independientemente para ser una entrada, una salida con control de 3^{er} estado o una terminal bidireccional. Las entradas pueden ser programadas para reconocer niveles TTL o CMOS. Cada IOB (Figura 1.3) incluye también flip-flops que pueden ser utilizados como reforzadores de entradas o salidas. Cada IOB configurable por el usuario provee una interfaz entre la terminal externa del IC y la lógica interna del mismo. Su configuración permite a cada IOB su inversión, un slew rate controlado y una alta impedancia de pull up. Así mismo cada circuito provee diodos de entrada de conmutación que proporcionan protección electrostática y circuitos para inhibir bloqueos producidos por corrientes de entradas.

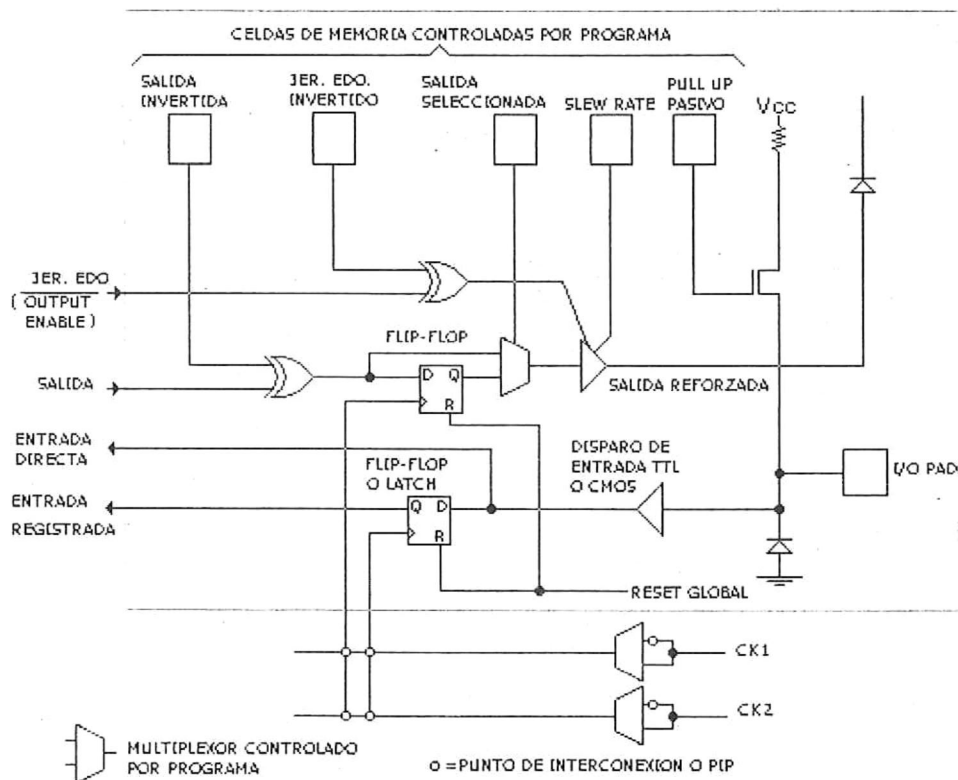


Figura 1.3. Estructura interna de IOB

⁸ Bloques de Entrada/Salida (Input Output Block)

Interconexión

La flexibilidad de un LCA es debido a los recursos de programación que controlan la interconexión de 2 puntos en el chip. Al igual que otros arreglos de compuertas los recursos de interconexión (Figura 1.4) incluyen una red de líneas de metal de 2 capas que corren horizontalmente y verticalmente entre los renglones y columnas de los CLB's. Las conexiones programables unen las entradas y salidas de los IOB's y CLB's a las líneas de metal cercanas. Los puntos de unión e intercambio de intersecciones de renglones y columnas pueden unir señales de una trayectoria a otra. Líneas largas corren a lo largo o ancho del integrado, desviando intercambios para proveer distribución de señales críticas con un retardo mínimo.

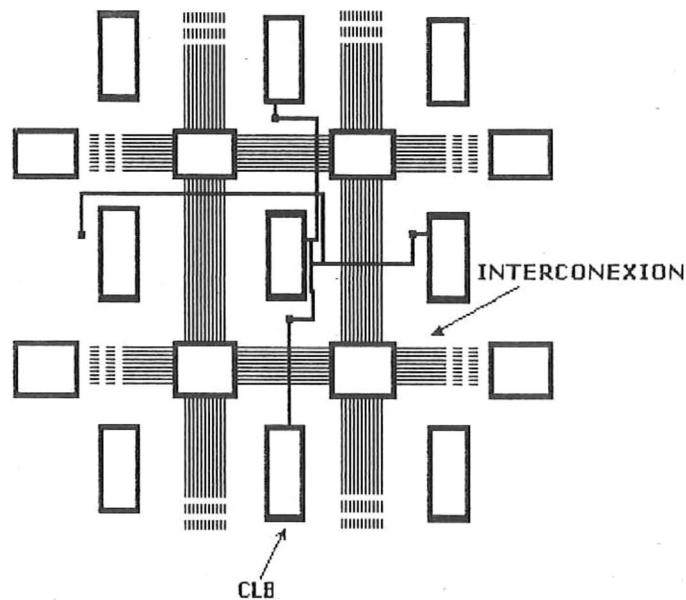


Figura 1.4. Interconexiones

I.2 PROCESADOR DE LA VARIANZA DE ALLAN

Antecedentes

Dentro de los diversos aspectos que involucra la caracterización de componentes de comunicaciones ópticas coherentes, se encuentra el aspecto de la medición de la estabilidad de fuentes osciladoras estables tal como el láser semiconductor. Para poder realizar la medición de dicha característica es necesario utilizar equipo muy sofisticado, el cual la mayoría de las veces, es de usos general y de costo elevado.

Dado que se conoce tanto al estimador como el algoritmo de medición para la determinación de la estabilidad, el paso inmediato es su implementación. En CICESE se ha trabajado desde 1993 en este tipo de instrumentación; sin embargo, en la actualidad se hace necesario una innovación empleando nuevas herramientas de síntesis y desarrollo que beneficien tanto al proyecto como a la formación de recursos humanos en el área de instrumentación para comunicaciones ópticas.

El sistema de Procesamiento en tiempo real de la varianza de Allan elaborado en CICESE[2] había sido implementado empleando componentes MSI y VLSI sobre una tarjeta de circuito impreso de 4 capas presentando un desempeño aceptable. Dado que el procesador emplea dispositivos de alta frecuencia y maneja señales con transiciones rápidas – con tiempos de subida de 2 ns- se incorporaron al circuito impreso técnicas de reducción de ruido apropiadas para la aplicación. Debido a las limitaciones para la elaboración de circuitos impresos –área acotada, baja confiabilidad en multicapa- el funcionamiento de la tarjeta se fue degradando al presentarse un desprendimiento paulatino de las capas. Por estos motivos se realizaron estudios para incorporar FPGA de XILINX en la circuitería, como un medio de disminuir el número de pistas, vías y área de impreso reduciendo con esto las posibilidades de falla, así como una rápida detección de éstas al concentrar funciones en forma modular.

Descripción del funcionamiento

El procesador de la Varianza de Allan [3] permite capturar la información sobre la estabilidad de la señal empleando una referencia de calidad bajo el criterio de medición de frecuencia directa. Para lograr lo anterior se implementa un sistema de adquisición de frecuencia basado en un muestreador de cuenta, al que se denomina generador de intervalo de integración, mismo que proporciona la cadencia de muestreo sobre una serie de contadores de alta velocidad. La muestra, en este caso de frecuencia, se almacena en un banco de memoria de alta velocidad que al llenarse en el número de muestras solicitadas por el usuario se canaliza hacia la computadora personal IBM compatible empleando el ducto EISA. Esta memoria de alta velocidad es necesaria para intervalos de integración del orden de algunos microsegundos donde la velocidad de transferencia y la demanda de atención de la computadora personal no pueden competir. De acuerdo a la condición necesaria para el cálculo de la Varianza, sin posteriores compensaciones, en ningún momento se detiene el contador. Obtenidos los datos, que pueden ser del orden de 4000 palabras de 32 bits por evento y hasta 65000 eventos, se procede para cada evento, a calcular la varianza de Allan empleando las respectivas ecuaciones y agregando a la gráfica logarítmica un nuevo punto. Dicha gráfica se presenta en el monitor de la computadora teniendo la posibilidad de almacenar y recuperar los datos de disco en formato ASCII en forma tabular lo que permite su utilización por otros paquetes de cálculo.

A nivel usuario el sistema se configura fácilmente introduciendo el número deseado de muestras por evento así como el intervalo de integración que comprende de 1 μ s a 10 millones de segundos.

I.2.1 DIAGRAMA A CUADROS

El diagrama a cuadros del sistema se muestra en la Figura 1.5. Las funciones que desempeñan los diferentes elementos se describen a continuación.

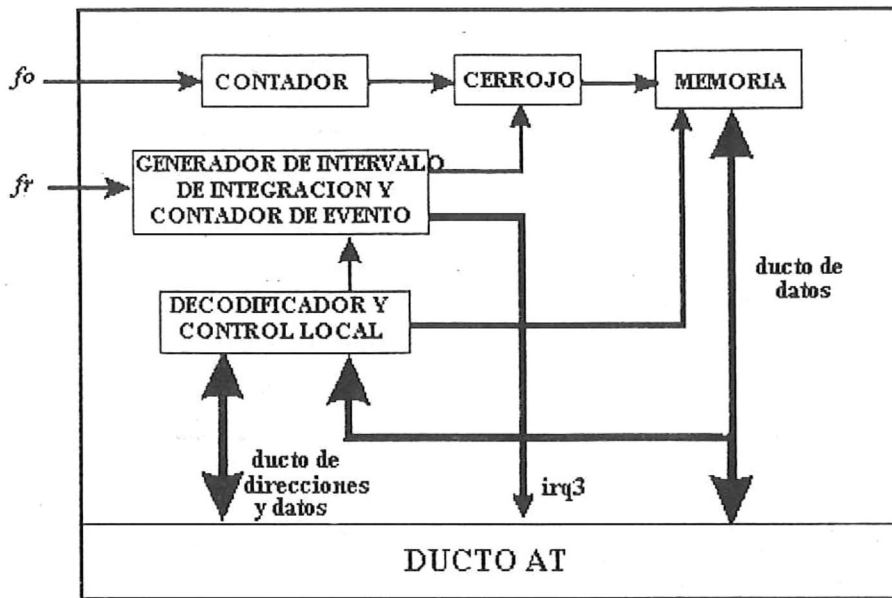


Figura 1.5. Diagrama a cuadros del procesador de la varianza de Allan

Contador híbrido: Contabiliza los pulsos ocurridos dentro de un intervalo de integración τ , a la frecuencia del oscilador bajo prueba f_0 . Este dispositivo se compone de dos tipos de contadores; el primero emplea un esquema de rizo que le permite trabajar a altas frecuencias mientras proporciona un reloj de baja frecuencia al siguiente contador; la segunda etapa es un contador síncrono de baja frecuencia que contiene la mayor parte de los bits de conteo.

Cerrojo: Captura la cuenta de los pulsos del contador al finalizar cada intervalo de integración. Este dispositivo se divide en una parte de alta frecuencia en donde se capturan los bits menos significativos empleando un esquema de retardos; y, una parte de baja frecuencia donde se capturan los bits restantes.

Memoria: Almacena localmente las muestras capturadas permitiendo que el sistema trabaje a intervalos de integración cortos.

Decodificación y Control Local: Direcciona el flujo de mandos de configuración de la tarjeta.

Generador de intervalo de integración y Contador de evento: Genera los intervalos de integración de acuerdo a los requisitos de la medición y contabiliza el número de eventos o muestras capturadas. Los intervalos de integración pueden configurarse de 10^{-6} a 10^6 s, mientras que el número de eventos puede ser hasta de 1000.

Interfaz con usuario: Configura a la tarjeta, manipula la información proveniente de esta y la transforma para su visualización y almacenamiento temporal o permanente. El ambiente de interacción es amigable y emplea, de manera intensiva, ventanas, iconos y gráficas.

II. ESPECIFICACIONES Y CONSIDERACIONES SOBRE EL ESTIMADOR

Como se describió en el capítulo anterior, el sistema consta de 3 bloques fundamentales: Un sistema de muestreo de frecuencia directa, un muestreador de cuentas y la programación. El sistema de muestreo está formado por un contador de 32 bits, con acceso en palabras de 8 ó 16 bits lo cual permite al estimador trabajar en el ducto de una computadora personal AT (por lo que se elimina la utilización de una fuente de alimentación adicional); el muestreador de cuentas se encarga de generar el intervalo de integración de las muestras; y la programación es utilizada para manejar tarjetas, recuperar y procesar la información adquirida, así como interactuar con el usuario. En este capítulo nos enfocaremos a la descripción del adquisidor de frecuencia para realizar su implementación.

La función del sistema de muestreo de frecuencia es contabilizar los pulsos de la señal de entrada en forma directa, por lo cual se requiere un contador de alta velocidad. Este contador es dividido en un contador pre-escalador que le permite al estimador trabajar a altas frecuencias, así como proporcionar un reloj de baja frecuencia al resto del contador (contador sincrónico).

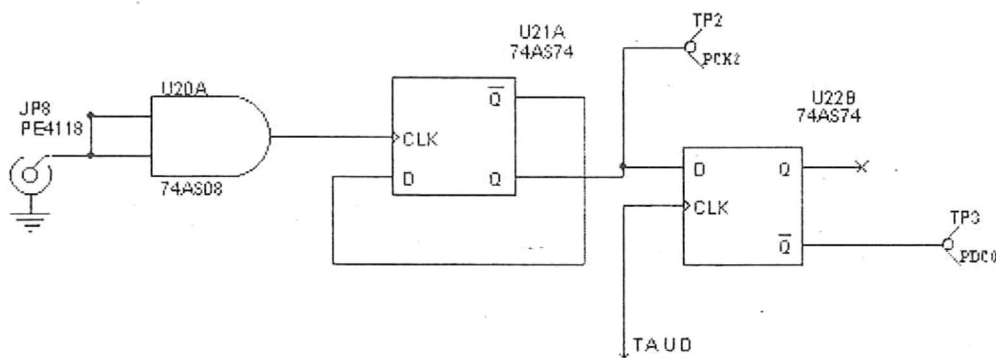


Fig. 2.1 Contador asincrónico

El contador asincrónico (fig 2.1), emplea un conteo de rizo que permite generar los 3 bits menos significativos, mientras reduce la frecuencia del oscilador bajo prueba por un factor de 2 en cada etapa; este pre-escalador cuenta con un cerrojo el cual almacena temporalmente el valor del contador, asegurando de esta manera un evento estable el cual es atrapado por el mando de captura (fig. 2.2)

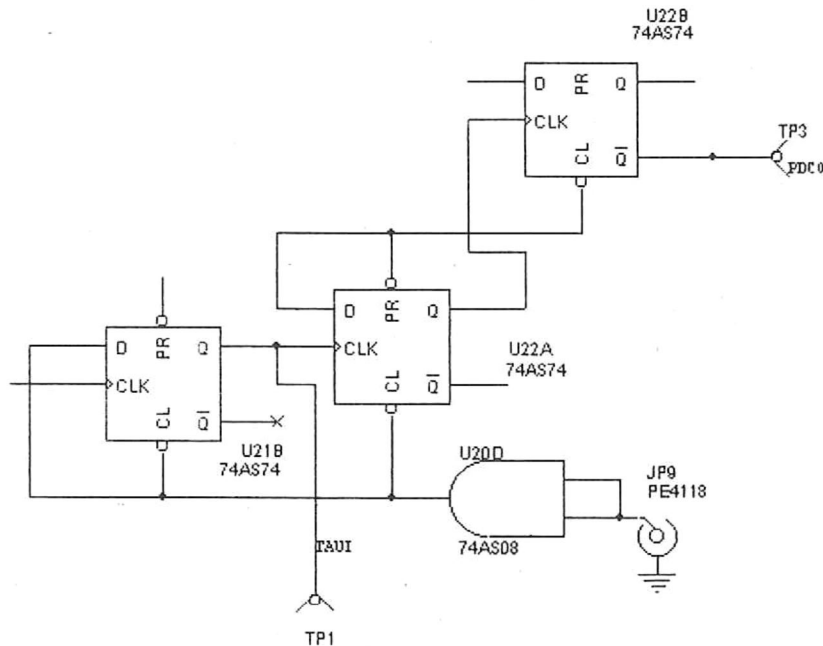


Fig. 2.2 Circuito de mando de captura

El sincronizador de mando de captura se encarga de la sincronía de la señal del oscilador bajo prueba con el mando de habilitación de conteo. El 3^{er} bit a su vez proporciona el reloj del contador síncrono; debido al cambio de velocidad de este respecto al contador asíncrono, es necesario retardar esta señal y así poder asegurar la captura de un evento estable del resto del contador y prevenir un posible estado de Metaestabilidad¹. Esto se logra implementando un circuito de retardo (fig. 2.3). El contador síncrono a su vez, da los 29 bits más significativos de la cuenta, obteniéndose de esta manera la cuenta total que será almacenada en memoria para su implementación en el algoritmo propuesto.

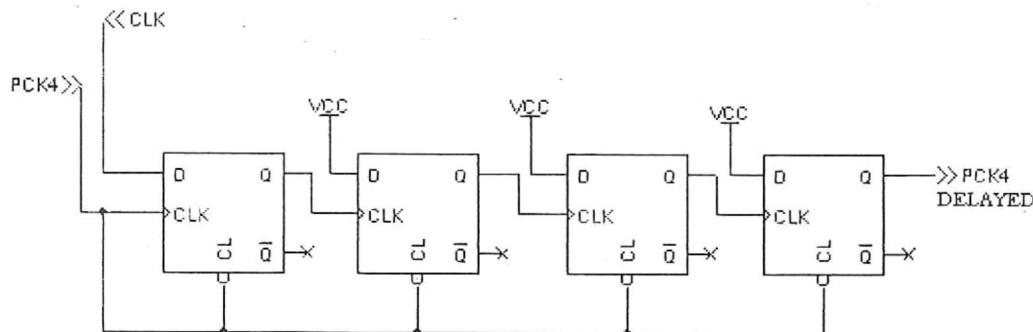


Fig.2.3 Circuito de retardo

¹ Un estado de meta-estabilidad es definido como el período de tiempo en el cual la salida de un dispositivo digital no es un nivel lógico 1 (voltaje mayor a 2.5 V) o un nivel lógico 0 (voltaje menor a 0.8 V) pero se encuentra entre 0.8 V y 2.5 V (estado de indecisión).

Para acelerar el proceso de cuenta es preciso también, utilizar un arreglo lógico el cual nos reduce el retardo de propagación del acarreo (fig. 2.4), producido por los mismos contadores.

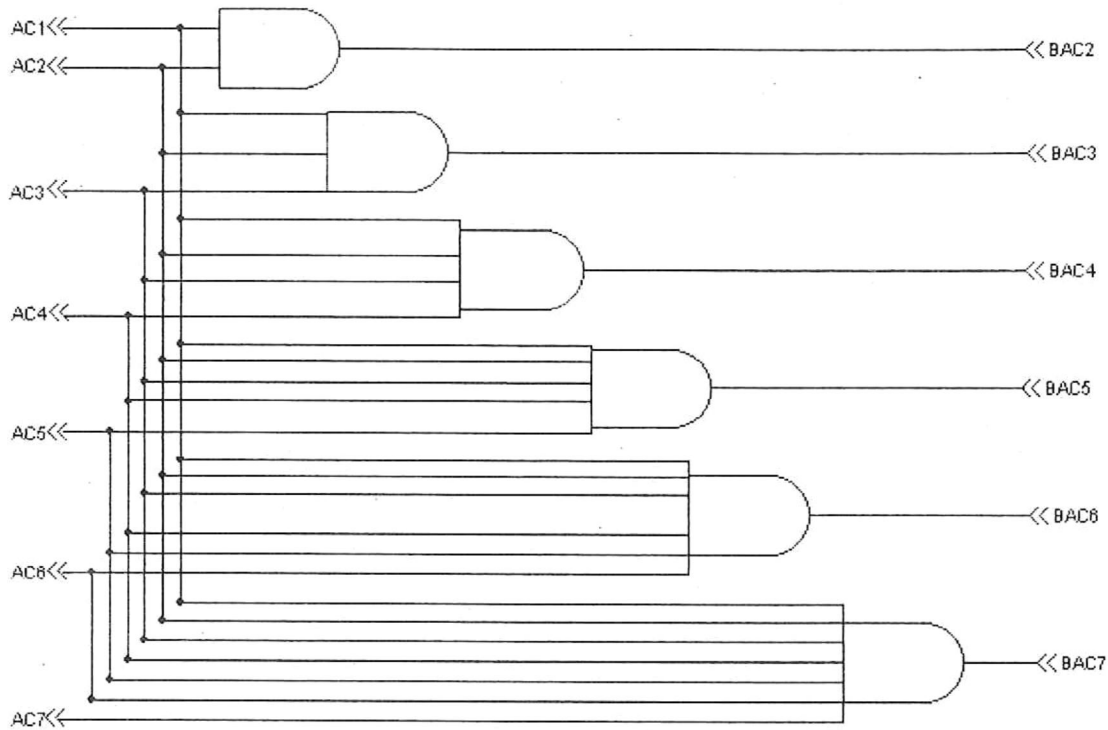


Fig. 2.4 Circuito acelerador de acarreo

Antes de definir el proceso de diseño, es necesario tomar en cuenta varias consideraciones para la implementación del mismo.

II.2 CONSIDERACIONES PARA LA IMPLEMENTACIÓN DE CIRCUITOS EMPLEANDO FPGA's.

Dentro de la implementación empleando FPGA's es necesario considerar los siguientes puntos:

- Observabilidad de las señales de configuración de la pastilla.
- Observabilidad de las señales de verificación de las funciones implementadas en el dispositivo.
- Disposición adecuada de terminales.
- Reducción de ruido de "ground bounce" a través de:
 - ◊ Desacoplamiento de la fuente de alimentación.
 - ◊ Acoplamiento de impedancias con otros dispositivos externos.
 - ◊ Separación física y/o desapareamiento de señales que presenten transiciones simultáneas.
- Consideraciones de Metaestabilidad

A continuación describiremos cada uno de los puntos mencionados centrandó nuestra atención al caso particular de la implementación de la tarjeta SPTRVA para la cual se empleará un FPGA configurado en modo periférico (existen 3 modos de configuración del mismo como son modo Maestro -Master- el cual recibe los datos de configuración de una memoria externa, modo Periférico -Peripheral- cuyos datos de configuración son enviados por un procesador periférico y modo Esclavo -Slave- en el cual los datos de configuración son recibidos de la salida de datos de otro dispositivo similar).

II.2.1 OBSERVABILIDAD DE LAS SEÑALES DE CONFIGURACIÓN DE LA PASTILLA.

De acuerdo a XILINX, XACT User Guide, en su capítulo “Configuration, Length Count, and Debugging”, es de suma importancia proveer a la circuitería con los elementos que permitan visualizar las señales de configuración de la pastilla de manera de descartar, en el eventual caso de falla, el error proveniente por una inadecuada secuencia de configuración. Antes de describir cual sería una secuencia correcta describiremos las señales involucradas [XILINX, The Programmable Logic Data Book].

DESCRIPCIÓN DE TERMINALES

Terminales permanentemente dedicadas:

CCLK: En modo periférico, este reloj se genera internamente presentando un valor nominal de 1 MHz pero que puede variar debido a la temperatura, proceso y voltaje de alimentación.

DONE / $\overline{PROG}$: Terminal de salida de colector abierto configurable con o sin resistor interno de levantamiento (pull-up resistor) de 2 a 8 K Ω . El estado bajo de esta terminal junto con el de *$\overline{RESET}$* indica que el LCA² está en la fase de limpia o de configuración.

Al finalizar la configuración, la circuitería del dispositivo LCA se activa de manera sincrónica; *DONE* es programable a nivel alto ya sea un ciclo antes o un ciclo después de que las salidas de la pastilla se activan.

Una vez inicializado el dispositivo, una transición de alto a bajo en esta terminal causará una inicialización del LCA y el inicio de una reconfiguración.

$\overline{RESET}$: Terminal de entrada a la que se asocian tres funciones.

²Un LCA es un arreglo de celdas lógicas (Logic Cells Arrays), el cual es el esquema de un FPGA; este a su vez se divide en CLB's e IOB's. Un CLB es un bloque lógico configurable (Configurable Logic Block) el cual conforma la lógica a mediana escala de integración; un IOB es un bloque configurable de entrada/salida (Input/Output Block) los cuales conforman la lógica para la señales de Entrada/Salida.

Antes de iniciar la configuración, un nivel bajo de entrada retarda el inicio del proceso de configuración. Un circuito interno sensa la aplicación de potencia e inicia un ciclo de “tiempo-fuera” (11 a 33 ms). Cuando el tiempo fuera y $\overline{RESET}$ terminan, los niveles de las terminales M0 a M2 son muestreados y la configuración inicia.

Si un $\overline{RESET}$ es dado durante la configuración, el LCA es reinicializado y vuelve a comenzar la configuración al finalizar $\overline{RESET}$.

Si un $\overline{RESET}$ es dado después de completada la configuración, se comporta como un reinicializador global asíncrono tanto de los elementos de almacenamiento de los CLBs como de los IOBs del LCA.

TERMINALES DE USUARIO QUE PUEDEN TENER FUNCIONES ESPECIALES:

$RDY/\overline{BUSY}$: Durante la configuración en el modo periférico esta terminal indica cuando la pastilla está lista para recibir otro octeto. Después de terminar la configuración la terminal puede ser programada por el usuario como de entrada/salida.

$\overline{LDC}$: Durante configuración esta salida se mantiene a nivel bajo para mostrar que la configuración no ha terminado. Después de la configuración esta terminal puede programarse como entrada/salida.

$\overline{INIT}$: Terminal de salida de colector abierto con una débil resistencia de levantamiento. El nivel de salida es bajo durante la estabilización de potencia y limpia de la memoria de configuración. Puede emplearse como una señal que describe el estado del dispositivo. Después de la configuración esta terminal puede programarse como entrada/salida.

$\overline{CS0}, \overline{CS1}, \overline{CS2}, \overline{WS}$: Estas cuatro entradas son empleadas para controlar el proceso de entrada de datos para configuración en el modo Periférico. Los datos D0 a D7 son reconocidos por la lógica de configuración al presentarse una transición de bajo a alto provocada por la falta de cualquiera de estas señales. Después de la configuración estas terminales pueden emplearse como entrada/salida.

D0-D7: Este juego de ocho terminales representa el octeto de configuración para el modo periférico. Después de la configuración estas terminales pueden emplearse como entradas/salidas.

DOUT: Durante configuración esta terminal se emplea como salida de datos de configuración para otros dispositivos conectados en un esquema de cadena de margaritas. Después de la configuración esta terminal puede programarse como entrada/salida.

DESCRIPCIÓN DE LA CONFIGURACIÓN DE UN FPGA XC30XX EN MODO PARALELO

En general podemos dividir en tres las fases de configuración de un FPGA: fase de inicialización, fase de limpia y propiamente la fase de configuración tal como lo muestra la Figura 2.5.

La fase de inicialización comienza cuando un circuito interno recibe un disparo como resultado de aplicar potencia al dispositivo. Dicho circuito promueve un estado de “tiempo-fuera” utilizando un temporizador auto-generado de 14 bits alimentado por un reloj interno de frecuencia nominal de 1 MHz -que puede variar entre 0.5 y 1.5 MHz con la temperatura y voltaje de alimentación- produciendo un letargo de 11 a 33 ms, tiempo suficiente para que el voltaje de alimentación se estabilice.

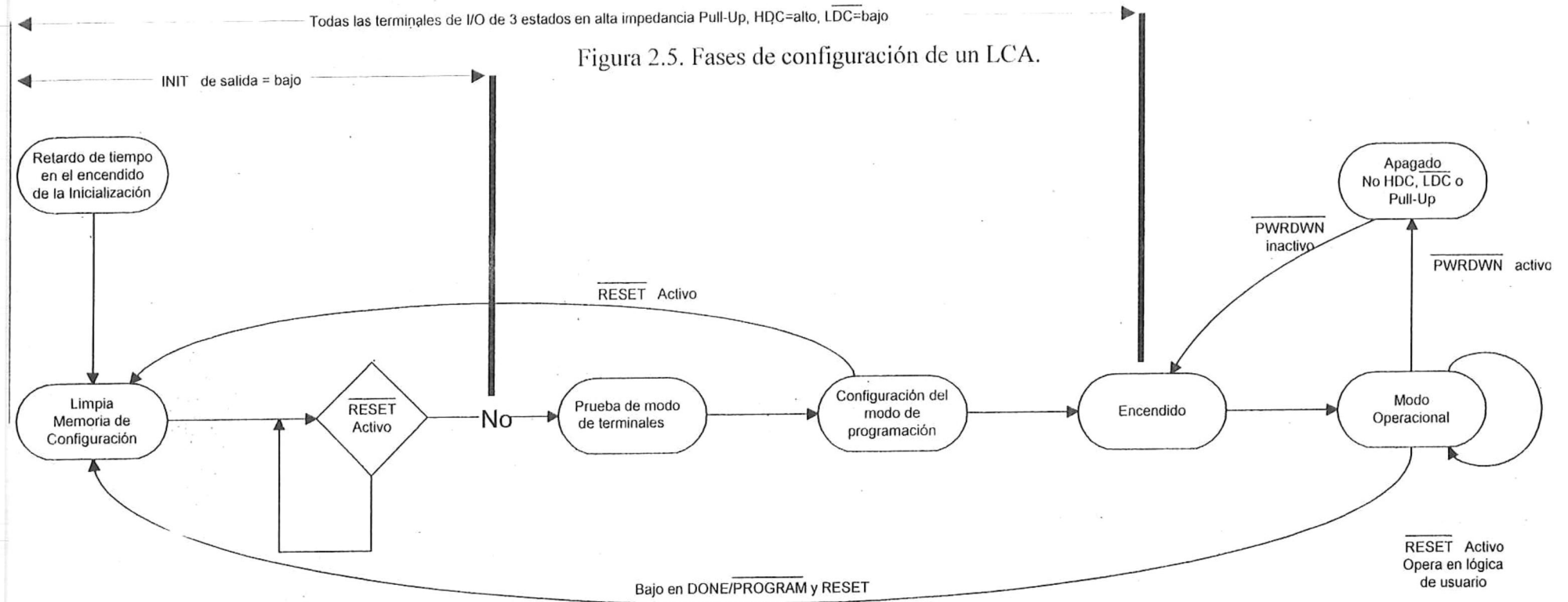
Si el voltaje de alimentación no alcanza de 2.0 V a V_{cc} en menos de 25 ms se recomienda mantener en bajo la señal $\overline{RESET}$ hasta que el voltaje alcance los 4.0 V. En el peor de los casos en el cual el tiempo de subida de V_{cc} sea mayor de 100 ms o presente un comportamiento de subida no monotónico será necesario editar un estado alto de $\overline{RESET}$ por más de 6 μs seguido por un estado bajo tanto en $\overline{RESET}$ como en $\overline{DONE/PROG}$ de no menos de 6 μs después de que V_{cc} haya alcanzado los 4.0 V.

Para establecer el estado en la terminal $\overline{DONE/PROG}$ deberá utilizarse una compuerta que permita formar una Y alambrada.

Dicha compuerta puede llevarse a estado alto y la terminal *DONE / $\overline{PROG}$* por sí misma permanecerá en estado bajo hasta el final de la configuración. De cualquier manera, si por olvido se dejase la compuerta en estado bajo, al terminar la configuración el LCA se tornará activo.

Al finalizar la inicialización, la fase de limpia realiza lo propio sobre la memoria de configuración. Este proceso toma un ciclo de reloj interno (1 μ s nominal) por marco de datos de configuración (197 marcos para XC3020 y XC3020A lo que equivale a 197 μ s nominal o hasta un máximo de 394 μ s).

Figura 2.5. Fases de configuración de un LCA.



Antes de entrar a la fase de configuración, el LCA sensa la ausencia de la señal $\overline{RESET}$ y si esta acción arroja como resultado un estado alto lee las terminales M0 a M2 para detectar el modo en el que se configurará la pastilla. La señal $\overline{INIT}$ indica cuando las fases de inicialización y limpia se llevan a cabo mediante la edición de un estado bajo, por lo tanto permite observar si el integrado está siendo inicializado, con su memoria de configuración borrada ya que al volverse a colocar $\overline{INIT}$ en alto el dispositivo indicará que está listo para recibir datos para configuración. Si la fase de configuración ya inició, entonces al colocar la señal $\overline{RESET}$ en estado bajo al menos durante tres ciclos de reloj interno el LCA abortará la fase de configuración y regresará a la fase de limpia.

Si el dispositivo se encuentra configurado y sensa una transición negativa seguida de un estado bajo por más de 6 μs en la terminal $DONE/\overline{PROG}$ o, si se mantiene a esta terminal permanentemente en bajo y la transición y el estado se da en la terminal $\overline{RESET}$ se inicia la reconfiguración del LCA regresando a la fase de limpia.

Dentro de la fase de configuración, todos los elementos de almacenamiento de los CLBs e IOBs se mantienen reinicializados, los seguidores de salida se mantienen en alta impedancia, la terminal $\overline{INIT}$ se torna activa y se mantiene en nivel bajo y las terminales que tienen funciones especiales, dependiendo del tipo de configuración, se activan (para modo periférico se activan las señales de las terminales $D0-D7$, $\overline{CS0}$, $\overline{CS1}$, $CS2$, $\overline{WS}$, $DOUT$ y $\overline{LDC}$).

El programa de configuración generado por MakePROM³, del sistema de desarrollo XACT⁴, inicia con un preámbulo de 11111110010 seguido de una palabra de 24 bits que indica el número total de pulsos de reloj necesarios para cargar el programa de configuración en el LCA al cual se denomina "longitud del conteo" para finalizar con 1111 (un total de 40 bits). La trama de datos se presenta en la Figura 2.6. La terminal

³ **MAKEPROM** da formato a un archivo de agrupación de bits de configuración (BIT) creando un archivo que puede ser cargado a una PROM.

⁴ **XACT** paquete de **XILINX**, el cual cuenta con utilerías las cuales nos permiten realizar la captura esquemática, la simulación de dicho circuito, así como la implementación de este en un FPGA.

DOUT permite observar los 40 bits y después fija su estado a nivel alto. Por su parte la terminal *RDY / $\overline{BUSY}$* se alterna en estado indicando cuando el dispositivo está listo o no para recibir un nuevo octeto de configuración.

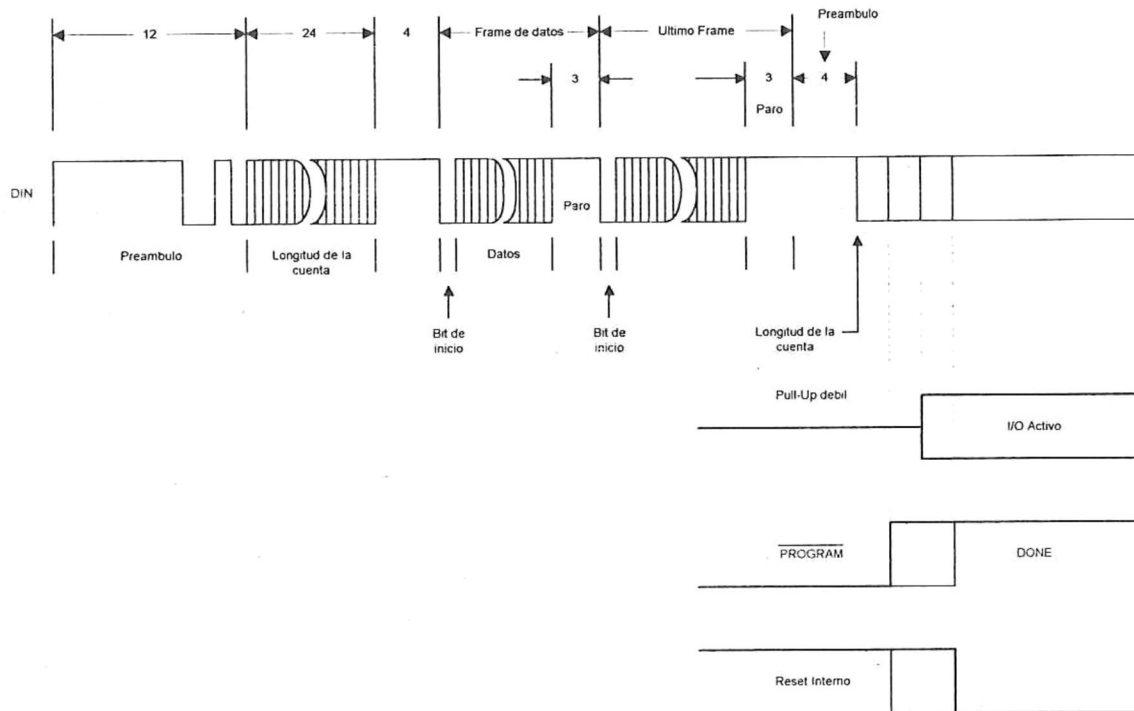


Figura 2.6. Secuencia de configuración.

Al momento de iniciarse la fase de configuración un contador en el LCA parte de cero y comienza a contar el número de pulsos empleados para cargar el programa. El proceso de cargado para configuración termina cuando la longitud de cuenta iguala a la del contador interno y la memoria de configuración está llena. En ese momento, el dispositivo ejecuta una inicialización sincrónica y se vuelve operacional colocando a niveles conocidos las terminales de entrada/salida.

Si la configuración dura de 15 a 25 segundos -indicado por la terminal *DONE / $\overline{PROG}$* - entonces se ha presentado una incompatibilidad entre la longitud del conteo y el número de pulsos de configuración generados. Lo anterior podría ser debido principalmente a ruido y a problemas mecánicos (pistas mal enrutadas o rotas, etc.).

La secuencia de inicialización sincrónica puede programarse a través de MakeBits⁵ de manera de obtener la liberación de la terminal *DONE* / $\overline{PROG}$ junto con la desactivación de la señal de reinicialización global interna ya sea un ciclo antes o un ciclo después de la liberación de las terminales de entrada/salida según lo muestra la Figura 2. Lo anterior permite evitar la contención de señales en las terminales que se encontraban en alta impedancia durante la configuración y que, al finalizar el proceso, se tornan activas.

II.2.2 OBSERVABILIDAD DE LAS SEÑALES DE VERIFICACIÓN DE LAS FUNCIONES IMPLEMENTADAS EN EL DISPOSITIVO.

Dentro de las claves para lograr la depuración de la lógica se encuentra la de observabilidad del comportamiento del LCA para funciones generadas internamente, es decir, señales que no tienen salida hacia el exterior pero que son trascendentales para el funcionamiento adecuado de la lógica programada en el dispositivo. Es claro que estas señales sufren un retardo provocado tanto por la celda de entrada/salida como por la ruta de donde parten por lo que debe tenerse en consideración sobre todo si se manejan señales de reloj o señales de circuitos asincrónicos. De cualquier manera, el retardo puede reducirse desde la captura esquemática con VIEWDRAW (apéndice A) empleando los atributos CRITICAL, LOC y FAST [XILINX, VIEWlogic WORKVIEW SERIES I] o bien, por optimización de la ruta a través del empleo de XDE [XILINX, XACT User Guide] o, como trataremos más adelante, ubicando la terminal de observación cerca de la celda bajo prueba.

Para el caso de la tarjeta SPTRVA se añaden un par de puntos de prueba correspondientes a la sección de retardo del mando TAU y generación de la señal TAUD.

II.2.3 DISPOSICIÓN ADECUADA DE TERMINALES

Una buena distribución de terminales facilita tanto la localización y enrutado de la lógica interna de la pastilla como el enrutado externo de pistas en el circuito impreso.

⁵ MAKEBITS crea la agrupación de bits para un diseño LCA

Si la disposición física de las terminales favorece el enrutado interno de la pastilla, la restricción provocada por el atributo LOC se compensa y permite que PPR produzca, sin muchos compromisos, el archivo .LCA. De lo anterior, existe un compromiso entre el enrutado interno de la pastilla y el externo hacia otros componentes, por lo que habrá de establecerse un análisis que permita estimar las áreas críticas del diseño en cuanto a desempeño o a ingeniería de producto.

II.2.4 REDUCCIÓN DE RUIDO DE “GROUND BOUNCE”

Como es por todos conocido, el ruido de rebote en el lazo común o ruido de “Ground Bounce” se presenta en dispositivos digitales con transiciones rápidas y cuyo comportamiento debe analizarse de manera analógica empleando los conceptos de acoplamiento de impedancias e impedancia distribuida.

Para evitar esta clase de problemas se sugiere tener en cuenta las siguientes soluciones:

- Desacoplamiento de la fuente de alimentación.
- Acoplamiento de impedancias con otros dispositivos externos.
- Separación física y/o desapareamiento de señales que presenten transiciones simultáneas.

II.2.5 CONSIDERACION DE METAESTABILIDAD

Es muy importante considerar no violar los tiempos de **SETUP** (tiempo requerido para que un dato sea estable, antes del flanco de reloj) y **HOLD** (tiempo que se debe mantener el dato para que la salida sea válida) puesto que, de no tomarse en cuenta se puede producir un estado de meta-estabilidad, el cual se puede mantener de acuerdo al tiempo de recuperación de la lógica utilizada y volver a producirse, dándonos como resultado una lectura errónea.

III.

DISEÑO CON FPGA DE XILINX

En el desarrollo del diseño utilizando FPGA's es necesario tomar ciertas consideraciones observadas en el capítulo anterior, así como conocer el funcionamiento del proceso de configuración del FPGA. Para este fin es necesario contar con el paquete de desarrollo que nos permitirá realizar esta tarea. En este capítulo se observará de manera general los pasos a seguir para la obtención del mismo.

El paquete de **XILINX** denominado **XACT**, cuenta con utilerías las cuales nos permiten realizar la captura esquemática, la simulación de dicho circuito, así como la implementación de este en la pastilla; entre las utilerías tenemos el paquete **WORKVIEW**, con el cual se lleva a cabo la captura esquemática y el programa **XDM** el cual nos ofrece la unión de todas las utilerías en un solo ambiente.

III.1 FUNCIONAMIENTO DE XACT

En la figura 3.1 se muestra las tres partes del flujo de diseño de un LCA™, que realiza el sistema de desarrollo XACT™: Captura Esquemática, Implementación de Diseño y Verificación de diseño.



FIGURA 3.1. Metodología de diseño del Sistema de Desarrollo XACT™.

III.1.1 INTRODUCCIÓN DEL DISEÑO

Introducción del Diseño es el proceso que lleva al diseño a un concepto de listado del circuito. Existen varias formas de introducir un diseño, incluyendo esquemáticos, expresiones booleanas o de estados y por medio de lenguajes de descripción de hardware (HDL). Estos métodos de introducción requieren sus propias herramientas para realizar el listado del circuito. El presente proyecto es realizado mediante captura esquemática utilizando las herramientas de **XILINX**.

III.1.2 CAPTURADOR ESQUEMATICO

La captura esquemática puede ser realizada en cualquier paquete **CAD**, ya que existe el soporte por parte de **XILINX** para trabajar con los paquetes comerciales (por ejemplo **OrCad**); el paquete de interfase utilizado para la realización de este proyecto es **WORKVIEW**, por lo que nos enfocaremos a los programas que ofrece este paquete.

VIEW LOGIC

El programa de interfase **VIEWLOGIC**, soporta el sistema **WORKVIEW**, el cual contiene herramientas como **ViewDraw**, **ViewSim**, **ViewWave**.

ViewDraw, es una herramienta de **WORKVIEW** para captura de esquemáticos. En esta se puede introducir un diseño en forma de niveles, de tal manera que puede introducirse un esquemático con la distribución de terminales tanto de entrada como de salida en el nivel principal y en niveles inferiores la lógica interna utilizada.

ViewSim, es un simulador lógico interactivo de **VIEWLOGIC**; el cual nos permite inicializar un archivo con especificaciones de simulación de las señales a visualizar; este incluye a **ViewWave**, un visor y editor de formas de onda de las señales seleccionadas.

La ejecución de las herramientas antes mencionadas se realiza desde ViewLogic de la siguiente forma:

(➤ Indica la selección requerida para ejecutar comando.)

➤ Window ➤ Open ➤ ViewDraw

➤ Window ➤ Open ➤ ViewSim

➤ Window ➤ Open ➤ ViewWave

La lista de mandos utilizadas en los paquetes anteriores son mencionadas en el apéndice A. Una novedad del sistema XACT, son las llamadas **Bibliotecas** unificadas, la cuales utilizan los mismos nombres de símbolos, nombres de terminales, salidas gráficas y funcionalidades, entre las diferentes familias de productos; para poder convertir fácilmente los diseños de una familia a otra. Los nombres de las bibliotecas individuales son XC2000, XC3000, XC4000 y XC7000.

Actualmente se introduce la utilización de XSimMake, un programa que automáticamente corre todos los programas y traductores requeridos para realizar una simulación funcional o de temporización.

Generar diseños con ViewLogic, involucra los pasos que se muestran en la figura 3.3, como metodología de diseño utilizando Viewlogic.

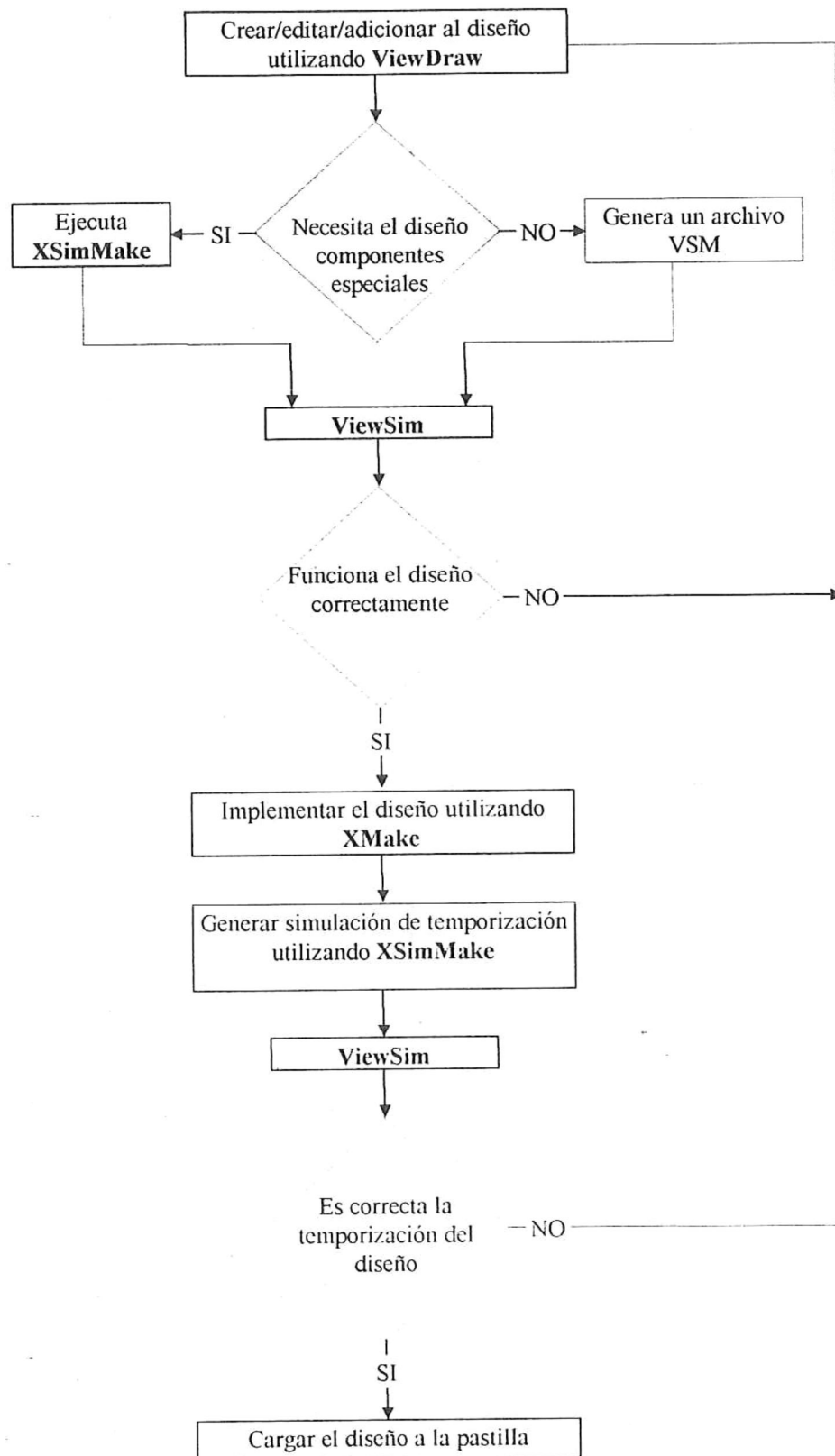


FIGURA 3.3. Metodología de diseño utilizando Viewlogic

III.1.3 IMPLEMENTACION DEL DISEÑO

El proceso de implementación del diseño empieza con la captura de un diagrama a bloques o esquemático. El esquemático - o entrada basada en ecuaciones - es convertido al formato de listado de circuito **XNF** (Xilinx Netlist Format). Una vez en este formato, el diseño es partido en bloques lógicos, se localiza el enrutado más óptimo y posteriormente se selecciona la ruta de interconexión (**Partitioning, Placement and Routing**). Ya completado el diseño, este es documentado en un archivo **LCA** el cual contiene una agrupación de bits (bitstream), los cuales pueden ser enviados a una memoria para programar el dispositivo o ser adecuados a un archivo para ser enviados por un microprocesador.

Los programas para realizar la implementación del diseño utilizados por el sistema de desarrollo **XILINX** son descritos a continuación.

III.1.3.1 XDM, XMAKE

XDM es la interfase computacional con el usuario, donde se accionan todos los programas del sistema de desarrollo **XACT**, utilizados para diseños con **FPGA**.

Como ejemplo de ello tenemos: en el menú de *Introducción de Diseño* al programa **SYMGEN**; en el menu de *Traducción* los programas **XMAKE**, **MAP2LCA**, **MEMGEN**, **XNFMAP**, **XNFMERGE**, **XNFPREP**; en el menú de *Localización y Enrutado* los programas **PPR** y **XDE**; en el menú de *Verificación* los programas **LCA2XNF**, **MAKEBITS**, **MAKEPROM**, **XSIMMAKE**, **XDELAY**, **XCHECKER**, **XNFBA**. Además en los menús de *Utilerías* y *Silueta* los programas de utilerías necesarios para la configuración y el aprovechamiento del sistema.

XMAKE convierte automáticamente un esquemático en archivos **LCA** y **BIT**. En la figura 3.4, se muestra el flujo realizado automáticamente por **XMAKE**, al ser ejecutado desde los diferentes medios como **MSDOS** o **XDM**.

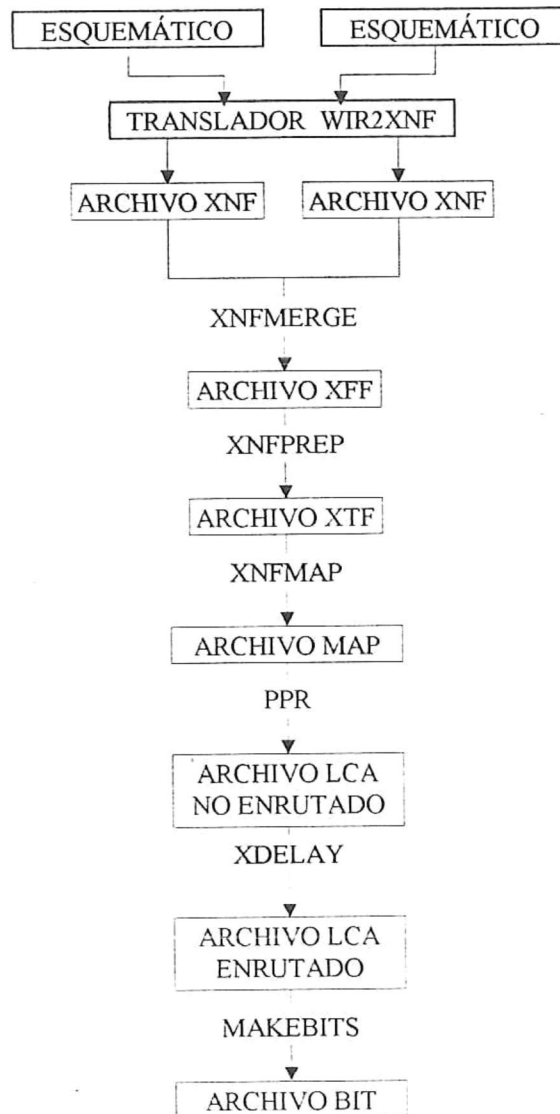


FIGURA 3.4. FLUJO DE OPERACIÓN DE XMAKE

Para trasladar un diseño, XMAKE requiere uno de los cuatro posibles formatos de **archivos de entrada**: archivo esquemático de alto nivel, un archivo de descripción ASCII HDL, un archivo MAK generado previamente por XMAKE o un archivo de alto nivel XNF.

XMAKE crea un número de **archivos de salida** basados en las opciones especificadas, los cuales se describen a continuación.

III.1.3.2 ARCHIVOS GENERADOS POR XMAKE

DESIGN.MAK: Desde un archivo esquemático de entrada, XMAKE crea un archivo MAK (DESIGN.MAK) que documenta como cada sub-módulo de diseño es procesado, incluyendo las opciones utilizadas en los programas de translación.

DESIGN.OUT: XMAKE utiliza varios programas de translación y translaciones directas de pantalla a un archivo DESIGN.OUT, solo si la opción -O es seleccionada. Este es un archivo de texto ASCII que contiene todas las salidas de pantalla de los programas que XMAKE utiliza. El archivo OUT contiene todos los mensajes de advertencia y error generados durante el proceso de diseño, por lo que hay que revisar hasta determinar que el diseño esté libre de errores.

DESIGN.XFF: XMAKE crea un archivo XNF nivelado, DESIGN.XFF, ejecutando el programa XNFMERGE.

DESIGN.MAP: XMAKE genera un archivo XNF particionado, DESIGN.MAP, ejecutando XNFMAP.

DESIGN.LCA: XMAKE crea un archivo LCA particionado, localizado y enrutado ejecutando PPR.

DESIGN.BIT: XMAKE genera un archivo de agrupación de bits para todos los diseños que se enruten completamente con el PPR. Se puede regresar el archivo BIT a un archivo LCA. Las opciones de configuración del generador de agrupación de bits son determinadas por las opciones colocadas en el archivo XDM.PRO.

XMAKE requiere de archivos de entrada para realizar el proceso de diseño y archivos de salida para generar resultados y operaciones.

Para configurar las opciones del flujo desarrollado por XMAKE, se tienen una variedad de opciones de ejecución a las que se pueden acceder con gran facilidad.

Para que XMAKE genere el archivo .BIT, necesita generar paso a paso los archivos descritos anteriormente o utilizar un archivo MAK donde se localizan la serie de objetivos a lograr (con opciones y comandos). Cada objetivo definido especifica que tipo de archivo (*archivo_objetivo*) es generado, basados en los cambios del *archivo_dependiente*, y de los programas (*comandos*) utilizados.

III.1.3.3. XNFMERGE, XNFPREP, XNFMAP, MAP2LCA, PPR, MAKEBITS, MAKEPROM.

XNFMERGE

XNFMERGE une archivos juntos XNF para formar un simple archivo XFF, y genera los parámetros para obtener el desempeño del sistema de desarrollo XACT. Para configurar las funciones desarrolladas por XNFMERGE, se tienen una variedad de opciones de ejecución a las que se pueden acceder. Requiere archivos de alto nivel de entrada tipo XNF. Genera el archivo XFF donde localiza el diseño nivelado y el archivo MRG que es un reporte del XNFMERGE donde escribe los archivos leídos, las señales de alerta y error, y varias opciones más.

XNFPREP

XNFPREP realiza un análisis de las condiciones de alambrado (DRC) de un archivo nivelado XNF y elimina lógica no usada y/o redundante desde un archivo XNF nivelado. También analiza la sintaxis de los parámetros de XACT encontrados en el diseño y prepara

información de retardos para un posterior análisis. Así mismo, genera mensajes de alerta y error que conciernen a los problemas encontrados.

El archivo de entrada utilizado por XNFPREP es el archivo XNF nivelado con extensión *.xnf*. XNFPREP genera un archivo de reporte que lista las alertas, los errores, el diseño lógico y el abanico de salida (fan-out) del reloj. Además incluye un resumen de especificaciones del desempeño de XACT. El nombre por omisión utilizado es *input_design.prp*. En caso de no encontrar errores XNFPREP genera un archivo *input_design.xnf*. XNFPREP genera además un archivo log, llamado *xnfprep.log*, el cual contiene toda la información de salida de la pantalla durante la ejecución de XNFPREP. Para configurar las funciones desarrolladas por XNFPREP, se tienen una variedad de opciones de ejecución a las que se pueden acceder.

XNFMAP

XNFMAP mapea (reduce y particiona) la lógica definida por un archivo XTF (archivo XNF sin errores y sin lógica sin diseñar, generado por XNFPREP) en elementos de LCA como lo son CLB, IOB y TBUF, que pueden ser localizados y enrutados por PPR. El archivo de salida de XNFMAP es un archivo MAP que describe el particionamiento del diseño en elementos de LCA. Este archivo es formateado de diferente forma de acuerdo a la familia que corresponde el diseño. XNFMAP produce también un archivo de reporte de referencia de diseño-a-LCA llamado CRF y un PGF. Estos archivos sirven de guía para conocer el mapeo, que servirán como apoyo para guiar el ruteamiento realizado por PPR. Genera también un archivo PBK, que funciona como respaldo de los anteriores.

El proceso desarrollado por XNFMAP es el siguiente:

- Lee el archivo de diseño de entrada (XTF)
- Lee el archivo guía de diseño, si ya existe (CRF)
- Usa cualquier archivo de diseño CLBMAP de símbolos y el archivo de guía para mapear la lógica.

CAPITULO III

- Mapea la lógica restante en recursos de LCA.
- Crea los archivos de salida

XNFMAP además genera mensajes de alerta y de error durante su procesamiento, que sirven para la corrección de las posibles fallas.

Para configurar las funciones desarrolladas por XNFMAP, se tienen una variedad de opciones de ejecución a las que se pueden acceder.

MAP2LCA

MAP2LCA translada un archivo de entrada MAP a uno LCA para ser introducido a un programa de localización y ruteamiento automático como PPR.

MAP2LCA genera un archivo de salida LCA que describe como el diseño es particionado en un particular dispositivo LCA. MAP2LCA produce un archivo SCP, que contiene la localización y el ruteamiento especificado en el esquemático; también puede producir un archivo AKA que contiene patrones jerárquicos de nombres para símbolos, señales, y abreviaciones asociadas.

MAP2LCA genera mensajes de alerta y de error durante su procesamiento, que sirven para la corrección de las posibles fallas. Para configurar las funciones desarrolladas por MAP2LCA, se tienen una variedad de opciones de ejecución a las que se pueden acceder.

PPR

PPR particiona, localiza y enruta automáticamente diseños de las familias XC3000A/L, XC3100A, XC4000 y XC4000A/H. La figura 3.5 muestra su flujo de operación.

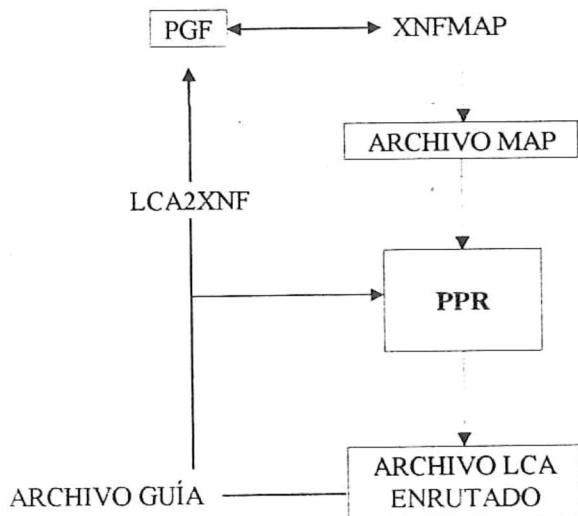


FIGURA 3.5. FLUJO DE OPERACIÓN DE PPR

Permite la opción de retornar la temporización de un diseño ya particionado, localizado y enrutado, y subsecuentemente cargado.

El archivo de entrada de diseño en un archivo XTF generado por XNFPREP o un archivo MAP generado por XNFMAP. Su archivo de salida es un archivo de diseño LCA que es utilizado por MAKEBITS, y puede servir además como una guía para volver a ejecutar PPR. PPR genera archivos de salida como los que a continuación se describen. *design.lca*, es un archivo LCA localizado y enrutado.

DESIGN.LCB es un respaldo del anterior archivo LCA.

PPR.LOG contiene la información de salida de pantalla cuando PPR es ejecutado.

DESIGN.RPF es un archivo reporte generado en los primeros pasos de enrutado.

DESIGN.RPT es un archivo reporte que incluye estadísticas de diseño y utilización de dispositivo, información de programas ejecutados, localización de las terminales de I/O y otra información. Esta información es útil para analizar el diseño desarrollado.

PPR por su gran aplicación e importancia en el proceso de diseño, tiene una gran variedad de opciones que simplifican y refuerzan el trabajo que realiza.

MAKEBITS

MAKEBITS crea la agrupación de bits para un diseño LCA. La agrupación de bits describe las funciones internas lógicas y las interconexiones de un dispositivo LCA. MAKEBITS provee comandos para generar agrupaciones de bits de configuración.

Para realizar su función, MAKEBITS requiere de un archivo LCA, que contenga el diseño particionado en una arquitectura LCA.

MAKEBITS genera varios archivos de salida:

DESIGN.BIT es un archivo binario que contiene los datos de configuración para un diseño LCA.

DESIGN.LL es un archivo creado con la opción MAKELL, que lista las posiciones de las celdas de las salidas de los Flip-Flop, CLB en la agrupación de bits.

DESIGN.MBO es un archivo generado automáticamente cuando MAKEBITS genera la agrupación de bits. El archivo MBO contiene mandos de configuración que graban los estados de todas las opciones de configuración, en el tiempo que la agrupación de bits es creada.

DESIGN.RBT es un archivo creado que es una versión ASCII del archivo de agrupación de bits de configuración.

DESIGN.MSK es un archivo necesitado para leer otra vez los datos de configuración de un dispositivo LCA funcionando.

DESIGN.LCA es un archivo donde se almacena el diseño LCA con este nuevo nombre, para ser utilizado para la simulación.

MAKEBITS tiene una variedad de opciones dependiendo de la familia con la que se esta trabajando, que refuerzan el trabajo que realiza.

MAKEPROM

MAKEPROM da formato a un archivo de agrupación de bits de configuración (BIT) creando un archivo que puede ser cargado a una PROM. MAKEPROM también combina multiples archivos BIT para utilizarlos en una cadena de dispositivos LCA. El archivo PROM contiene datos de configuración del dispositivo LCA. MAKEPROM convierte el archivo BIT en uno de los siguientes tres formatos: MCS-68 (Intel), EXORMAX (Motorola), o TEKHEX (Tektonix).

MAKEPROM crea dos archivos; un archivo PROM que contiene la información de configuración y un archivo imagen PROM que contiene un mapa de memoria del archivo PROM generado. El archivo PROM tiene la extensión .mcs, .exo o .tek dependiendo del formato seleccionado. El archivo imagen tiene la extension PRM.

Debido a lo crítico de este paso final del diseño, se tienen varias opciones para variar las características del formato.

III.1.4 VERIFICACION DE DISEÑO

La VERIFICACION DE DISEÑO incluye simulación, análisis de temporización estática y verificación en circuito. La simulación es realizada en herramientas soportadas por Xilinx. La entrada a esas herramientas requiere una traducción a un archivo específico de la herramienta desde un archivo XNF. Las herramientas del análisis de temporización estática y la verificación en circuito son parte del sistema de desarrollo de Xilinx.

Los programas que utiliza el sistema de desarrollo XACT, para realizar la verificación del diseño se describen a continuación.

III.1.4.1 XDELAY, LCA2XNF, XNFBA, XDE.

XDELAY

El programa XDELAY es una herramienta analizadora de temporización estática que reporta información detallada de la temporización de retardo acerca de un diseño de LCA enrutado y puede ser utilizado por cualquier análisis de alto desempeño. XDELAY puede crear una lista detallada de patrones y proveer un análisis de diseño, incluyendo estimaciones sobre el desempeño del sistema para cada reloj en el diseño.

El programa XDELAY tiene varias interfaces con una gran variedad de funciones y aplicaciones a realizar, así como una variedad de opciones dependiendo de la familia utilizada y los modelos de temporización.

LCA2XNF

El programa LCA2XNF convierte un archivo LCA a un archivo en formato Xilinx de listado de circuito (XNF), el cual puede ser utilizado en la simulación funcional o en una simulación de temporización. Si el archivo de entrada LCA está debidamente localizado y enrutado, el archivo XNF resultante contendrá todos los retardos en el peor de los casos para la simulación de temporización.

LCA2XNF requiere de un archivo LCA como entrada, y genera un archivo XNF y opcionalmente un archivo SPC. El archivo de diseño LCA de entrada es creado por PPR o por XDE, y puede estar completamente, parcialmente o sin enrutar. La extensión LCA es opcional. El archivo de salida del LCA2XNF es un archivo XNF con la temporización

anotada. El archivo de salida SPC es un archivo de texto que lista las especificaciones de temporización para el tipo del dispositivo seleccionado así como de su grado de velocidad; es un archivo opcional generado con la opción -S.

LCA2XNF tiene una variedad de opciones para configurar el proceso realizado. LCA2XNF genera mensajes de alerta y de error, utilizados para el análisis del diseño y la corrección de fallas.

WIR2XNF

Se necesita traducir el diseño en un archivo XNF antes de poder hacer una implementación o una simulación del diseño. El programa WIR2XNF es usado para generar el archivo de formato nivelado de XNF (XFF) de los archivos del subdirectorío de VIEWDRAW. Usa los archivos WIR del subdirectorío de VIEWLOGIC como archivos de entrada y los convierte en sus correspondientes archivos XNF. WIR2XNF acepta como entrada el archivo de VIEWDRAW *design.1* localizado en el subdirectorío wir. WIR2XNF crea un archivo XNF de alto nivel con el mismo nombre que el archivo de entrada.

XDE

XDE (Editor de Diseño de Xilinx) es una interfase computacional con el usuario que permite al diseñador editar y analizar manualmente un diseño LCA. XDE contiene varios programas que se describen a continuación, que permiten hacer cambios en un diseño existente, o crear un nuevo diseño. EDITLCA es un editor gráfico a nivel de dispositivo para los diseños en LCA. DRC es un analizador de condiciones eléctricas de los diseños de LCA a nivel de dispositivo. MAKEBITS es un programa que genera agrupación de bits de un diseño de LCA que se pueden cargar a un dispositivo. MAKEPROM es un programa que reformatea los datos de las agrupaciones de bits para ser programadas en memorias PROM.

CAPITULO III

Debido a su gran versatilidad de aplicaciones tiene una gran variedad de funciones y opciones para realizarlas.

IV.

DISEÑO DE LA TARJETA

Como se mencionó anteriormente (Capítulo II), el sistema a implementar es un contador de frecuencia en forma directa, lo cual se realiza utilizando el esquema de contadores de alta velocidad; este sistema es dividido en 2 contadores, un pre-escalador (fi.g 4.1) y un síncrono (fig.4.2); el circuito pre-escalador es adoptado debido a las frecuencias de trabajo del estimador. La utilización de este, nos hace obtener una cuenta la cual difiere de la original, pero puede ser conocida por medio de la tabla 4.I.

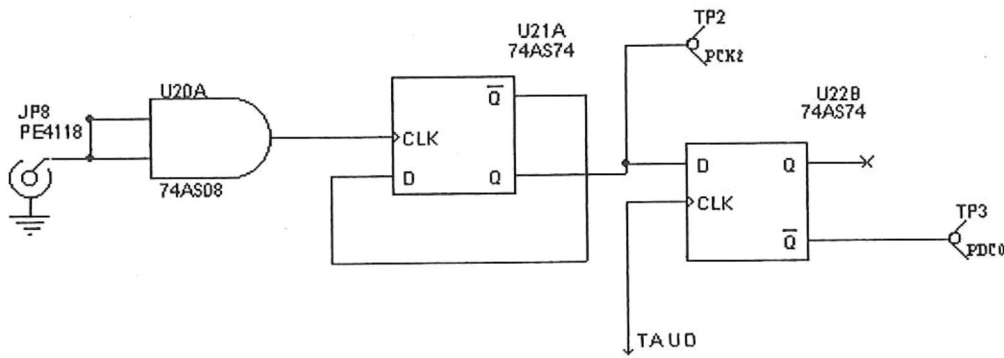
$(DC2 \ DC1 \ DC0)_t$	$(PDC2 \ PDC1 \ PDC0)_{t+1}$
0 0 0	1 1 1
0 0 1	0 0 0
0 1 0	0 0 1
0 1 1	0 1 0
1 0 0	0 1 1
1 0 1	1 0 0
1 1 0	1 0 1
1 1 1	1 1 0

Tabla 4.I

donde

$()_t$ representa el dato en el contador y

$()_{t+1}$ representa el dato en el cerrojo



Fi.g 4.1 Contador pre-escalador (circuito parcial)

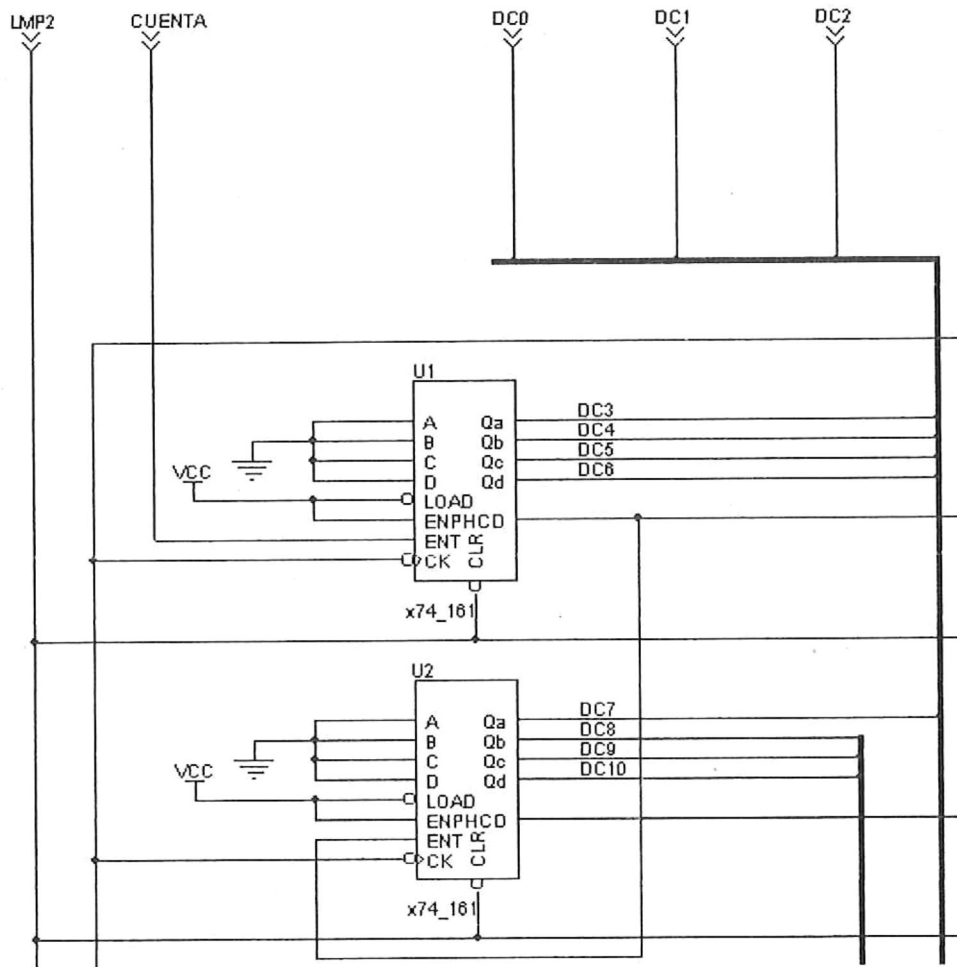


Fig.4.2 Contador síncrono (circuito parcial)

Dado que el contador asincrónico cuenta en forma descendente, al comenzar en 000 el siguiente estado será 111, y dado que el bit más significativo de este contador es también el reloj del contador síncrono se producirá una cuenta de 8 como desbalance. Además, dado que el proceso de captura de la cuenta, asincrónica y síncrona es grande, se

presentan pulsos al contador síncrono entre la sincronización del mando TAU y la edición del mando TAU que deberán eliminarse de la cuenta atrapada. Por lo anterior es necesario restar 8 a todas las cuentas y adicionar 8 a todas aquellas cuentas que no presenten 111 en el cerrojo asincrónico para de esta manera determinar la cuenta real. El cerrojo (fig. 4.3), es utilizado para almacenar temporalmente el dato al obtener un flanco ascendente del mando de captura, lo cual asegura un dato estable al momento de ser almacenado.

En el circuito síncrono son implementados los 29 bits restantes y este emplea como señal de reloj, la cadencia de la última etapa del contador pre-escalador. Para optimizar el proceso de cuenta se implementa un acelerador de acarreo. Este es utilizado con el fin de reducir el retardo de propagación del acarreo.

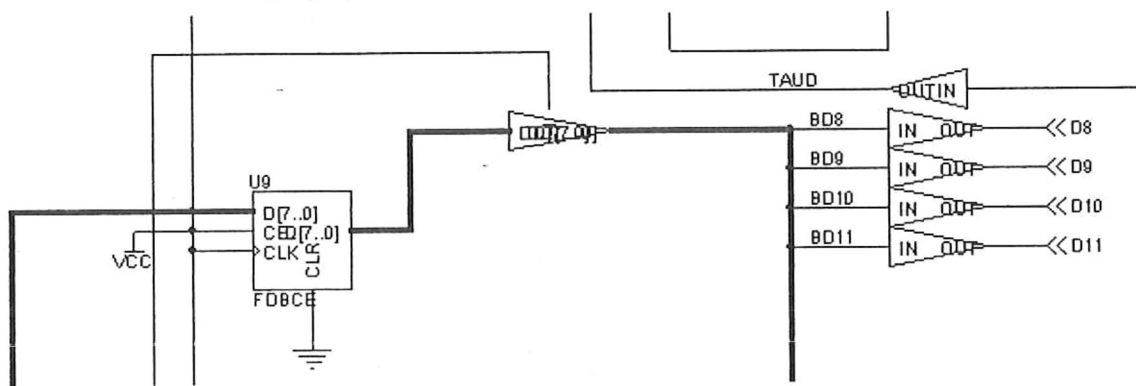


Fig. 4.3. Cerrojo (circuito parcial)

Una vez analizado el circuito propuesto (capítulo II), se procede a implementar el circuito en FPGA tomándose en cuenta las consideraciones especificadas.

Para la implementación del diseño, es necesario realizar la captura esquemática del circuito propuesto, utilizando las herramientas del paquete XACT descritas en el capítulo anterior.

IV.1 CAPTURA ESQUEMATICA

Dado que la captura esquemática se puede dar en forma jerárquica, el diagrama para implementación en FPGA se integra por niveles, quedando de la sig. manera:

- ◆ Nivel 1 Esquema del circuito FPGA (terminales de entrada y salida) fig. 4.4
- ◆ Nivel 2 Esquema del contador síncrono, cerrojos y reloj de contador fig. 4.5
- ◆ Nivel 3 Esquema del generador de acarreo fig. 4.6

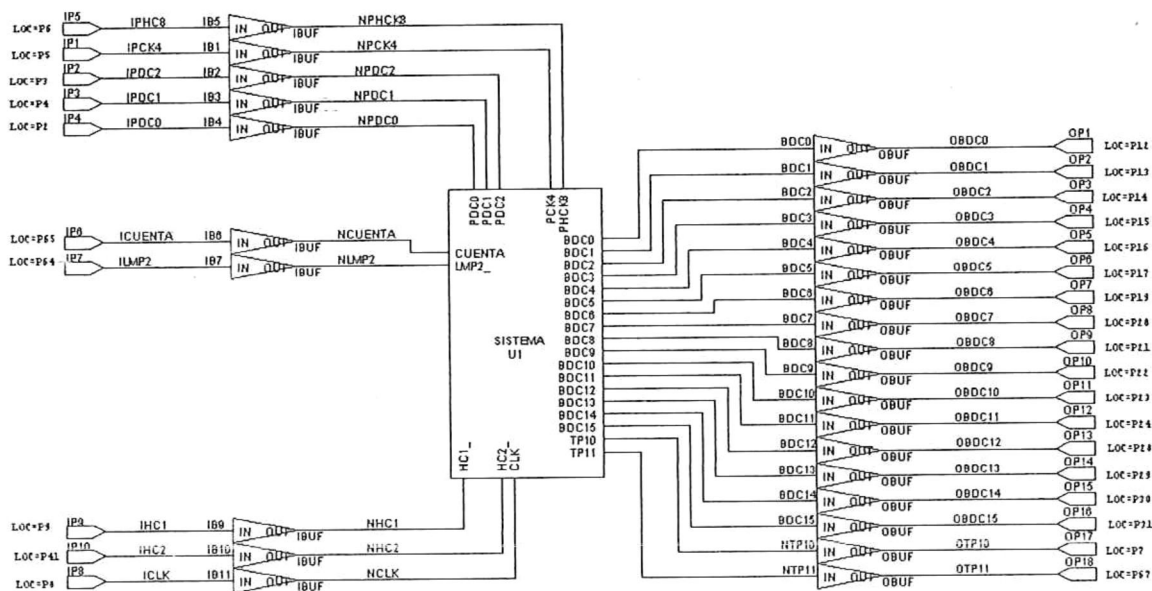


Fig. 4.4. Circuito FPGA (terminales de entrada y salida)

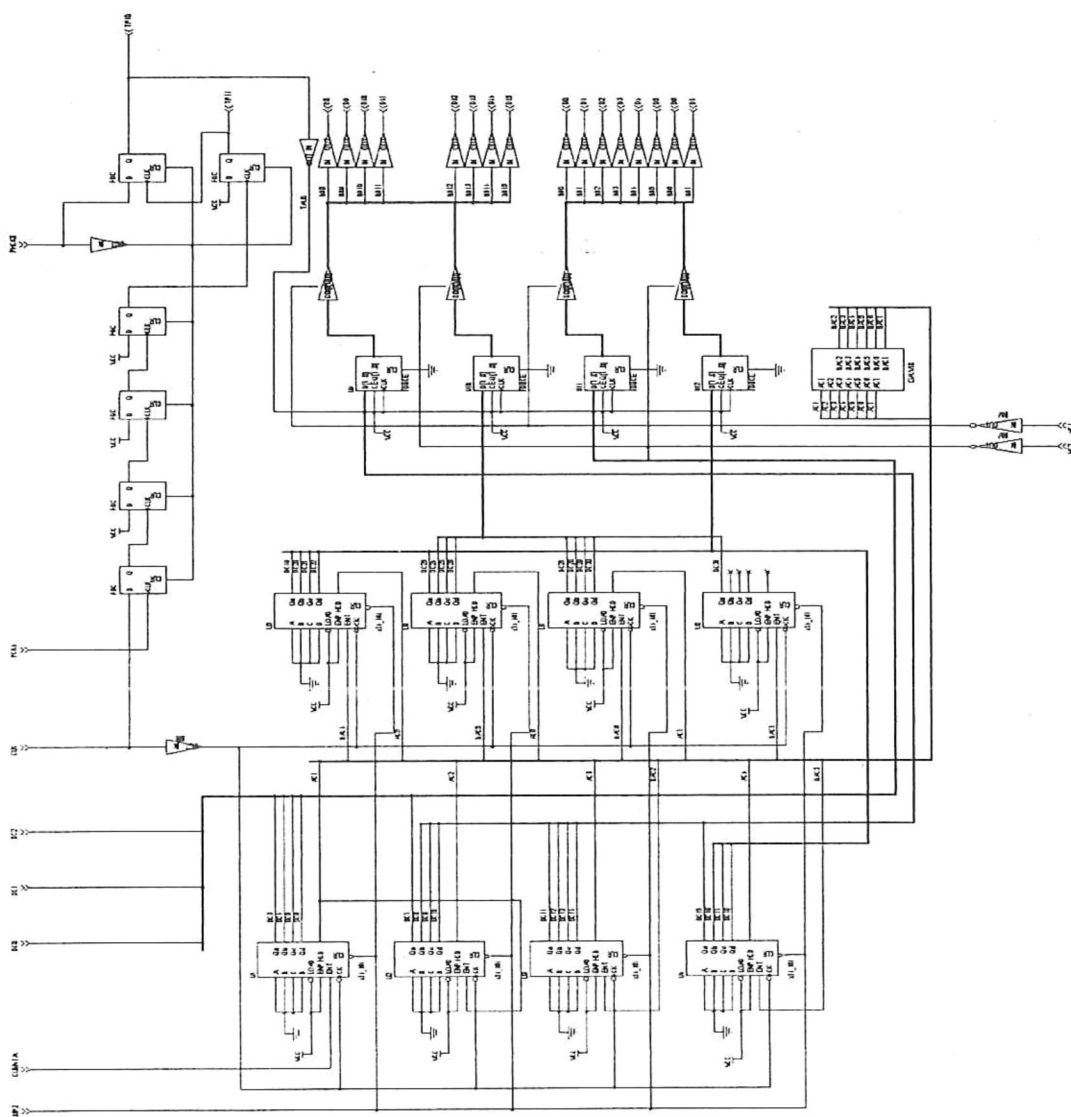


Fig 4.5 Contador síncrono, cerrojos y reloj de contador

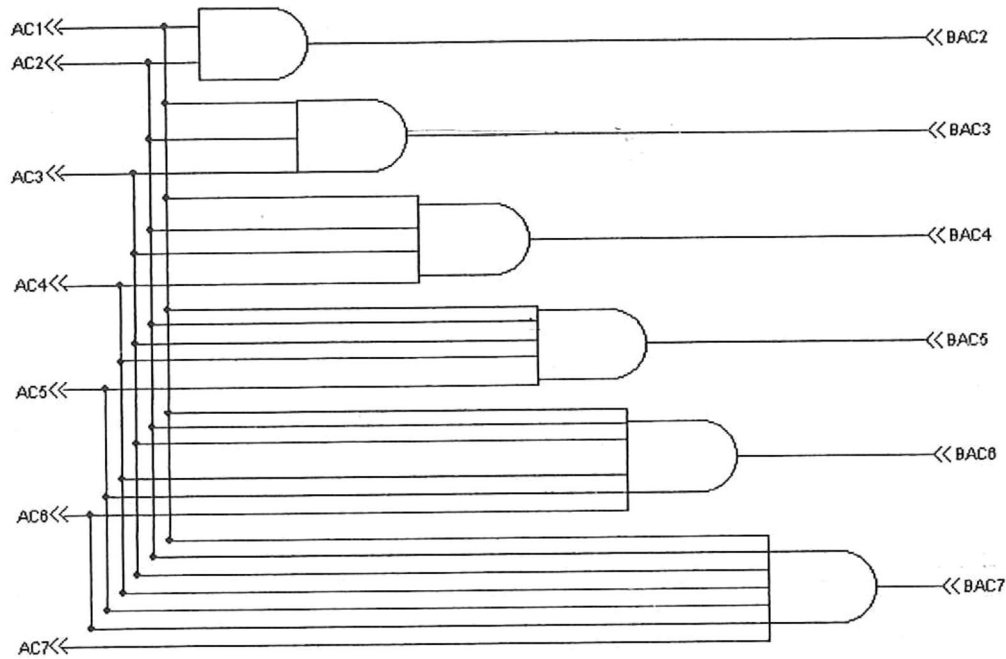


Fig. 4.6 Generador de acarreo

La tabla 4.II muestra la distribución de las terminales y las señales asignadas a cada una de estas.

PIN	SEÑAL	PIN	SEÑAL	PIN	SEÑAL	PIN	SEÑAL
1	GND	18	VCC	35	GND	52	VCC
2	PDC0	19	BDC6	36	NC	53	BD3
3	PDC2	20	BDC7	37	NC	54	GND
4	PDC1	21	BDC8	38	NC	55	BD2
5	PCK4	22	BDC9	39	NC	56	BD1
6	PHCK8	23	BDC10	40	NC	57	TP14
7	TP10	24	BDC11	41	/HC2	58	BD0
8	CLK	25	M1	42	NC	59	NC
9	/HC1	26	M0	43	NC	60	NC
10	VCC	27	M2	44	/RESET	61	/BIOW
11	NC	28	BDC12	45	DONE,/PG	62	VCC
12	BDC0	29	BDC13	46	BD7	63	NC
13	BDC1	30	BDC14	47	TAUD	64	/LMP2
14	BDC2	31	BDC15	48	BD6	65	CUENTA
15	BDC3	32	NC	49	BD5	66	NC
16	BDC4	33	NC	50	GND	67	TP11
17	BDC5	34	INIT	51	BD4	68	NC

Tabla 4.II

Una vez realizada la captura esquemática y tomando en cuenta las consideraciones de temporización, se procede a elaborar el archivo que genera la configuración de circuito FPGA.

IV.2 CONFIGURACION DEL FPGA

Para generar el archivo de configuración, es necesario indicar los pasos a realizar uno por uno (como se vió en funcionamiento de XMAKE en capítulo anterior), o utilizar la opción de que se genere el archivo, que nos permite realizar estos pasos de manera automática; este archivo es conocido como .MAK, el cual se muestra a continuación:

```
#
# Created by XMAKE Version 5.1.0 on Wed Sep 11 12:36:26 1996
#
# The following options were used: -G
#
# The following is the hierarchy of the design 'sch\allan.1'
#
# sch/allan.1
#   sch/sistem.1
#     sch/galv18.1
#
DEFAULT_TARGET allan.bit
:
allan.bit : allan.lca
            makebits -R2 -S0 -XB -YA allan.lca

allan.lca : allan.cst allan.map
            ppr allan.map parttype=3020APC68-7
            xdelay -TIMESPEC -D -W allan.lca

allan.map : allan.xtf
            xnfmap -P 3020APC68-7 allan.xtf allan.map

allan.xtf : allan.xff
            xnfprep allan.xff allan.xtf parttype=3020APC68-7

allan.xff : xnf\galv18.xnf xnf\sistem.xnf xnf\allan.xnf
```

```
xnfmmerge -A -D xnf -D . -P 3020APC68-7 xnf\allan.xnf allan.xff
```

```
xnf\allan.xnf : sch\galv18.1 sch\sistem.1 sch\allan.1  
wir2xnf -B -OD xnf allan allan.xnf
```

El funcionamiento de cada uno de ellos se explicaron en el capítulo anterior. Una vez creado el archivo de configuración .BIT, es necesario vaciar este archivo a la pastilla y observar si fué configurada adecuadamente, lo cual de antemano puede saberse ya que podemos observar una simulación con otra de las herramientas de XILINX (**ViewWave**).

IV.2.1 SIMULACION

El circuito implementado puede ser simulado como se ha mencionado, con la herramienta de XILINX **ViewWave**; para este fin es necesario crear el archivo de conexiones .VSM, utilizado por el programa **ViewSim**.

Para llevar a cabo este fin se selecciona el esquemático generado por XSIMMAKE (descrito en el capítulo anterior) en el editor de esquemáticos de **WORKVIEW** (**VIEWDRAW**) de la siguiente manera:

Windows ➤ Open ➤ Utility ➤ Workview ➤ Check ➤ Project

posteriormente se seleccionan el conjunto de nodos a ser simulados y se genera el archivo de conexión con el paquete simulador de la siguiente manera:

Windows ➤ Open ➤ Viewsim

ya generado este archivo, se introducen las condiciones de simulación o se ejecutan de un archivo. cmd previamente generado, ejecutándolo de la siguiente manera:

Execute ➤ CmdFile

dadas las condiciones de simulación, se ejecuta el paquete editor de ondas en el cual pueden ser observadas las señales previamente seleccionadas; para este fin se ejecutan los comandos de **WORKVIEW** de la siguiente manera:

Windows ➤ Open ➤ ViewWave ➤ Viewsim

Una vez obtenido el enlace se puede observar el funcionamiento del circuito, tomándo en cuenta las consideraciones de tiempo.

IV.2.2 CONFIGURACION

Para la configuración de la pastilla se realiza un programa el cual convierte el archivo generado por el sistema a formato binario, para enviar el archivo de configuración a esta por medio del ducto. El archivo utilizado es de la extensión .MCS el cual se encuentra en formato ASCII y el programa para la traducción de los datos es generado en lenguaje C el cual se muestra a continuación :

```
/* PROGRAMA QUE LEE UN ARCHIVO EN FORMATO ASCII Y LO CONVIERTE A FORMATO
BINARIO PARA CONFIGURAR EL INTEGRADO XILINX DE SPTRVA*/
```

```
#include<stdio.h>
#include<alloc.h>

#define XILINX 0x314          /* HABILITA U19 DE SPTRVA XILINX*/
#define LATCH_6 0x318        /* HABILITA U17 DE SPTRVA XILINX*/

unsigned char far ahex(unsigned char palta, unsigned char pbaja);
unsigned char far * archdat;
unsigned char far * memoria;
FILE *pin;

/*PAGE*/
main()
{ /* principio de programa */
unsigned char ndatos,tipodesarta=0,chksum=0,dato;
unsigned int i,k,direcc, imem, iRXD,tiRXD, tdatos;
int j;
clrscr();
pin=fopen("allan.mcs","rt"); /* asigna archivo de entrada test.hex */
if(pin==NULL){ /* pide archivo si no existe archivo de entrada */
printf("proporciona el nombre del archivo con extensión .mcs");
exit(1);}
archdat=(unsigned char far *)farmalloc(16834);/* asigna espacio para datos de 64kb */
memoria=(unsigned char far *)farmalloc(2048);/* asigna espacio para memoria de 32kb */
if(archdat==NULL||memoria==NULL){
printf("No hay suficiente memoria");
exit(1);}
tdatos=0; /* contador del total de datos del archivo */

do{
j=fgetc(pin); /* lee datos de archivo ASCII y almacena en memoria */
archdat[tdatos++]=j;
}while(j!=EOF);
```

```

fclose(pin);          /* cierra archivo leído */

iRXD=0; /* inicializa apuntadores */
imem=0;
tdatos=0;
/* rutina para convertir los datos en ASCII a BINARIO */
while(tipodesarta!=0x01){ /* diferente de Record de final de archivo */
    chksum=0;          /* inicializa el checksum */
    if(archdat[iRXD]!=':') /* ignora el 1er. car cter */
        return(1);
    ndatos=ahex(archdat[iRXD+1],archdat[iRXD+2]); /* convierte los 2 bytes iniciales a número de
datos en hex */
    chksum+=ndatos; /* iguala checksum al número de datos */
    direcc=dato=ahex(archdat[iRXD+3],archdat[iRXD+4]); /* convierte 2 bytes sig. a la dirección en hex
*/
    chksum+=dato; /* suma el dato al checksum */
    direcc<<=8; /* acomoda como bytes más significativos de direccionamiento */
    dato=ahex(archdat[iRXD+5],archdat[iRXD+6]); /* convierte los sig. 2 bytes de la dirección */
    chksum+=dato; /* suma el dato al checksum */
    direcc|=dato; /* obtiene la dirección real de 1er. dato */
    tipodesarta=ahex(archdat[iRXD+7],archdat[iRXD+8]); /* obtiene el tipo de direccionamiento */
    iRXD+=9;
    chksum+=tipodesarta; /* suma el dato al checksum */
    tiRXD=iRXD;
    for(i=0;i<ndatos;i++){ /* calcula el checksum para todos los datos leído */
        chksum+=ahex(archdat[iRXD],archdat[iRXD+1]);
        iRXD+=2;}
    dato=ahex(archdat[iRXD],archdat[iRXD+1]); /* convierte el dato a formato hexadecimal */
    if((unsigned char)(chksum+dato)!=0x00) /* verifica el número de datos leídos */
        return(1);
    if(tipodesarta==0x00){ /* verifica si los datos han sido almacenados en formato normal */
        iRXD=tiRXD;
        imem=direcc; /* inicializa apuntador de dirección para almacenamiento de datos */
        for(i=0;i<ndatos;i++){
            memoria[imem]=ahex(archdat[iRXD],archdat[iRXD+1]); /* almacena datos en
formato binario */
            iRXD+=2;
            if(imem>tdatos)
                tdatos=imem; /* indica la última dirección en caso de ser menor */
            imem++;}
        iRXD+=2;}
    if(tipodesarta!=0x01){ /* diferente de Record de final de archivo */
        while(archdat[iRXD]!=':')
            iRXD++;}
} /* fin del ciclo inicial */
printf("programando XILINX ");
for(i=0;i<=tdatos;i++){
    outportb(LATCH_6,memoria[i]); /* envía los datos de configuración del XILINX */
    outportb(XILINX,0Xff); /* habilita integrado */
}
printf(" integrado programado");
getch();
return 0;
} /* fin de programa */

```

```
/*PAGE*/
unsigned char far ahex(unsigned char palta, unsigned char pbaja)
/*-----*/
/* convierte un par de caracteres ASCII en uno hexadecimal */
/*-----*/

{
    palta-=0x30;
    pbaja-=0x30;
    if(palta>0x09)
        palta-=0x07;
    if(pbaja>0x09)
        pbaja-=0x07;
    palta<<=4;
    return(palta|pbaja);
}
```


V.

PRUEBAS Y RESULTADOS

En este capítulo se observan las pruebas realizadas a la implementación del circuito para verificar la funcionalidad del diseño. Como podemos recordar el resultado del desempeño de este puede ser conocido por medio de las herramientas proporcionadas por el paquete de programación del FPGA, como son **VIEW SIM** y **VIEW WAVE** por lo que este capítulo solo viene a ser la confirmación del circuito anteriormente analizado.

Para la realización de las pruebas se utiliza un Analizador de Estados Lógicos/Osciloscopio HP 1660as y el programa "XILTEST.C", escrito en lenguaje C de Borland V2.0. Para cada prueba se especifican la conexiones necesarias entre el analizador y el procesador SPTRVA. Los esquemáticos respectivos se anexan en el apéndice C.

V.1 VERIFICACIÓN DE DATOS EN EL LATCH6

En esta prueba se realiza un envío de datos para asegurarse de la manipulación de las líneas $\overline{DONE/PROG}$ y $\overline{RESET}$, a través de las líneas PGR y $\overline{RESET}$ respectivamente así como el resto de las líneas presentadas en el LATCH6 referenciado como U17 de la tarjeta SPTRVA. Las terminales del POD1 del analizador de estados lógicos son alambradas como se muestran en la tabla 5.I.

terminal en C.I. U17	Nombre de la señal	terminal del POD1
11	/CS6	8
12	PGR	7
13	/RS	6
14	/RT	5
15	/RESET	4
16	CUENTA	3
17	/LMP2	2
18	/LMP3	1
19	/HINT	0

Tabla 5.I

Para la adquisición de los datos el analizador es disparado con el flanco ascendente de /CS6. En esta prueba el programa XILTEST.C repite un lazo infinito de escritura al cerrojo LACTH6 en el cual se envían los siguientes estados:

bit	7	6	5	4	3	2	1	0
señal	PGR	/RS	/RT	/RESET	CUENTA	/LMP2	/LMP3	/HINT
estado	H	H	H	H	L	L	L	H

La figura 5.1 muestra la captura en el analizador de los datos enviados por el programa

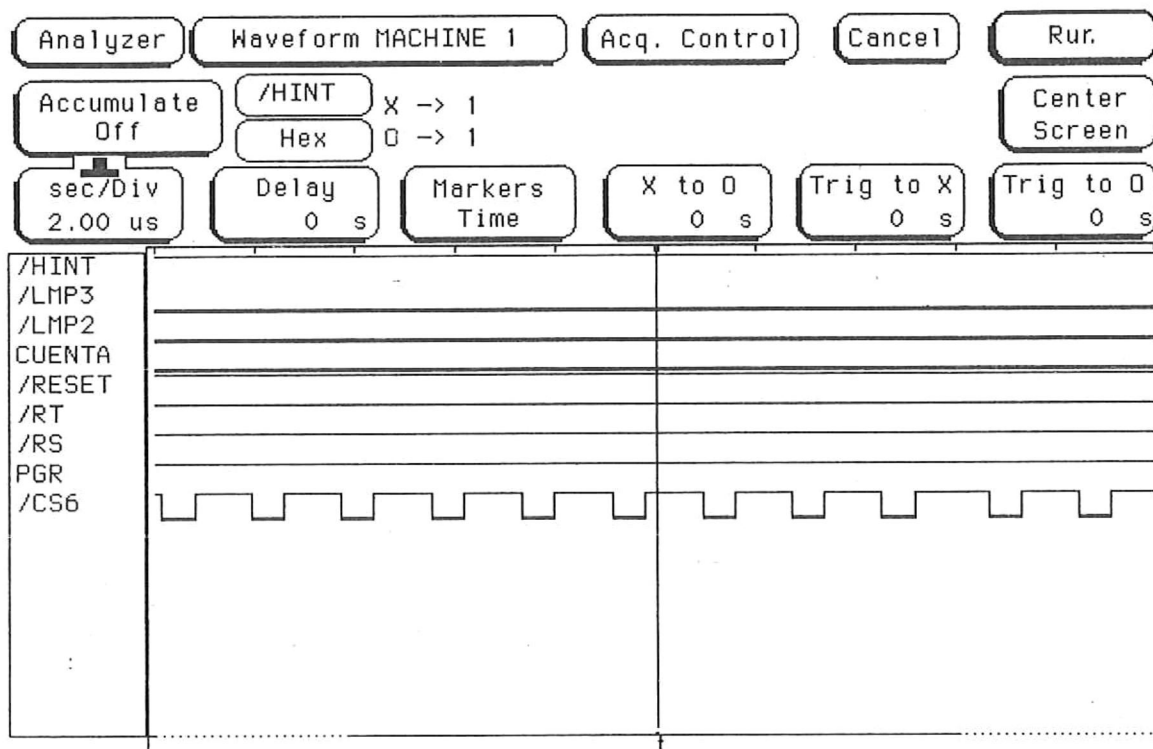


Figura 5.1. Diagrama de tiempos que muestra el comportamiento del LATCH6

V.2 VERIFICACIÓN DE ALIMENTACIÓN, OBSERVACIÓN DE LA FASE DE INICIALIZACIÓN Y LIMPIA DEL INTEGRADO

Esta prueba es realizada para observar la condición de monotonicidad y tiempo al momento de alimentar el FPGA. Para esto, solo es necesario observar con el osciloscopio alguna terminal de alimentación del FPGA y la terminal correspondiente a la señal $\overline{INIT}$ (terminal 41 o punto de prueba TP13 en la tarjeta SPTRVA). La figura 5.2 muestra el comportamiento de la fuente de alimentación la cual cumple con las condiciones antes mencionadas; el voltaje presenta una razón creciente y tarda menos de 25ms en ir de 2.0 V a VCC. También se observa que la señal $\overline{INIT}$ actúa adecuadamente indicando las fases de inicialización y limpia.

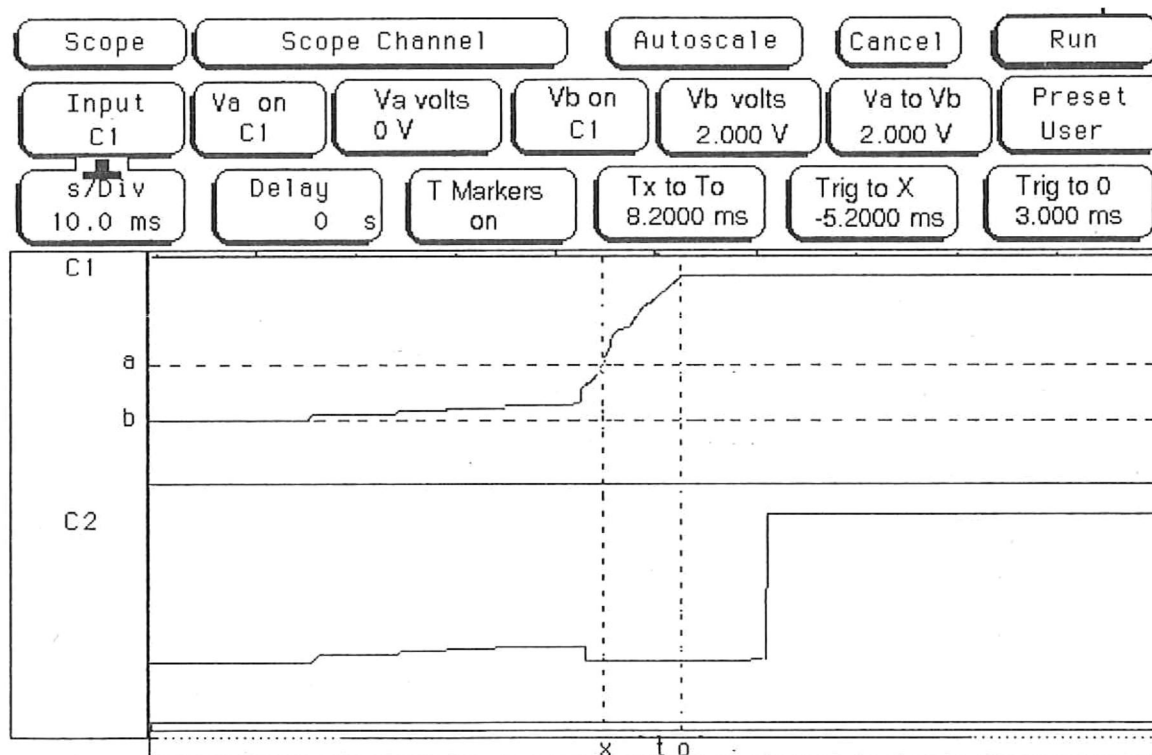


Figura 5.2. Comportamiento de VCC (C1) y señal de $\overline{INIT}$ desde el momento de encendido de la fuente (C2).

V.3 FASE DE CONFIGURACIÓN DE LA PASTILLA

En esta prueba la cual se divide en dos, se observa el proceso de inicialización de configuración de la pastilla, así como el fin de este proceso. La primera se relaciona con la fase de inicialización y limpia del XILINX a través de la observación de la señal $\overline{INIT}$; mientras que la segunda está relacionada con la fase configuración mediante la observación de la señal $DONE/\overline{PROG}$.

Para la realización de ambas pruebas, el conector del analizador de estados lógicos es conectado como se muestra en la tabla 5.II.

terminal del C.I. o punto de prueba	Nombre de la señal	terminal del POD1
20 de U18	/CS5	10
3 de U18	/BIOW	11
15 de U17	/RESET	12
18 de U13	BD0	0
17 de U13	BD1	1
16 de U	BD2	2
15 de U	BD3	3
14 de U	BD4	4
13 de U	BD5	5
12 de U	BD6	6
11 de U	BD7	7
TP14	RDY	8
TP13	/INIT	9
TP16	$DONE/\overline{PROG}$	13

Tabla 5.II

En la figuras 5.3 y 5.4 se muestran la señales observadas en el proceso de configuración del FPGA.

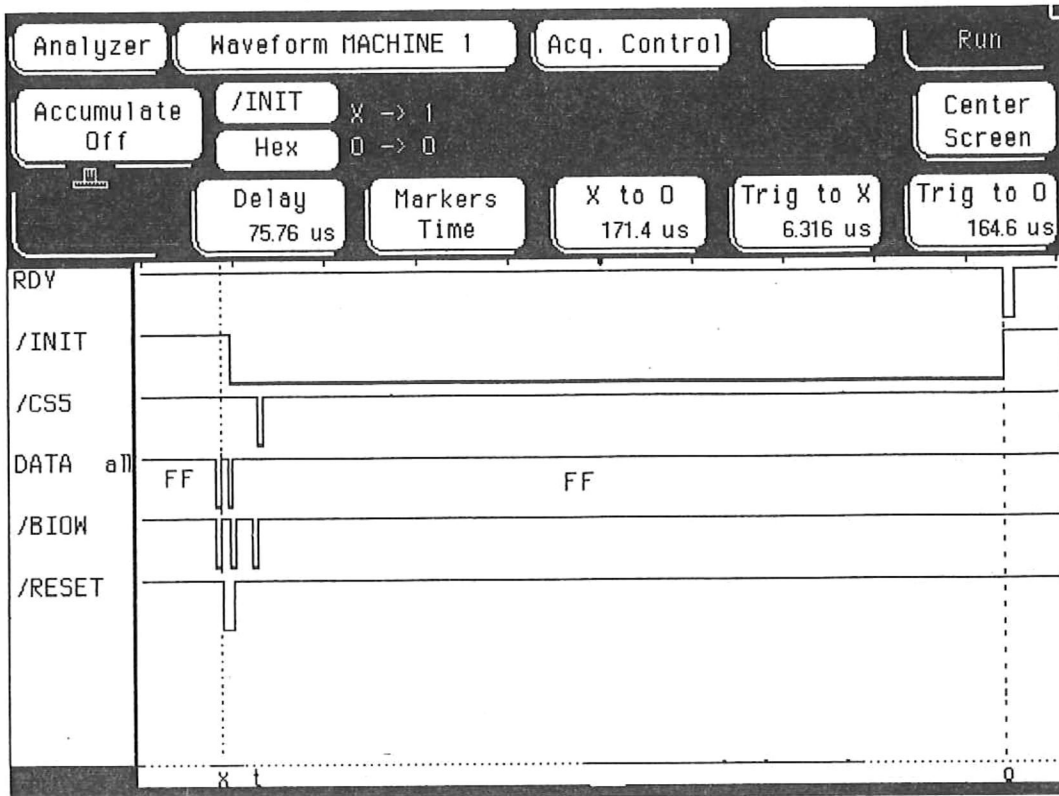


Fig. 5.3 Fase de inicialización configuración del FPGA

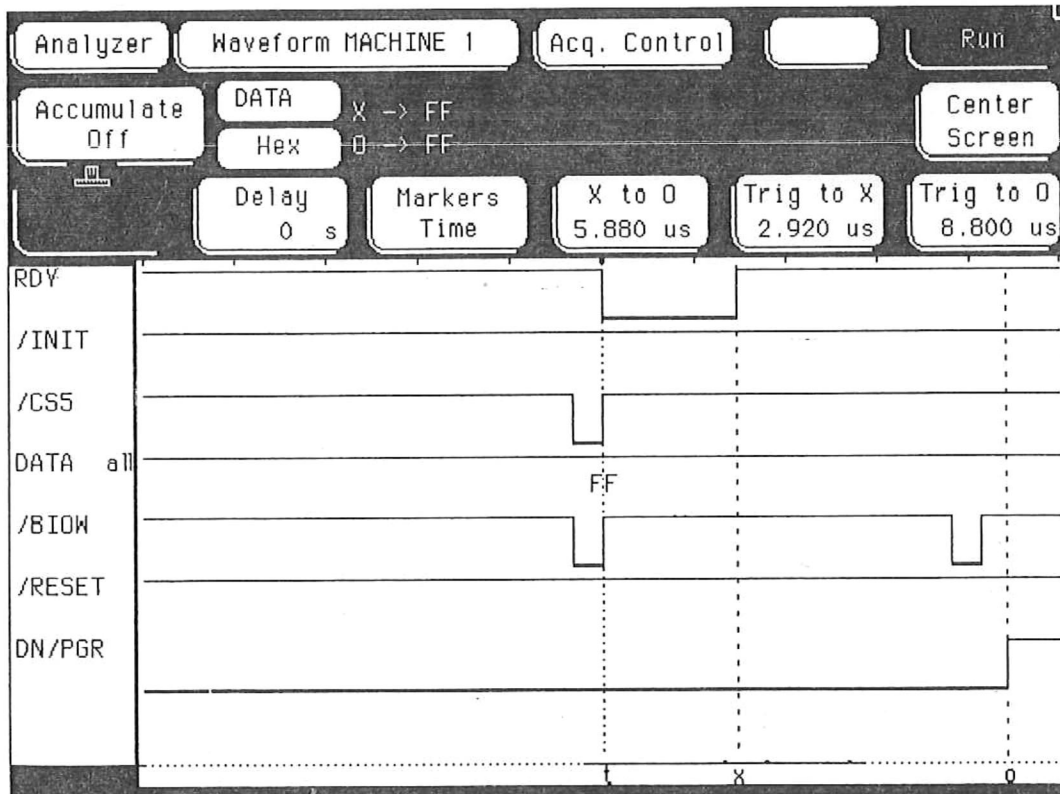


Fig. 5.4 Fase de fin de configuración del FPGA

V.4 VERIFICACIÓN DEL FUNCIONAMIENTO DEL CONTADOR Y CERROJOS ASINCRÓNICOS.

En esta prueba se pretende observar y comprobar que el funcionamiento del contador y cerrojos asincrónicos sean de la manera esperada; Para este fin es necesario conectar las señales descritas en la Tabla 5.III

Terminal del C.I. o punto de Prueba	Nombre de la señal	Terminal del POD 1
TP1	TAUI	0
TP2	PCK2	1
TP3	PDCO	2
TP4	PCK4	3
TP5	PDC1	4
TP6	PCK8	5
TP7	PDC2	6
TP8	PHCK8	7
TP9	TAUD	8
3 de U20	CLK	9
11 de U20	TAU	10

Tabla 5.III

Con las conexiones anteriores realizadas, es posible observar el funcionamiento del contador asincrónico el cual se muestra en la figura 5.5

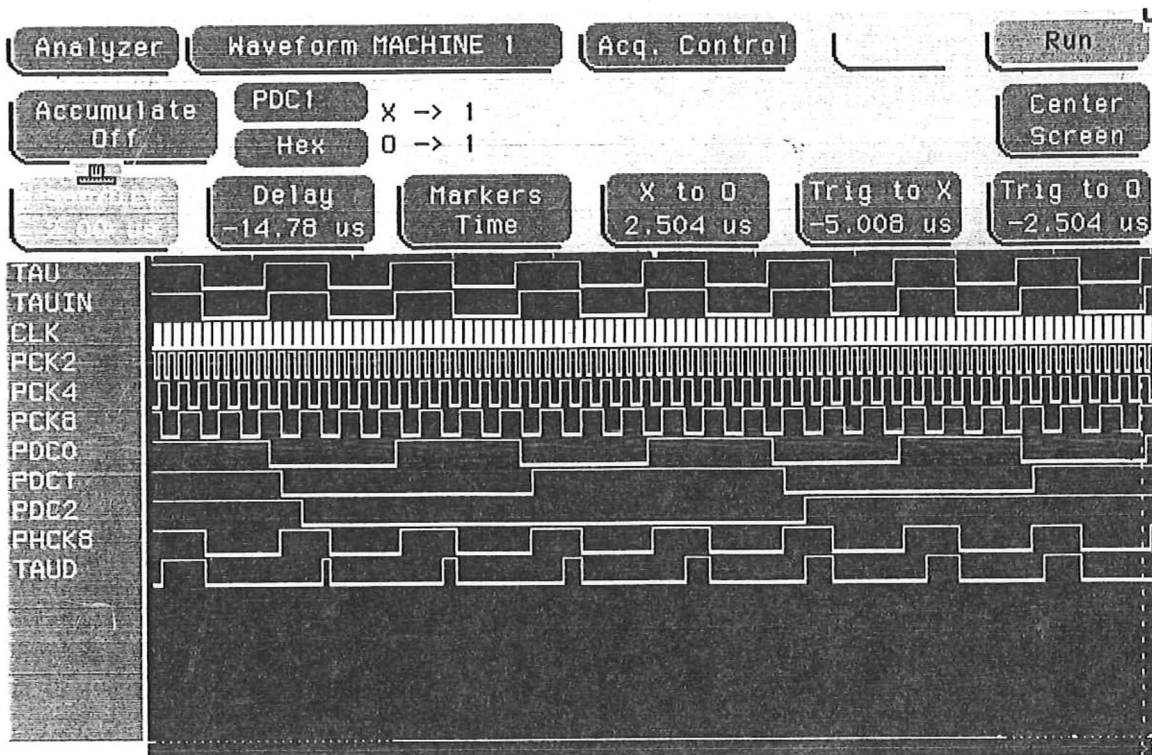


Figura 5.5 Diagrama de tiempos del contador asincrónico

V.5 VERIFICACIÓN DEL CONTADOR Y CERROJOS SINCRÓNICOS

En esta prueba se observa el conjunto de señales implementadas en el FPGA. Para llevar a cabo este fin es necesario conectar las señales especificadas en la tabla 5.IV.

Terminal del header indicado o punta de prueba	Nombre de la señal	Terminal del POD indicado
1 de JP4	BDC0	0 del POD 1
2 de JP4	BDC1	1 del POD 1
3 de JP4	BDC2	2 del POD 1
4 de JP4	BDC3	3 del POD 1
5 de JP4	BDC4	4 del POD 1
6 de JP4	BDC5	5 del POD 1
7 de JP4	BDC6	6 del POD 1
8 de JP4	BDC7	7 del POD 1
9 de JP5	BDC8	8 del POD 1
10 de JP5	BDC9	9 del POD 1
11 de JP5	BDC10	10 del POD 1
12 de JP5	BDC11	11 del POD 1
13 de JP5	BDC12	12 del POD 1
14 de JP5	BDC13	13 del POD 1
15 de JP5	BDC14	14 del POD 1
16 de JP5	BDC15	15 del POD 1
TP10	TAUD	0 del POD 3
TP11	-----	1 del POD 3
TP4	PCK4	2 del POD 3
TP8	PHCK8	3 del POD 3
TP6	PCK8	4 del POD 3
TP7	PDC2	5 del POD 3

Tabla 5.IV

Los octetos menos significativos son observados conectando la línea /HC1 al común mientras que la línea /HC2 es conectada a VCC. La diagrama de tiempos correspondiente se observa en la figura 5.6

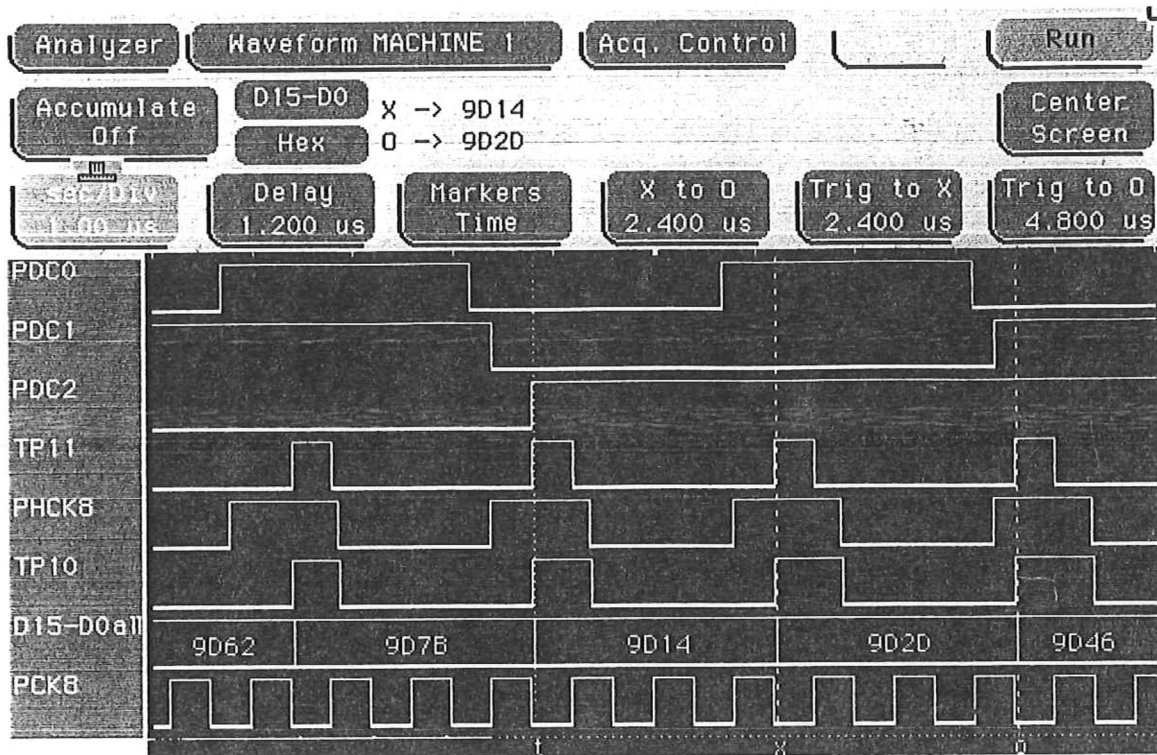


Figura 5.6 Parte baja del contador sincrónico

Para la observación de los octetos más significativos del contador, la línea /HC1 es conectada a VCC mientras que /HC2 es conectado a Tierra. La figura 5.7 muestra el comportamiento del contador en el estado anterior.

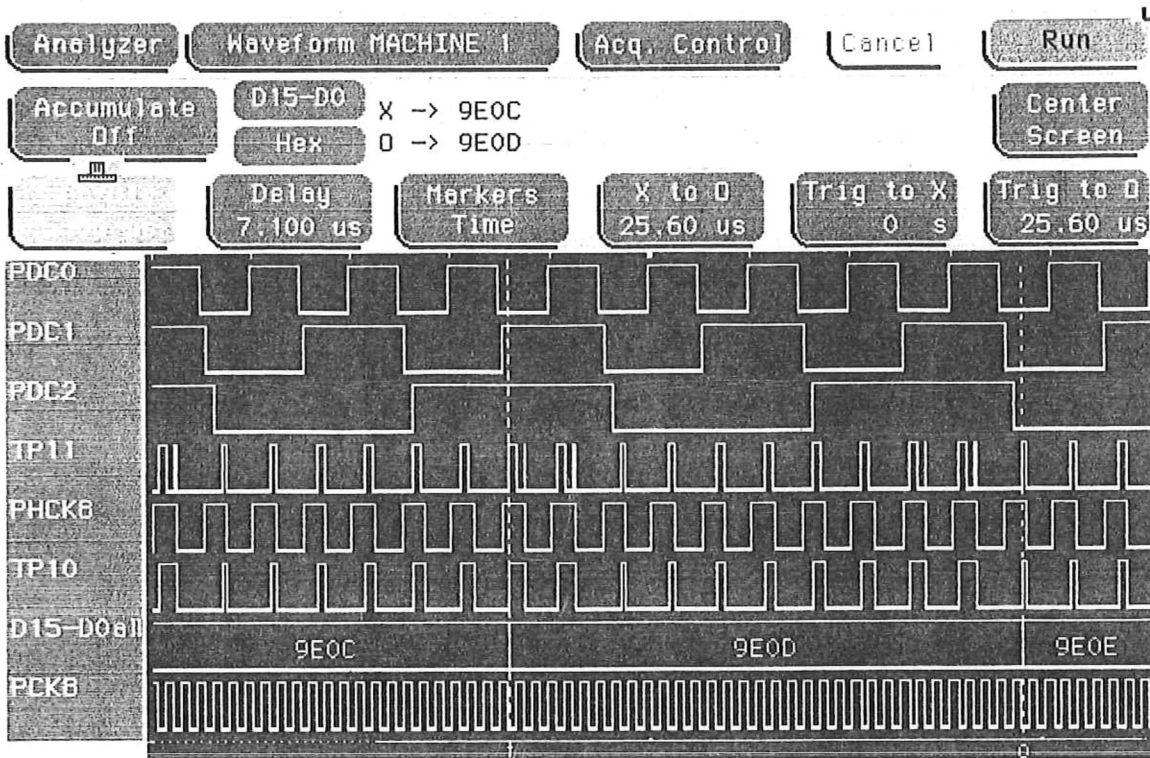


Figura 5.7 Comportamiento de la parte alta del contador sincrónico para /HC2 activa

V.6 GENERACIÓN DEL ALGORITMO DE COMPENSACIÓN PARA LA CUENTA CAPTURADA

Dadas las pruebas anteriores, es necesario realizar una tabla de compensación para obtener la cuenta real como ya se había mencionado en capítulos anteriores (Capítulo IV).

Para obtener esta, es necesario recordar que la salida del contador asincrónico es el reloj del contador sincrónico; Dado que el contador asincrónico siempre comienza en cero y este proporciona un pulso al contador sincrónico al presentarse el primer pulso al asincrónico; siempre se observa que cada cuenta atrapada tiene un desbalance de +8 bits. Dadas las características del contador siempre que se presenta una cuenta diferente de 111_b genera un pulso al contador sincrónico en el intervalo en que la cuenta asincrónica es atrapada, lo que produce de nuevo un desbalance de +8 bits; de tal manera que el algoritmo de compensación sería de la siguiente forma:

- 1.- Aislar los 3 bits menos significativos correspondientes al contador asincrónico
- 2.- Trasladar el valor de acuerdo a la Tabla 5.V
- 3.- Reintegrar los bits

4.- Restar 8 a la cuenta

5.- Si la cuenta asincrónica del contador asincrónico (3 bits menos significativos) volver a restar 8.

Dato del contador asincrónico D2 D1 D0	Dato del cerrojo asincrónico DC2 DC1 DC0	Flancos ascendentes de OBP para capturar en cerrojo asincrónico	Flancos ascendentes de OBP para capturar en cerrojo sincrónico	Flancos ascendentes de OBP para que se de el flanco ascendente en TAU
0 0 0	1 1 1	1	-	4
0 0 1	0 0 0	7	7	11
0 1 0	0 0 1	1	6	10
0 1 1	0 1 0	3	5	9
1 0 0	0 1 1	1	4	8
1 0 1	1 0 0	7	3	7
1 1 0	1 0 1	1	2	6
1 1 1	1 1 0	3	1	5

Tabla 5.V

Como se mencionó con anterioridad una vez generado el algoritmo de compensación, este es implementado en el programa de despliegue de datos y de esta manera se puede observar los datos reales.

VI.

CONCLUSIONES

Con la realización del presente trabajo se obtuvieron muchas experiencias y se aprendieron nuevos conceptos, los cuales quizás no fueron cubiertos totalmente durante el transcurso de la licenciatura, pero que de alguna manera se estaba un poco familiarizados con ellos; de las que en resumen podemos concluir las siguientes:

- Se aprendió que en la realización de circuitos digitales es muy importante tomar en cuenta los tiempos de estabilización (SETUP TIME) y sostenimiento (HOLD TIME) de un dato lógico, para prevenir posibles estados de Meta-estabilidad (o de indecisión) los cuales nos evitarán muchos dolores de cabeza en la realización de diseño de circuitos como el realizado en el presente proyecto (contadores de alta velocidad).
- En el desarrollo del proyecto antes mencionado, se obtuvieron los resultados esperados; se incursionó en la utilización de una nueva tecnología (FPGA's) con lo cual se participó en un congreso con la exposición del tema y se dejaron las bases para la impartición del tema en un curso de titulación.
- Se aprendió también que la utilización e implementación de nuevas tecnologías, abren un nuevo panorama en el desarrollo profesional; simplifican en muchos casos diseños complejos y agilizan la velocidad de los mismos; por lo cual es aconsejable mantenerse siempre a la vanguardia en lo que a desarrollo tecnológico se refiere.

REFERENCIAS:

- [1] XILINX, The Programmable Gate Array Data Book 1991
- [2] MEJIA VAZQUEZ HECTOR, MANUEL ORTIZ AGUIRRE, FRANCISCO DE LA TORRE Y FRANCISCO JAVIER MENDIETA JIMENEZ, 1997. Método de Diseño e Implementación con FPGA en la innovación al procesador de la varianza de Allan. Informe Técnico. Comunicaciones Académicas. Serie Electrónica y Telecomunicaciones. 106 P. CTETT9707
- [3] MEJIA VAZQUEZ HECTOR Y FRANCISCO JAVIER MENDIETA JIMENEZ, 1994. Desarrollo de un Sistema de Procesamiento en Tiempo Real de la Varianza de Allan. Informe Técnico. Comunicaciones Académicas. Serie Electrónica y Telecomunicaciones. 65PP. CIET9403.
- [4] XILINX, The Programmable Logic Data Book, 1994.
- [5] XILINX, ViewLogic Workview Series I vol. 1 April 1994
- [6] XILINX, XACT User Guide, April 1994 (040112801)

I.1 COMUNICACIONES ÓPTICAS[1]

En el mundo moderno, el afán por mantener un buen sistema de comunicación ha sido el principal objetivo del Hombre. Con el fin de llevar a cabo este sistema, se han explotado los diferentes intervalos de frecuencia en radio hasta llegar al punto en el cual se han agotado todo el espectro. Esta ha sido la razón por la cual se ha intentado establecer un sistema de comunicación más eficiente utilizando el espectro de luz; lo cual ha dado un giro en las comunicaciones y se ha enfocado en experimentos por mejorar las comunicaciones ópticas.

En las comunicaciones por fibra la modulación por intensidad de luz es la técnica tradicional utilizada; sin embargo esta impone severas restricciones al empleo óptimo de este medio de comunicación. Para mejorar su desempeño es necesario implementar técnicas empleadas comúnmente en radio; esto implica el control de la dispersión alrededor de la frecuencia de operación del láser, debido al ruido de fase causado por emisiones espontáneas de ocurrencia aleatorias inherentes a su operación. Dichas dispersiones en baja frecuencia pueden ser manipuladas mediante el empleo de lazos de amarre de fase (PLL's) y lazos de costas, pero dispersiones de más alta variación enmascararían la información incrementando la tasa de error. Por lo anterior es necesario caracterizar los dispositivos láser a emplearse en el experimento de comunicación coherente de manera de conocer y determinar sus limitaciones.

I.2 OBJETIVOS DE LOS PARÁMETROS DE CARACTERIZACIÓN

Los parámetros de caracterización de la estabilidad en frecuencia permiten:

- La posibilidad de comparación de dispositivos similares desarrollados por diferentes laboratorios o entre diferentes dispositivos en una aplicación dada.
- La obtención del desempeño de una aplicación en términos de la medida de la estabilidad en frecuencia de un oscilador.

I.3 CARACTERIZACIÓN DE LA ESTABILIDAD EN FRECUENCIA

El problema práctico es como caracterizar las propiedades de la señal de salida de un oscilador real. La señal de un oscilador ideal es una senoidal pura libre de derivas y ruido; sin embargo, cualquier dispositivo real, así sea el más estable, es perturbado por procesos inevitables: ruidos aleatorios, derivas debidas a envejecimiento y/o cambios ambientales.

Para generalizar nuestra discusión sobre osciladores (los cuales poseen un amplio intervalo de frecuencias nominales) es necesario introducir la técnica de normalización, así, la frecuencia instantánea normalizada es:

$$y(t) = \frac{v_r}{v_0} = \frac{1}{2\pi} \frac{d\phi}{dt}$$

Conocida como offset de frecuencia fraccional o simplemente frecuencia fraccional normalizada.

Otra cantidad útil es la integral de $y(t)$:

$$x(t) = \int_0^t y(t) dt = \frac{\phi(t)}{2\pi v_0}$$

Y tendremos también

$$y(t) = \frac{dx}{dt}$$

donde $x(t)$ es proporcional a la fase instantánea pero tiene dimensiones de tiempo por lo que se le denomina “fase-tiempo”.

El empleo de las cantidades antes definidas conlleva a un formalismo consistente, simple de usarse y que tiene un claro significado físico ya que diferencias de fase-tiempo pueden interpretarse directamente como diferencias en tiempo.

Tanto $y(t)$ como $x(t)$ son procesos aleatorios que deben ser descritos por medio de métodos estadísticos.

Los métodos existentes de medición no permiten la obtención de muestras instantáneas del proceso aleatorio $y(t)$. El resultado de una medición de frecuencia se obtiene siempre como el promedio sobre un intervalo de tiempo finito τ . Cualquier muestra y_k es de la forma:

$$\bar{y}_k(t_k, \tau) = \frac{1}{\tau} \int_{t_k}^{t_k + \tau} y(y) dt = \frac{1}{\tau} (x(t_k + \tau) - x(t_k))$$

I.4 MEDICIÓN DIRECTA DE FRECUENCIA POR CONTEO EMPLEANDO UNA VENTANA EN TIEMPO

Este método de medición es el más relacionado con el concepto de frecuencia, esto es, el número de eventos por unidad de intervalo de tiempo.

La figura 1 muestra el diagrama a cuadros de este tipo de sistemas incluyendo todas las funciones comúnmente encontradas en este tipo de instrumentos.

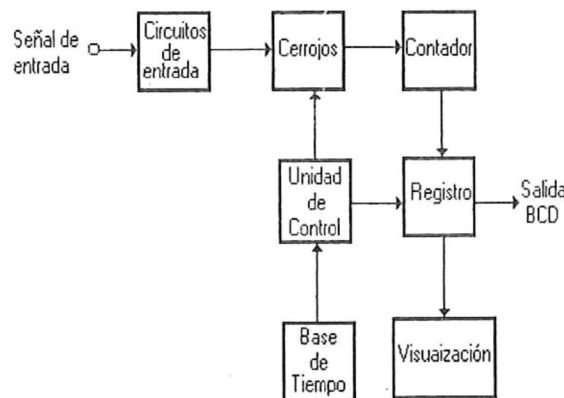


Figura 1

Básicamente, este método consiste en tomar muestras de los pulsos de la señal de entrada, a una razón de tiempo dada por la unidad de control la cual es un factor de la base de tiempo; las muestras tomadas son almacenadas en un registro del cual posteriormente serán tomados los valores obtenidos para procesarlos y desplegarlos.

I. 5 ERROR DE MEDICIÓN EN MEDICIÓN DIRECTA DE FRECUENCIA

La incertidumbre de la medición directa de frecuencia está limitada por dos factores:

- Ambigüedad de ± 1 cuenta.
- Exactitud de la base de tiempo.

de acuerdo con la relación

$$v = \frac{N}{\tau} \quad [Hz]$$

y

$$\Delta v = \frac{N}{\tau} \frac{\Delta \tau}{\tau}$$

donde N es la cuenta desplegada, $\Delta \tau$ es el error temporal, se tiene que

$$\frac{\Delta v}{v} = \pm \frac{1}{v \tau} \mp \frac{N}{v} \frac{\Delta \tau}{\tau^2}$$

$$y = \pm \frac{1}{v \tau} \mp \frac{N}{v} \frac{\Delta v_i}{v_i}$$

así

$$y = \pm \frac{1}{v \tau} \mp \frac{N}{v \tau} y_i$$

donde Δv , v y y se refieren a la frecuencia medida y Δv_t , v_t , y_t se refieren a la frecuencia de la base de tiempos del generador.

De esta manera hay una relación muy sencilla entre la incertidumbre en la medición, la frecuencia a medirse y el tiempo de muestreo. La figura 2 muestra dicha relación como una función de la frecuencia y el tiempo de muestreo.

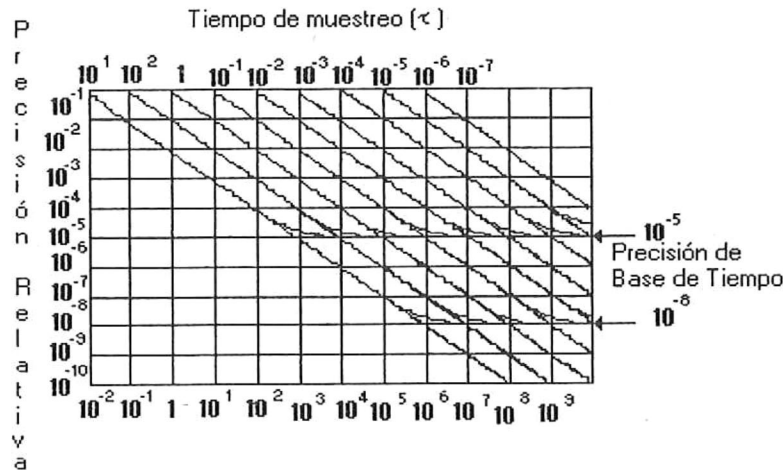


Figura 2

I.6 INCREMENTO DE LA RESOLUCIÓN EN MEDICIONES DE FRECUENCIA

El método clásico de incrementar la resolución de medición de frecuencia es medir un tono de frecuencia (beat frequency) producido por la mezcla de una armónica de la señal desconocida con una armónica del generador de frecuencia estándar.

El principio de esta técnica, denominada heterodinaje, se muestra en la figura 3

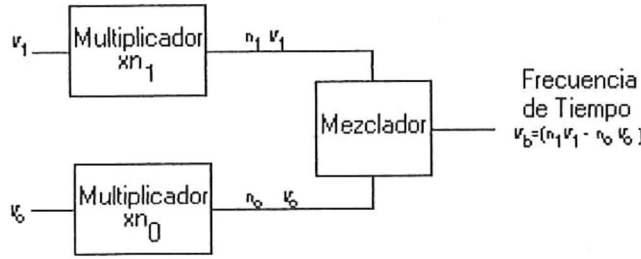


Figura 3

La frecuencia v_1 de la señal desconocida es multiplicado por un factor n_1 y la frecuencia estándar v_0 por un factor n_0 . Los factores n_1 y n_0 son elegidos para que el tono, esto es la diferencia $|n_1 v_1 - n_0 v_0|$ sea baja comparada con las frecuencias armónicas $n_1 v_1$ y $n_0 v_0$ aplicadas a las entradas del mezclador (mixer).

El incremento en resolución deseado empleando el método de heterodinaje se obtiene por medio de la medición del período empleando el tono v_b . Midiendo N períodos, el resultado desplegado es:

$$T_d = \frac{N}{v_b}$$

El error de la base de tiempos es $\pm \Delta T_d$ por lo que la resolución para el tono es:

$$\Delta v_b = \pm \frac{N}{T_d^2} \Delta T_d$$

Expresado como una resolución normalizada a la frecuencia medida n_1 tenemos:

$$\frac{\Delta v_1}{v_1} = y_1 = \mp \frac{N}{n_1 v_1 T_d^2} \Delta T_d$$

es decir, la resolución se incrementa por un factor n_1 con respecto al heterodinaje directo y

con respecto a la medición de frecuencia directa.

Referencias:

- [1]Mejía, F. J. Mendieta, *Sistema de Procesamiento en tiempo Real de la varianza de Allan*, (CICESE, México,1995), CIETT9501,48 p.p.

INSTRUCCIONES DE VIEW LOGIC

Como se menciona con anterioridad el programa de interfase **VIEWLOGIC**, soporta el sistema **WORKVIEW**, el cual contiene herramientas como **ViewDraw**, **ViewSim**, **ViewWave**.

Recordando **ViewDraw**, es una herramienta de **WORKVIEW** para captura de esquemáticos. **ViewSim**, es un simulador lógico interactivo de **VIEWLOGIC**; el cual incluye a **ViewWave**, un visor y editor de formas de onda.

La ejecución de estas herramientas se realiza desde ViewLogic de la siguiente forma:

(➤ Indica la selección requerida para ejecutar comando.)

➤ Window ➤ Open ➤ ViewDraw

➤ Window ➤ Open ➤ ViewSim

➤ Window ➤ Open ➤ ViewWave

VIEWDRAW

ViewDraw, es una herramienta de **WORKVIEW** para entrada de esquemático. Contiene los comandos para realizar capturas esquemáticas como se definen a continuación.

(➤ Indica la selección requerida para ejecutar comando.)

Window menu

Se tienen todas las opciones de ventanas.

➤ Window ➤ Open ➤ Viewdraw ➤ Schematic

Abre una ventana para iniciar o leer un esquemático.

➤ Window ➤ Open ➤ Viewdraw ➤ Symbol

Abre una ventana para iniciar o leer un símbolo.

➤ Window ➤ Close

Cierra la ventana activa sin salvar los cambios realizados.

➤ Window ➤ CloseAll

Cierra todas las ventanas activas sin salvar los cambios realizados.

View Menu

Se tienen todas las opciones de visualización de ventanas.

➤ View ➤ Refresh (F5 del teclado)

Actualiza la visión de ventana.

➤ View ➤ Pan (F6 del teclado)

Desplaza la visión de la ventana hacia la posición del mouse.

➤ View ➤ In (F7 del teclado)

Realiza un acercamiento de la visión de la pantalla.

➤ View ➤ Out (F8 del teclado)

Realiza un alejamiento de la visión de la pantalla.

➤ View ➤ Full (F10 del teclado)

Realiza un ajuste del diagrama completo en la ventana activa.

➤View ➤Select (F9 del teclado)

Realiza acercamiento de los objetos seleccionados

Add Menu

Se tienen todas las opciones para adicionar elementos al diagrama.

➤Add ➤Net

Traza una ruta de alambrado.

➤Add ➤Comp

Agrega un componente al circuito.

➤Add ➤Label

Coloca una etiqueta al elemento(s) seleccionado(s).

➤Add ➤Attr

Coloca un atributo al elemento(s) seleccionado(s).

➤Add ➤Bus

Traza una ruta de alambrado de un ducto.

➤Add ➤Pin

Agrega una terminal al componente seleccionado, en la edición de símbolos.

➤Add ➤Box

Agrega una caja al componente seleccionado, en la edición de símbolos.

➤Add ➤Text

Coloca una etiqueta de texto al elemento(s) seleccionado(s).

Level Menu

Se tienen todas las opciones para cambiar el nivel jerárquico del diagrama.

➤Level ➤Push ➤Schematic

Baja de nivel jerárquico visualizando el diagrama esquemático.

➤Level ➤Push ➤Symbol

Baja de nivel jerárquico visualizando el diagrama del símbolo seleccionado.

➤Level ➤Pop

Sube de nivel jerárquico visualizando el diagrama del símbolo o esquemático seleccionado.

File Menu

Contiene todos los comandos para el manejo de archivos.

➤File ➤Write

Salva el archivo activo y verifica las condiciones eléctricas del circuito.

➤File ➤Writeto

Salva el archivo con un nombre especificado.

Change Menu

Contiene todas las opciones para realizar cambios al esquemático.

➤Change ➤Attr ➤Dialog ➤All

Muestra todos los atributos de un objeto seleccionado para cambiarlos o agregar más.

Export Menu

Contiene todas las opciones para realizar exportaciones del archivo del esquemático a otros formatos.

➤Export ➤Wirelist ➤Viewsim

Exporta el archivo del esquemático a formato ViewSim para simulación.

Plot Menu

Contiene todas las opciones para realizar impresiones del esquemático.

➤Plot ➤Go

Ejecuta la impresión del archivo con las condiciones establecidas.

Bye Menu

Contiene la opción para abandonar el ambiente de la herramienta.

➤Bye ➤Quit

Ejecuta el abandono del ambiente de la herramienta sin salvar cambios.

FUNCIONAMIENTO DE VIEWSIM

VIEW SIM

ViewSim, es un simulador lógico interactivo de **VIEWLOGIC**.

Setup Menu

Contiene las opciones de inicialización.

➤ Setup ➤ Viewwave

Inicializa el archivo para simulación en ViewWave.

➤ Setup ➤ Waveform ➤ Periodic

Inicializa las condiciones de las señales periódicas conocidas.

Force Menu

Contiene comandos para obligar una señal a un nivel lógico conocido.

➤ Force ➤ L

Obliga a una señal a permanecer en un nivel lógico bajo.

➤ Force ➤ H

Obliga a una señal a permanecer en un nivel lógico alto.

➤ Force ➤ X

Obliga a una señal a permanecer en un nivel lógico indeterminado.

Sim Menu

Contiene las opciones para realizar la simulación bajo las condiciones establecidas.

➤ Sim [t]

Ejecuta la simulación en el tiempo determinado t.

Execute Menu

Contiene las opciones de ejecución de archivos.

➤ Execute ➤ Logfile

Realiza la ejecución de un archivo .LOG. La utilización de este archivo nos permite grabar condiciones iniciales de simulación.

FUNCIONAMIENTO DE VIEWWAVE

VIEW WAVE

ViewWave, es un visor y editor de formas de onda.

Window Menu

Se tienen todas las opciones de ventanas.

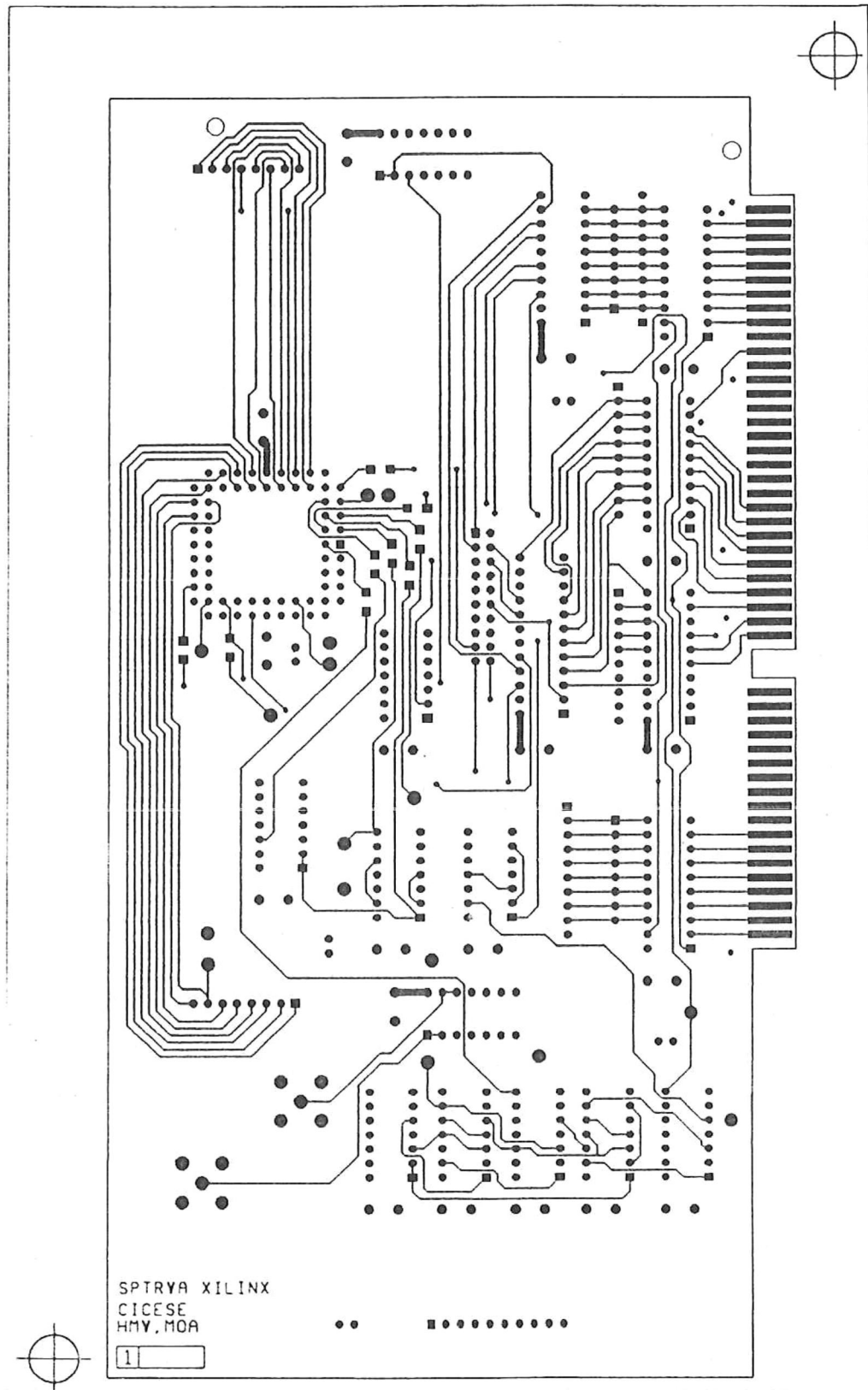
➤ Window ➤ Open ➤ Viewwave ➤ Viewsim

Abre una ventana para simulación en forma de onda, con señales definidas en un archivo generado en **ViewSim**.

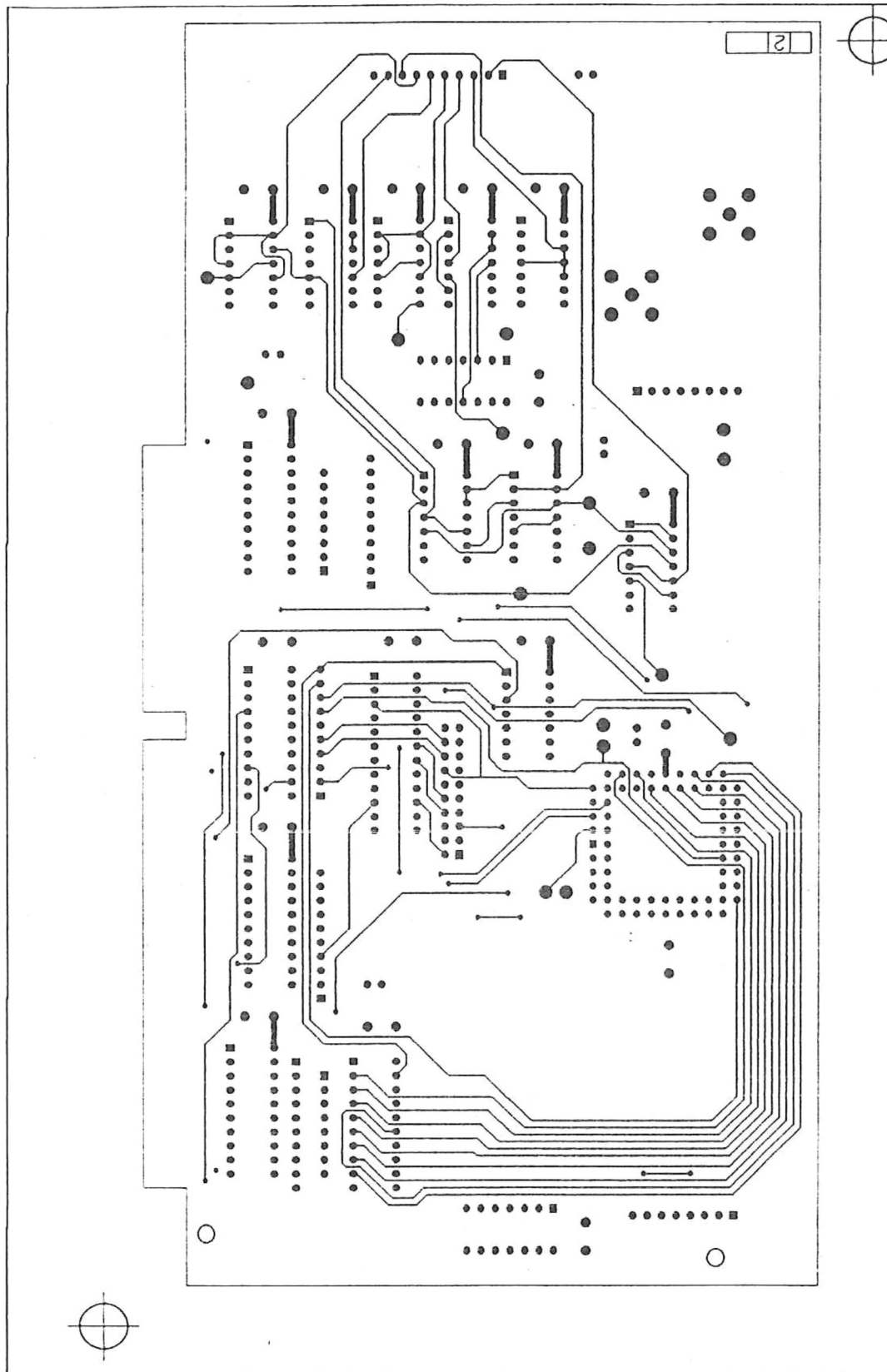
View Menu

Se tienen todas las opciones de visualización de ventanas. Estas opciones son las mismas que contiene **ViewDraw**.

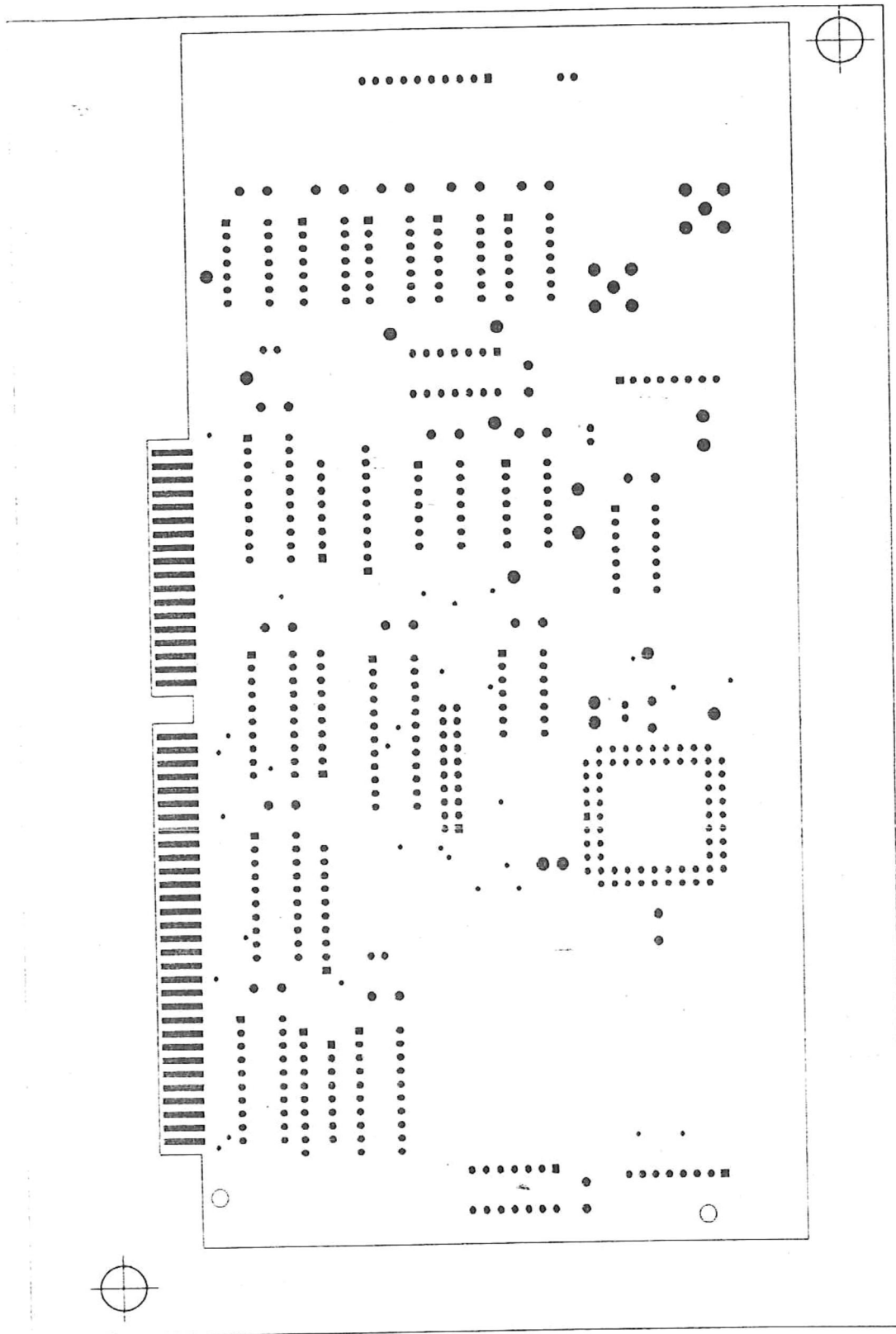
MASCARILLAS Y ESQUEMATICOS



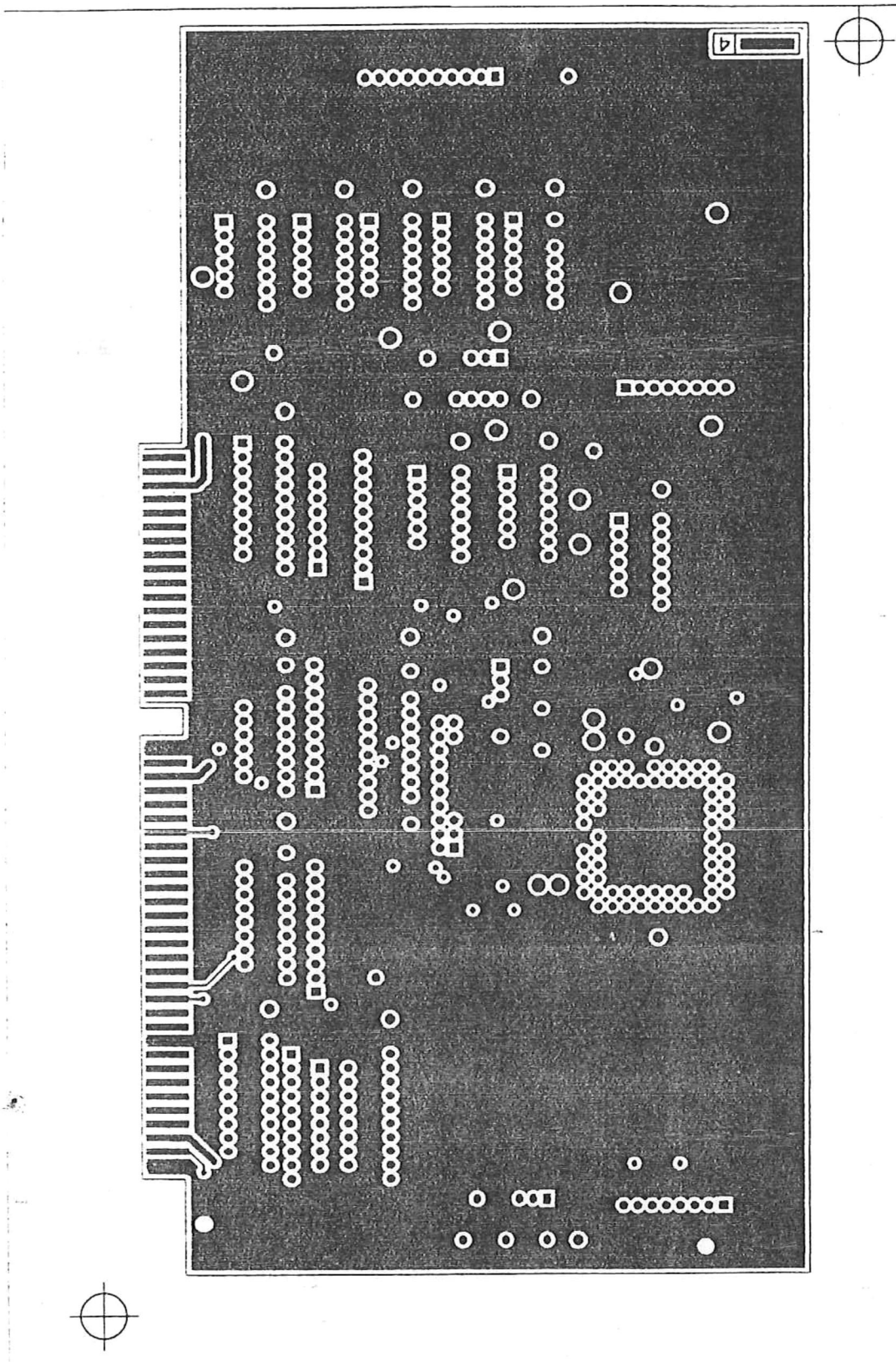
Mascarilla de lado de componentes



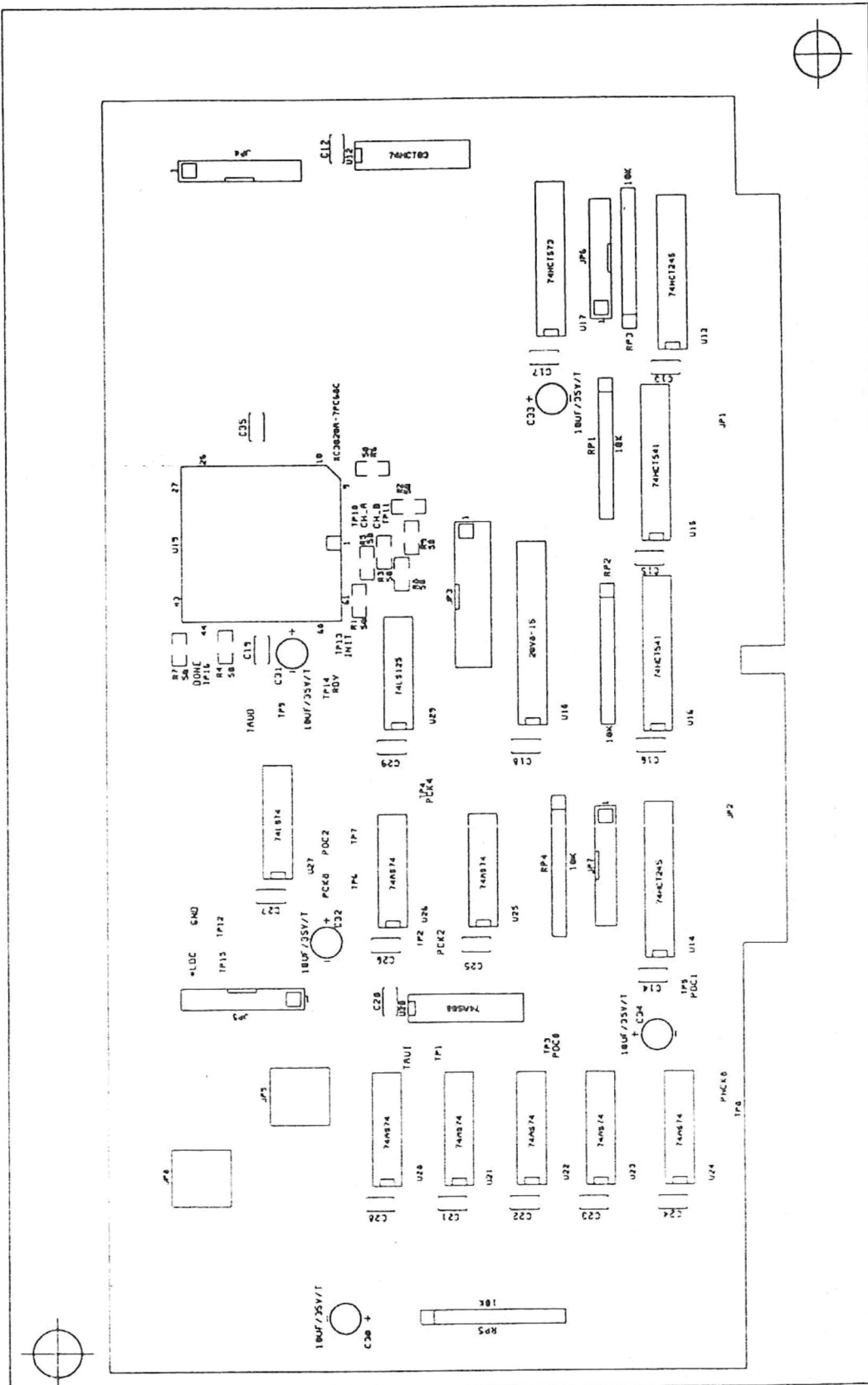
Mascarilla de lado de soldadura



Mascarilla de pads

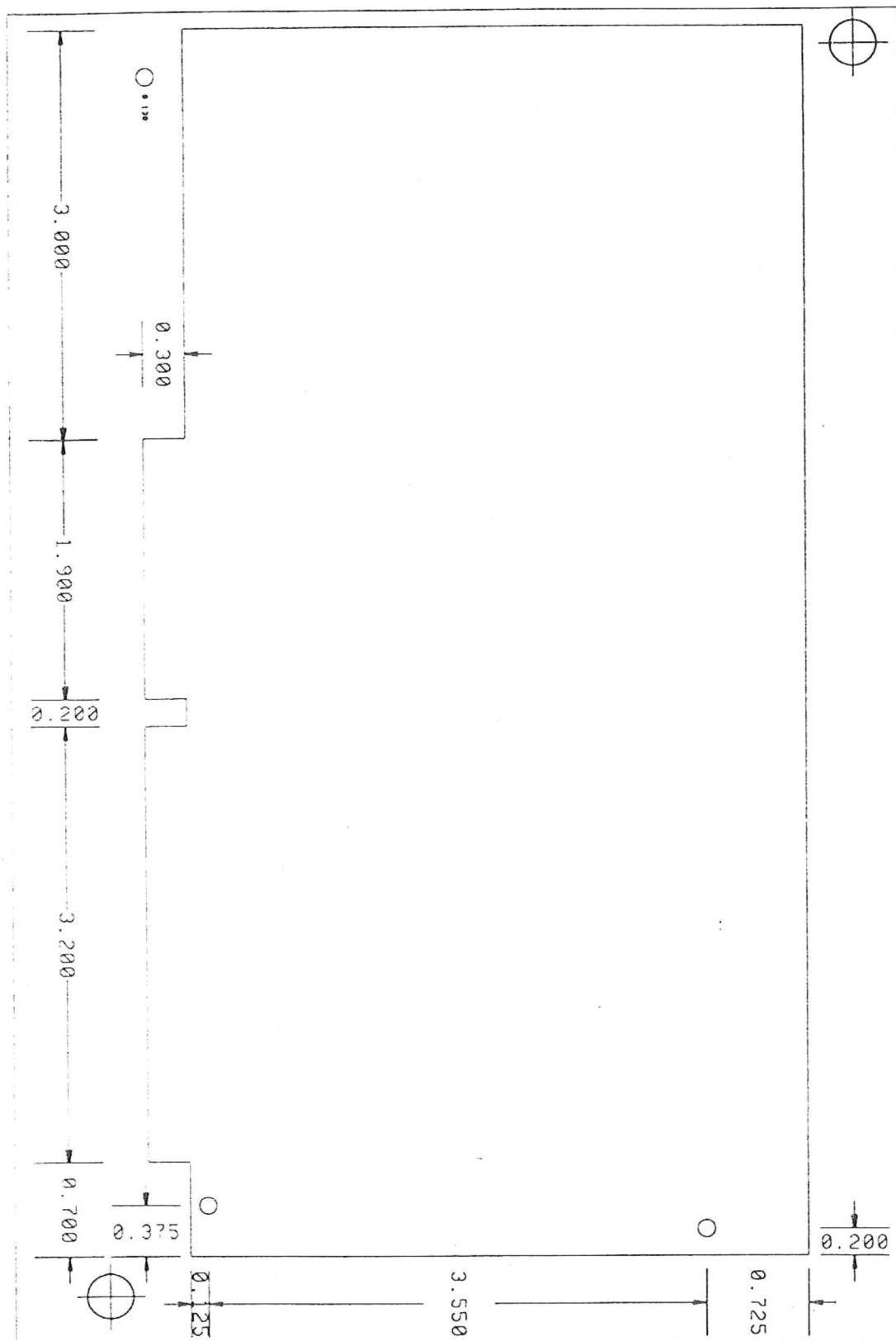


Mascarilla de plano de tierra



Mascarilla de etiquetas de componentes

APENDICE C



Mascarilla de dimensiones de circuito impreso

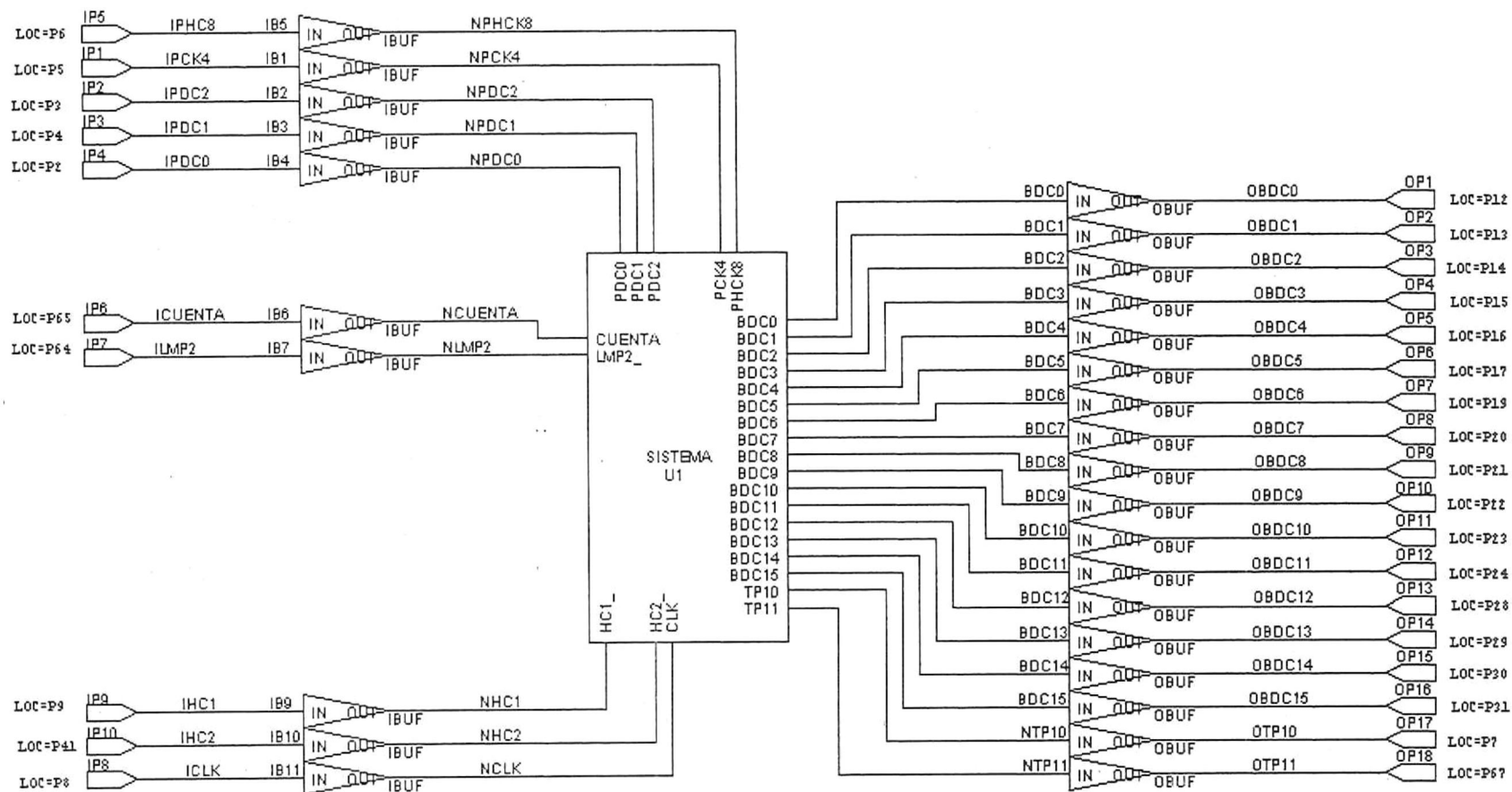


Diagrama de terminales de FPGA

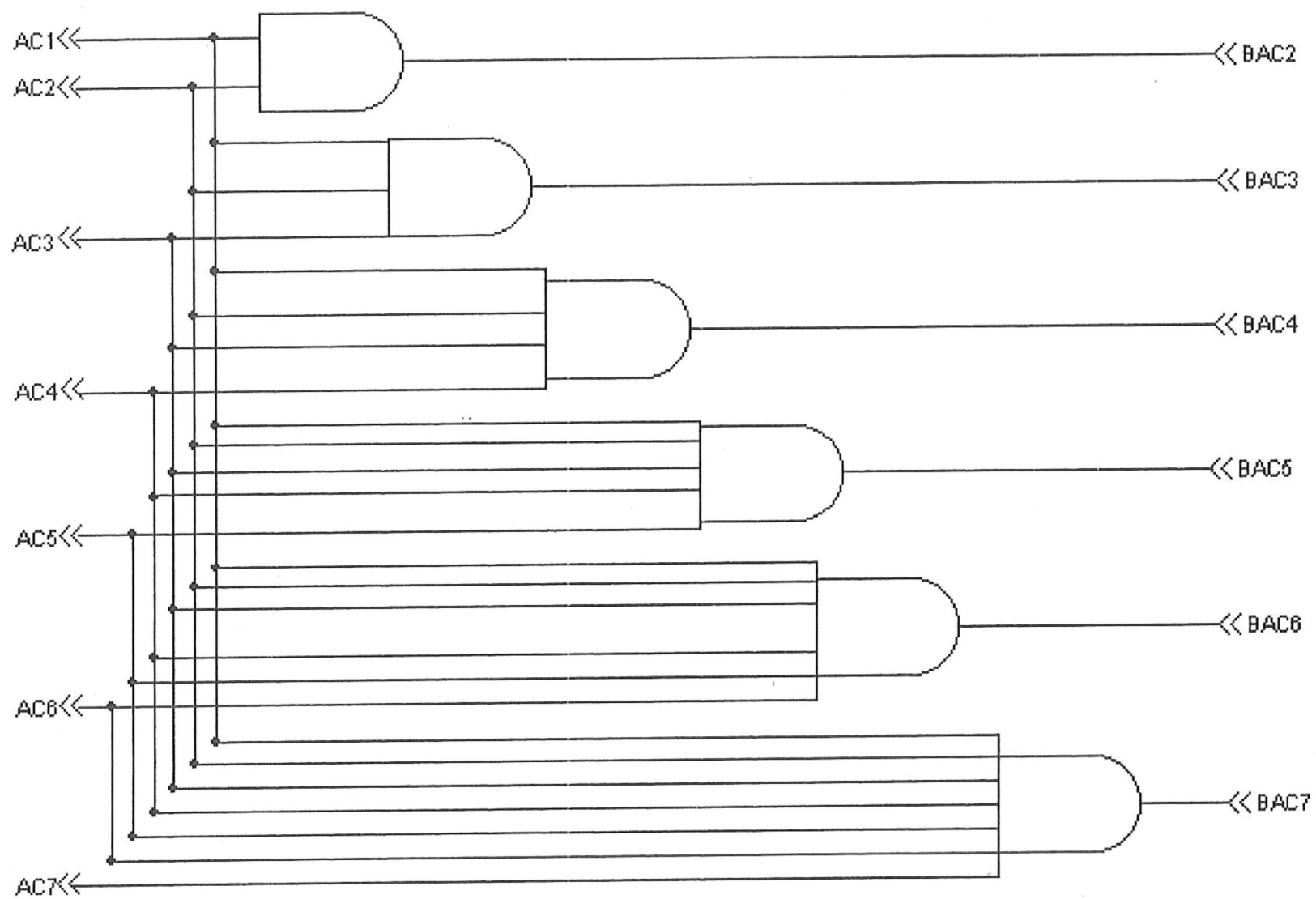


Diagrama de circuito acelerador de acarreo

