Universidad Central "Marta Abreu" de Las Villas
Facultad de Ingeniería Eléctrica
Departamento de Telecomunicaciones y Electrónica

Trabajo de Diploma

Desarrollo de un Analizador Lógico de 8 bits en lenguaje C y utilizando el puerto paralelo.

Autor: Curtis Bernard Dodds

Tutor: Msc. Juan Pablo Barrios Rodríguez

Curso 2003-2004
Año del 45 Aniversario del Triunfo de la Revolución

Hago constar que el presente trabajo fue realizado en la Universidad Central “Marta Abreu” de Las Villas como parte de la culminación de los estudios de la especialidad de Telecomunicaciones y Electrónica, autorizando a que el mismo sea utilizado por la Institución para los fines que estime conveniente, tanto de forma parcial como total y que además no podrá ser presentado en eventos, ni publicados sin autorización de la Universidad.

_____

Firma del Autor

Los abajo firmantes, certificamos que el presente trabajo ha sido realizado según acuerdo de la dirección de nuestro centro y el mismo cumple con los requisitos que debe tener un trabajo de esta envergadura referido a la temática señalada.

_____ _____

Firma del Tutor

Firma del Jefe de Dpto.
Donde se defiende el trabajo

_____

Firma del Responsable de
Información Científico- Técnica

PENSAMIENTO

"Es mucho mejor entender el Universo como realmente es que vivir en las tinieblas, por satisfactorias y reconfortantes que éstas nos puedan parecer."

Carl Sagan

"El último tribunal de apelación es el formado por la observación y la experimentación, no por la autoridad."

Thomas Huxley

DEDICATORIA

El Camino hacia el triunfo es caro y escabroso, no todos los hombres están dispuestos a pagar este precio. Solo profundizando en el conocimiento, la persistencia y en el propósito será la base de todo éxito máxime cuando ello ha sido precedido por una obra maestra de formación.

A los autores de esta obra en mi vida dedico esta tesis.

- ***Mi madre Saturine Dodds***
- ***Mi padre Cyrus Dodds***
- ***Mi hermanita Carol Dodds***
- ***Mi amigo que falleció Neville Pierre***

A mi madre que con tanta paciencia y amor me educó y a mi padre luz inspiradora en todos mis esfuerzos.

Curtis

AGRADECIMIENTO

- Agradezco todos los profesores del departamento de Telecomunicaciones y Electrónica por mi formación profesional.
- A mi Tutor Msc. Juan Pablo Barrios Rodríguez más por sus conocimientos aportados en la conformación de este trabajo.
- A mi familiar, sin cuya ayuda, hubiese sido mucho más dura y angosta esta trayectoria.
- A todos mis amigos del caribe por su incondicional apoyo.
- A todos los que de una forma u otra me ayudaron a realizar este sueño profesional.

Curtis Bernard Dodds

A todos, Gracias.

RESUMEN

En el presente trabajo se realiza un estudio de los analizadores lógicos, tanto desde el punto de vista de sus características principales, como de su disponibilidad en el mercado.

Este estudio permitió elegir, como respuesta a las necesidades estudiantiles de laboratorios reales de Electrónica Digital, el diseño de un analizador lógico: LA8bits que, utilizando un mínimo de recursos, permite convertir el display del ordenador en un analizador de señales digitales en número de 8 bits y con valores de muestreo de la señal de hasta 10 KHz.

Para la realización de este programa se utilizó el lenguaje C (Borland C) y el ensamblador para procesadores Intel.

La frecuencia de muestreo se obtiene de la reprogramación de la subrutina de atención a interrupción del tick del timer del ordenador.

El puerto paralelo se redefine en modo bidireccional para el muestreo y proyección a bajo nivel de las 8 señales del puerto de datos paralelo.

Por último se proponen un conjunto de prácticas que, utilizando este instrumento, facilitan el desarrollo de las habilidades prácticas de los estudiantes.

INDICE

INTRODUCCION

El presente trabajo se justifica ante la carencia de medios tecnológicos en nuestra facultad para la visualización de múltiples señales digitales. Lo anterior incide negativamente en el desarrollo de habilidades prácticas en los estudiantes.

La ventaja fundamental de este proyecto radica en contar con un instrumento real-virtual que, utilizando el puerto de impresora, el cable de la misma y un software apropiado; permita visualizar en el display del ordenador un mínimo de 8 señales digitales.

En el curso 1993-1994 se realizaron experiencias con un trabajo de curso de estudiantes de Telecomunicaciones y Electrónica relacionado con la visualización de varias señales digitales (5) por un solo canal de un osciloscopio utilizando la técnica de multiplexado en tiempo y desplazamiento del nivel DC. Los resultados fueron satisfactorios pero las mayores dificultades radican en no poder ampliar el número de canales a 8 y el deterioro continuado que han tenido los instrumentos de laboratorio y accesorios.

En el curso 2002-2003 se realizó un trabajo de diploma que dio como resultado un instrumento virtual, elaborado en LabView 6.0 que permitía visualizar hasta 8 señales digitales en el display del ordenador utilizando el puerto de impresora.

Como mayores dificultades de este trabajo están las limitaciones en frecuencia de dicho software (10 Hz) y la necesidad de realizar instalaciones de DLLs para los diferentes sistemas operativos (2000, XP, etc.) lo que conllevaba a que el mismo software no se comportara de la misma manera en todas las configuraciones.

Internacionalmente varias firmas ofertan analizadores lógicos (Texas Instruments, Hewlett Packard, Tektronix) tanto como equipos independientes, como combinados con una PC.

En todos los casos los precios de estos equipos superan las posibilidades del centro para su adquisición, además de las restricciones que impone el bloqueo con respecto a su venta a nuestro país.

Situación del problema:

El problema se puede plantear entonces mediante la pregunta: ¿Se podrá elaborar un instrumento real-virtual que, con un mínimo de recursos de hardware, permita convertir a la computadora en un analizador lógico de varios canales con una respuesta de frecuencia aceptable para las aplicaciones que las asignaturas de Electrónica Digital necesitan?.

Interrogantes Científicas:

1. ¿Cómo asumen otros países en vías de desarrollo la necesidades de contar con estos tipos de instrumentos para la enseñanza de la Electrónica Digital?.
2. ¿Es alguna de estas soluciones factible de aplicar a las características específicas de Cuba y de la enseñanza en nuestro centro?.
3. ¿Cuál es la disponibilidad en Internet de este tipo de aplicaciones?.
4. ¿Es necesario desarrollar una particular o aplicar alguna de las disponibles?
5. ¿Si se aplica alguna disponible, cómo evaluar su efectividad dentro del proceso docente de nuestro centro?.
6. ¿Tanto si se aplica una disponible como si se desarrolla una herramienta específica, cómo validar su utilidad?.
7. ¿En qué conjunto de temas, prácticas de laboratorio y/o actividades independientes se puede utilizar las misma?.

Objetivo del trabajo:

Contar con un instrumento real-virtual que permita convertir el ordenador en un analizador lógico de al menos 8 canales.

Objetivos específicos:

1- Contar con una revisión del estado del arte en lo que respecta al desarrollo de instrumentos reales, virtuales o mixtos destinados al análisis de señales digitales, sus parámetros más importantes y precios.
2- Determinar la necesidad o no de utilizar algún instrumento de este tipo que esté disponible en Internet y que cumpla con los requerimientos de bajo costo antes mencionados.
3- Desarrollar el instrumento de no utilizar ninguno de los disponibles en Internet.
4- Evaluar el instrumento desarrollado aplicado.

Tareas de investigación:

1- Realizar una revisión en Internet, centros de documentación y empresas que suministren analizadores lógicos.
2- Seleccionar y documentar variantes, a partir de la selección anterior, que tengan factibilidad de realización y/o adquisición en nuestro entorno.
3- Proponer y/o desarrollar una de las variantes antes señaladas.
4- Evaluar la efectividad de la misma en un conjunto de actividades prácticas y/o de estudio independiente correspondientes a las asignaturas de Electrónica Digital.
5- Elaborar el Informe final.

El informe se encuentra estructurado en tres capítulos:

CAPITULO I*:* Caracterización de los analizadores lógicos.

CAPITULO II: Selección y/o desarrollo de un analizador lógico utilizando el puerto paralelo del ordenador.

CAPITULO III: Desarrollo de actividades prácticas para Electrónica Digital utilizando el analizador lógico de 8 bits.

CAPITULO I*:* Caracterización de los analizadores lógicos.

I.1 Introducción.

Las funciones específicas que poseen los analizadores lógicos confieren a éstos una serie de posibilidades que no poseen otros equipos electrónicos de medición. En el presente capítulo se describen los distintos tipos de analizadores utilizados en el dominio digital, se realiza una comparación previa con los osciloscopios indicando sus similitudes y diferencias, así como su utilización más idónea en función de la aplicación de medida.

Posteriormente se realiza la descripción exhaustiva de los analizadores lógicos destinados al análisis del funcionamiento de sistemas digitales complejos, en los que se requiere la observación simultánea de multitud de canales y una gran potencia en sus sistemas de adquisición, disparo y presentación.

Tras mostrar el diagrama de bloques básico de un analizador lógico, se describen los diferentes modos de presentación con los cuales se pueden obtener cronogramas, tablas de estados, etc.

Seguidamente se estudian los diferentes modos de adquisición y disparo, y los métodos de muestreo para analizar señales digitales con un consumo óptimo de memoria sin perder resolución horizontal. También se tratan los aspectos concernientes a los sistemas de sondas y puntas de prueba que hacen posible el acceso a las señales de interés en sistemas digitales complejos.

I.2 Concepto básico de un analizador lógico.

Un analizador lógico es un instrumento de laboratorio que se utiliza para monitorear varias señales digitales sobre una referencia de tiempo común. Es útil para observar las señales en caso de un evento determinado.

El valor de las señales digitales muestreadas se visualiza de una manera gráfica o en formato numérico. Los parámetros más importantes de una medición son el tiempo de_muestreo y la condición de disparo. El tiempo de muestreo es el

tiempo que transcurre entre dos lecturas consecutivas de los canales de entrada.

Es importante que este tiempo se ajuste a la variación que va a sufrir la señal. Es decir, si la señal puede contener pulsos de 1 ms de duración, el tiempo de muestreo no deberá ser mayor que ese valor, pues se podría ocultar ese pulso.

La condición de disparo es la señal que inicia el registro o almacenamiento de los datos, lo que continúa hasta que se llena el buffer o la memoria interna del instrumento. Esta condición de disparo puede definirse como un nivel lógico en una o varias señales, una transición o, incluso, como una secuencia de eventos.

I.3 Clasificación inicial

En una primera clasificación de los equipos destinados al análisis lógico, se pueden distinguir tres tipos de equipos:

- **Sondas lógicas.** Son equipos muy simples destinados a detectar estados lógicos de uno o varios nodos del circuito bajo ensayo de modo estático, es decir, sin tener registro de la evolución temporal de los estados detectados.
- **Analizadores lógicos.** Es la denominación general que incluye a los equipos destinados a medir estados lógicos de un circuito en régimen dinámico, esto es, teniendo un registro de su evolución temporal.
- **Analizadores de protocolo.** Representan una herramienta comúnmente utilizada en la actualidad para la comprobación de funcionamiento de los sistemas que trabajan en red. Estos equipos observan, analizan o simulan los procesos de comunicación que rigen los intercambios de datos entre diferentes dispositivos interconectados entre sí.

Antes de continuar con la descripción de los equipos electrónicos anteriormente citados resulta conveniente realizar un rápido repaso de los principales conceptos ligados con la lógica digital.

I.3.1 Tipos de lógica digital. Niveles lógicos.

Las más comunes son:

- **Lógica TTL.** Está caracterizada por utilizar circuitos alimentados a +5 V. El nivel alto

(H) corresponde a tensiones superiores a 2V (40% de VCC) mientras que el nivel bajo

(L) a tensiones inferiores a 0,8 V (16% de VCC). Los valores de tensión intermedios definen un estado indeterminado.

- **Lógica CMOS.** Está caracterizada por utilizar circuitos de alimentación variable desde +0.5V hasta +18V aunque el valor típico es de +15V. El nivel alto (H) corresponde a tensiones superiores al 70% de la tensión de alimentación mientras que el nivel bajo (L) a tensiones inferiores a 30%. Los valores de tensión intermedios definen un estado indeterminado.

I.3.2 Sondas lógicas.

Su componente básico es el comparador lógico compuesto por un par de comparadores con el cual es posible realizar la definición de los estados lógicos.

La salida de los comparadores se conecta a elementos de visualización (leds) o sonoros (buzzers) para realizar la indicación del estado lógico detectado. Los circuitos comparadores se suelen diseñar con una pequeña histéresis con la cual se aumenta la inmunidad al ruido.

También suelen disponer de un circuito detector de pulsos, tanto positivos como negativos. En la figura 1 se puede ver el aspecto externo de una sonda lógica.

Figura.I.1. Aspecto externo de una sonda lógica.

I.3.3 Analizadores lógicos.

Como hemos dicho anteriormente, los analizadores lógicos permiten realizar un estudio dinámico de la evolución temporal de varios nodos de un circuito digital.

Figura I.2. Medidas con analizador lógico.

I.3.3.1 Analizadores lógicos y osciloscopios digitales.

A la hora de realizar el estudio de funcionamiento de los circuitos digitales, se pueden elegir como herramienta de análisis osciloscopios digitales DSO (digital storage oscilloscopes) o analizadores lógicos LA (logic analyzer).

Un osciloscopio se utiliza cuando se desea medir una pequeña excursión en la señal observada. También se utiliza cuando se necesita obtener información paramétrica como por ejemplo el tiempo entre dos puntos del borde de subida de un pulso.

En general, el osciloscopio se utilizará cuando se necesitan mediciones de voltaje o tiempo de gran precisión en señales analógicas. El osciloscopio es un equipo muy familiar que suele ser muy útil en determinadas aplicaciones donde existen pocas señales de interés (2 o 4 como máximo) y donde es necesario obtener medidas con gran exactitud tanto de tensiones (amplitudes, pendientes

de subida o bajada, oscilaciones, calidad de las formas de onda, transitorios (glitches), ruido, etc.) como de tiempos (retrasos o solapamientos, tiempos de propagación, períodos, estabilidad en frecuencia (jitter) etc.).

Sin embargo, cuando se necesita realizar el análisis simultáneo de un gran número de señales digitales, o cuando se requiere de un sistema complejo de disparo ligado a un determinado patrón establecido a partir de múltiples señales digitales, no siendo necesaria una gran exactitud en las medidas de amplitud y tiempo, es recomendable el uso de los LA.

Un analizador lógico se utiliza cuando se necesitan analizar muchas señales digitales al mismo tiempo, debido a que estos instrumentos permiten organizar y desplegar las señales de manera fácil y ordenada.

También se utiliza cuando se necesitan ver las señales de un circuito de la misma manera que lo hace el sistema. Esto permite saber cuando ocurren transiciones en un bus, relativo a las transiciones en otros buses. Otra aplicación muy frecuente del analizador es cuando se necesita activar el disparo del instrumento con un patrón específico de un bus en particular.

La diferencia básica desde el punto de vista de la arquitectura de estos dos equipos estriba en el número y complejidad de sus convertidores analógicos/digitales ADC (analog to digital converter).

Mientras que un DSO dispone de como máximo 4 ADC tipo Flash de 8 o 10 bits (2N-1 comparadores internos) los AL pueden disponer de un hasta un centenar (o más) de ADC de 1 bit (con 1 ó 2 comparadores internos).

En la actualidad, los DSO más avanzados están provistos de sistemas de disparo y adquisición suficientemente potentes para realizar medicions complejas en circuitos digitales, mientras que los modernos AL disponen de uno o más canales de adquisición tipo DSO que permiten mejorar la exactitud de algunas medidas.

1.3.4 Arquitectura de un analizador lógico.

En la figura 3 se muestra el diagrama de bloques de un analizador lógico donde se incluyen los componentes básicos que permiten realizar la funciones básicas, estas son:

- **Captura y muestreo.** Las señales se capturan mediante sondas o puntas de prueba específicas. El gran número de estas puntas de prueba hace que usualmente se presenten agrupadas en canales denominados “pods” que permiten conectar 8, 16 ó más señales cada uno. Suele existir un pod específico (de menor efecto de carga y mejor ancho de banda) para las señales especiales externas (señales de reloj, interrupciones, etc.). Las señales analógicas recogidas por las puntas de prueba se convierten en datos al pasar por los ADC (comparadores) y el registro de muestreo.
- **Disparo y adquisición.** El analizador lógico requiere de una o varias señales de reloj para realizar el análisis del resto de las señales. Este reloj, que se puede obtener a partir de alguna señal exterior o de la salida de un generador interno, se utiliza para sincronizar el sistema de disparo y gestionar la memoria de adquisición del analizador donde se almacena la información requerida relativa al evento de disparo (pretrigger o postrigger).
- **Análisis y visualización.** A partir de la información memorizada se realiza su análisis y presentación con los formatos o modos establecidos por el usuario. En muchos casos el analizador lógico dispone una arquitectura compatible PC de modo que se puede manipular la información adquirida mediante plataformas de software de alto nivel.

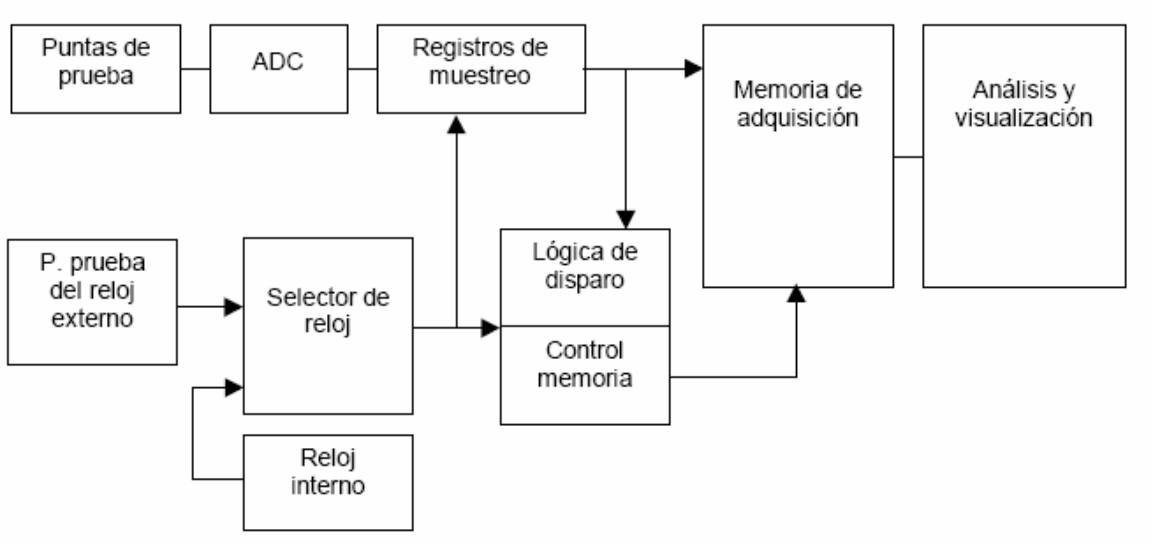

Figura I.3. Diagrama de bloques de un analizador lógico.

I.3.4.1 Modos de muestreo en analizadores lógicos.

Existen dos modos de funcionamiento típicos en un analizador lógico dependiendo del modo de muestreo utilizado.

- **Muestreo asincrónico (timing analyzer mode).** Es un modo de trabajo similar al de un

DSO. El analizador muestra gráficos de múltiples entradas verticales con un eje horizontal común que representa el tiempo (timing o cronograma). La información de los canales de entrada se muestrean a partir de una señal de reloj interno cuya frecuencia se puede seleccionar en función de la frecuencia propia de las señales de entrada y de la profundidad de la memoria de adquisición (como en un DSO). Este modo de funcionamiento se utiliza para analizar la evolución temporal de las señales de un sistema digital.

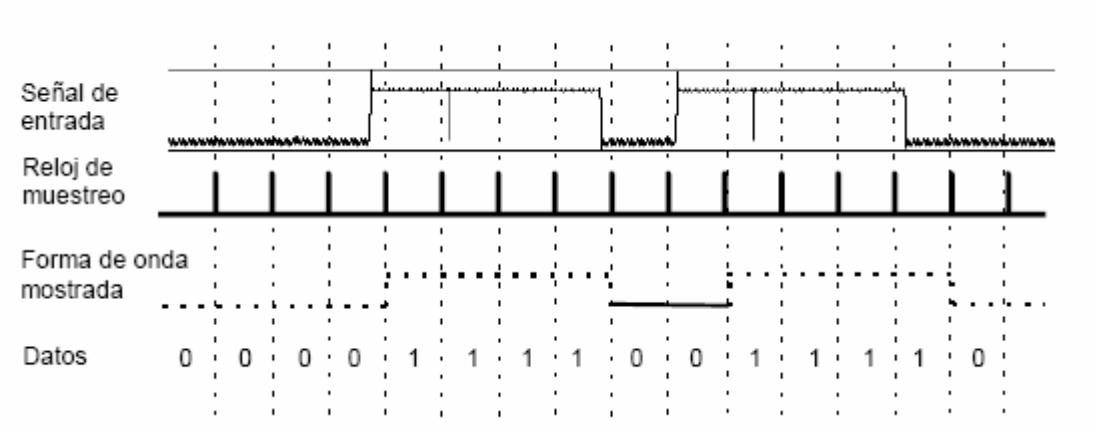

Figura I. 4. Ejemplo de medidas en modo “timing analyzer”

La exactitud de las medidas depende, por lo tanto, de la frecuencia de muestreo como se muestra en la figura 1.5. La máxima incertidumbre en tiempo coincide con el período del reloj de muestreo.

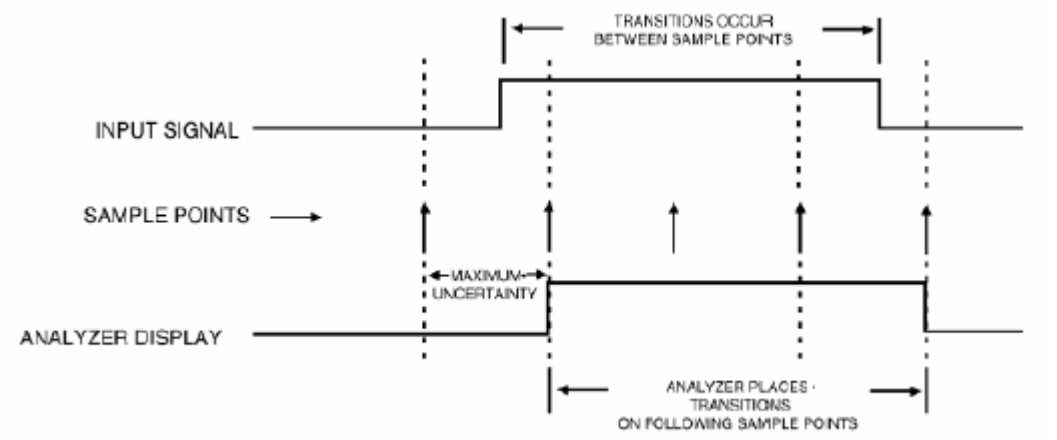

Figura I.5*.* Incertidumbre en la medición en modo “timing analyzer”

- Muestreo sincrónico (state analyzer mode). Una de las señales de entrada se toma como reloj de muestreo, por lo tanto, los datos introducidos en la memoria de adquisición están determinados por las transiciones del reloj externo.

Este modo es utilizado para analizar el funcionamiento de microprocesadores o dispositivos digitales específicos.

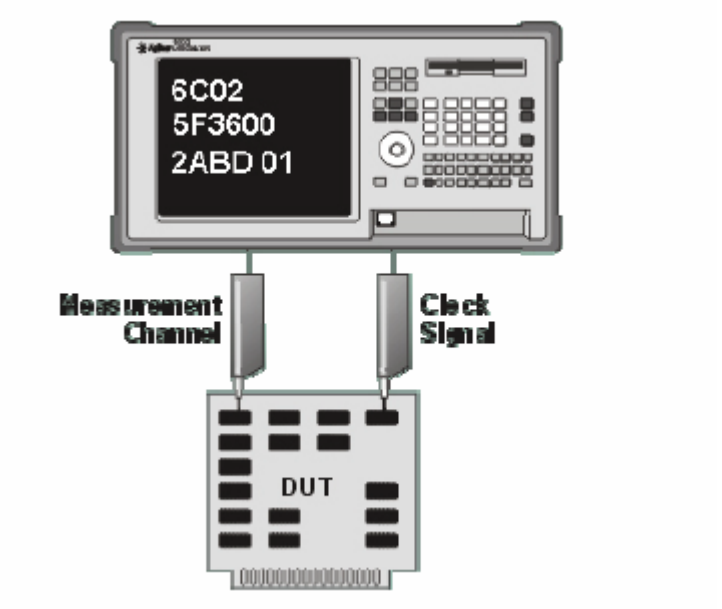

Figura I.6. Analizador lógico en modo “state analyzer”

I.3.4.2 Sistema de disparo.

Una vez muestreadas las señales de entrada se ha de decidir cuales son de interés para el análisis y determinar así su registro en la memoria de adquisición. Esto se realiza gracias al sistema de disparo. A continuación se realizará un estudio de los métodos de disparo usualmente utilizados en modo asíncrono (timing analyzer).

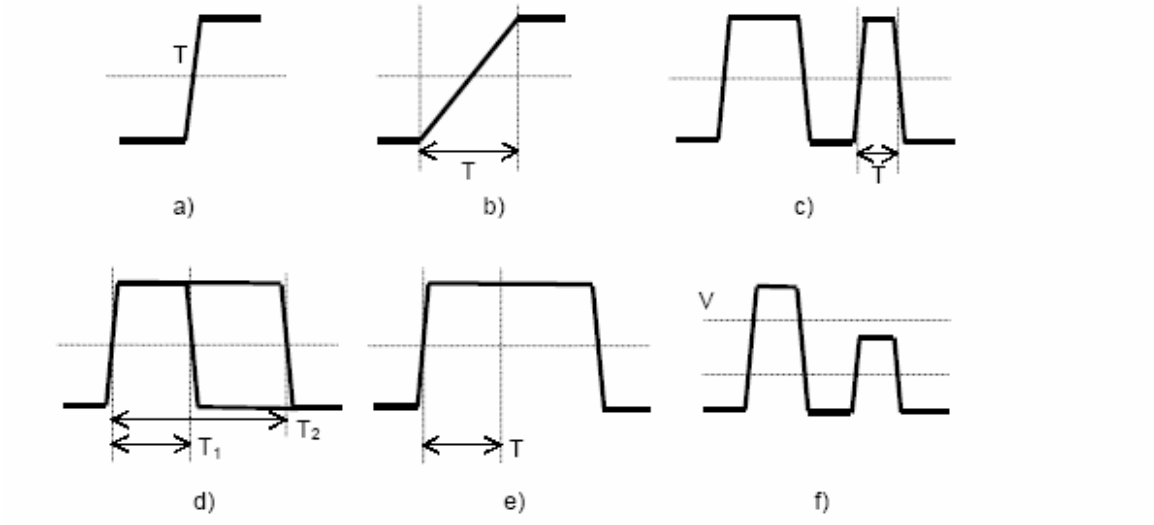

Figura I.7. Distintos métodos de disparo.

- **Disparo por pendiente (edge triggering).** La adquisición del analizador se realiza mediante la detección de la transición positiva o negativa de una determinada señal de entrada. (figura 1.7. a)
- **Disparo por tiempo de transición (slew-rate triggering).** Sólo se activa el disparo cuando se detectan flancos con tiempos de transición mayores (o menores) que una cierta cantidad de tiempo predefinida (figura 1.7. b).
- **Disparo por transitorio (glitch triggering).** Se utiliza para detectar transitorios estrechos (glitches) que suelen ser efecto o causa de un mal funcionamiento del sistema. Para su detección se configura un tiempo T. Todo pulso de duración inferior se considerará glitch y activará el disparo del sistema (figura 1.7. c).
- **Disparo por anchura de pulso (pulse width triggering).** Es similar al método anterior. Ahora, una vez definidos los tiempos T1 y T2, sólo los pulsos de anchura mayor que T1 y menor que T2 activarán el disparo (figura 1.7. d).
- **Disparo por exceso de duración (timeout triggering).** Cuando aparecen pulsos de anchura mayor que un tiempo T predefinido el sistema se dispara (figura 1.7. e).
- **Disparo por defecto de amplitud (runt pulse triggering).** Una vez definido un determinado nivel umbral de amplitud mínima se puede disparar el sistema tras la detección de pulsos defectuosos de escasa amplitud (figura 1.7. f).

- **Disparo lógico (logic triggering).** Se determina el disparo mediante combinación lógica de dos o más señales de entrada. Se suele configurar este método de disparo introduciendo una ecuación con operadores lógicos.
- **Disparo secuenciado (setup-and-hold triggering).** Este método de disparo evalúa la posición y duración temporal relativa entre determinadas señales y la transición de otra señal de referencia. El disparo se efectúa (o no) cuando cumple el cronograma establecido por los tiempos “setup” y “hold” como se muestra en la figura 1.8.

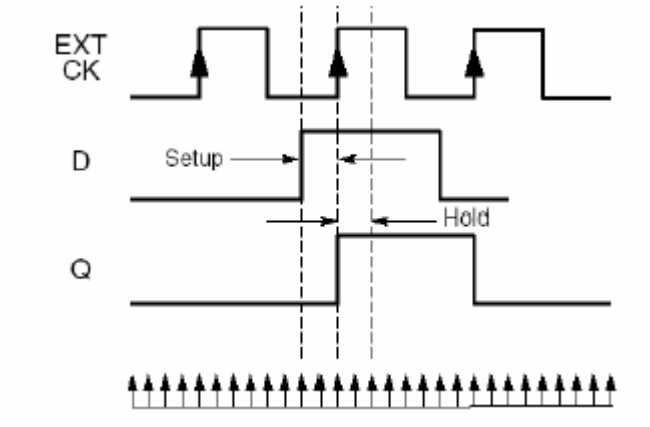

Figura I.8. Disparo secuenciado (setup-and-hold triggering)

I.3.4.3 Adquisición de datos.

Como se ha citado anteriormente, el proceso de almacenamiento en memoria (adquisición de datos) se realiza cuando se produce un disparo del sistema (triggering) con el objetivo de realizar el análisis de las señales seleccionadas en el intervalo de “interés”: antes (pretrigger) y/o después (posttrigger) del disparo.
La duración de este intervalo está limitado, entre otros factores, por el tamaño (profundidad) de la memoria de adquisición que suele ser de tipo cola anillada, es decir, cuando la memoria está llena, los datos nuevos “empujan” a los más antiguos eliminándolos de la memoria. En un modo de funcionamiento asíncrono en cada pulso del reloj se memoriza un nuevo dato. Si se persigue una buena

resolución horizontal la frecuencia de reloj debe ser alta y en poco tiempo de adquisición se puede sobrepasar la capacidad de la memoria.

En el ejemplo de la figura 1.9 se observa un ejemplo con una señal tipo “ráfaga” en la que existe una porción importante de tiempo sin transiciones separando intervalos estrechos de señal con transiciones. Si la memoria es de 4096 posiciones sólo se podría adquirir la primera ráfaga de señal y la memoria estaría llena en su mayoría con datos nulos.

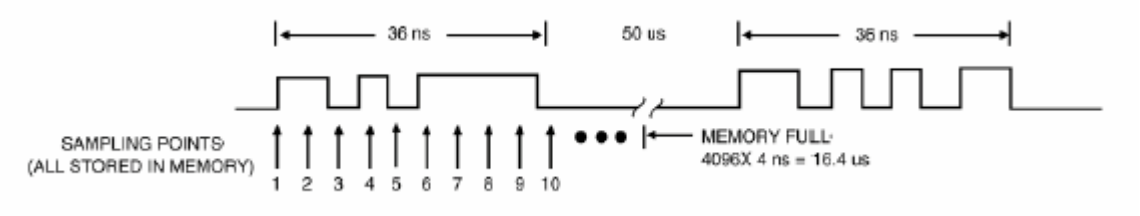

Figura I.9*.* Muestreo de alta resolución no optimizado

En la figura 1.10 se muestra el método de adquisición denominado “transitional sampling” con el cual se consigue optimizar el almacenamiento de la información capturada.

Consiste en guardar en memoria sólo las transiciones, mediante un detector de pendiente, y su duración, mediante un contador. En el ejemplo mostrado sería posible, mediante este método, almacenar decenas de ráfagas con tan solo 4096 posiciones de memoria.

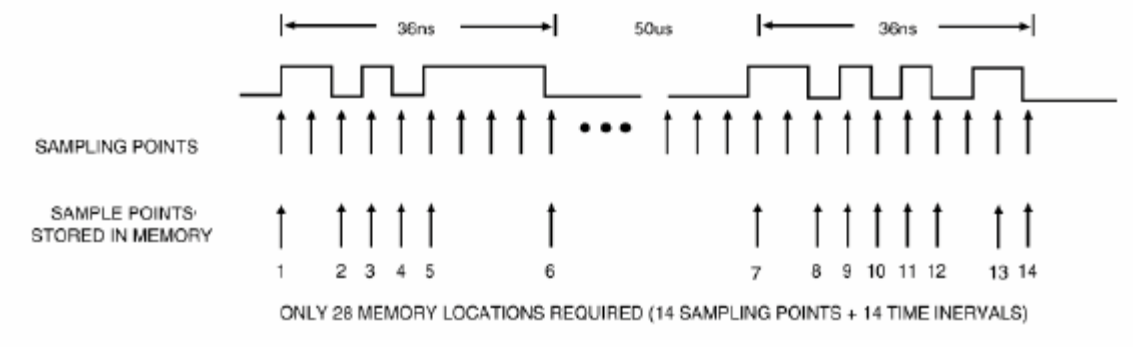

Figura I.10. Muestreo de alta resolución con detector de transiciones

Otro método interesante de adquisición es aquel que permite capturar glitches optimizando el consumo de memoria. En la figura 1.11 se muestra su principio de funcionamiento.

El glitch se detecta cuando la señal cruza el umbral de definición de estado lógico más de una vez entre instantes de muestreo. Una vez detectado, el analizador almacena este evento y lo muestra mediante un trazo discontinuo coincidente con el siguiente punto de muestreo. De este modo no es necesario incrementar la frecuencia de muestreo en exceso para la detección de glitches y por consiguiente se optimiza el uso de la memoria de adquisición.

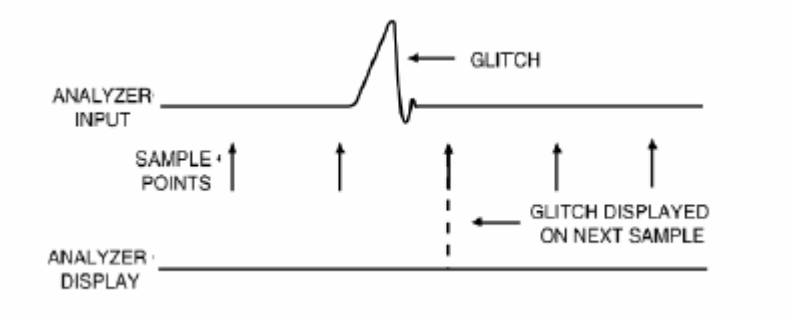

Figura I.11*.* Detección de un glitch

I.3.5 Sondas y puntas de prueba.

En cada uno de los “pods” del analizador se conecta un cable plano o cilíndrico conductor en cuyo extremo se sitúan las sondas o puntas de prueba a conectar

a los nodos de interés del DBE. En un sistema estándar para analizadores lógicos se realiza la conexión al DBE mediante micro-pinzas que se conectan normalmente a los pines de los circuitos integrados como se muestra en la figura 1.12.

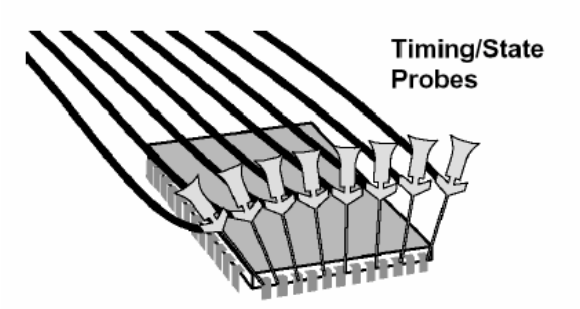

Figura I.12. Puntas de prueba tradicionales en LA

El circuito eléctrico equivalente de estas sondas se muestra en la figura 1.13.

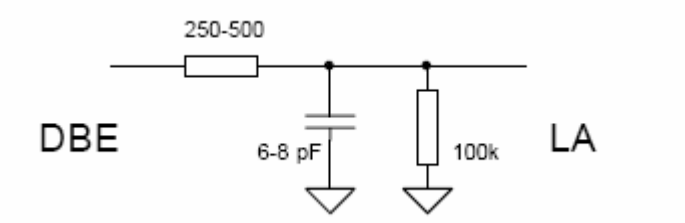

Figura I.13. Circuito equivalente de la punta de prueba de un LA

La existencia de una carga resistiva de 100kΩ y de otra capacitiva de 6 a 8 pF representa un potencial efecto de carga sobre la señal digital bajo estudio. En la figura 1.14 se muestran por separado estos posibles efectos de carga que tendrán mayor importancia cuanto mayor sea la impedancia de salida de los circuitos digitales que generan las señales.

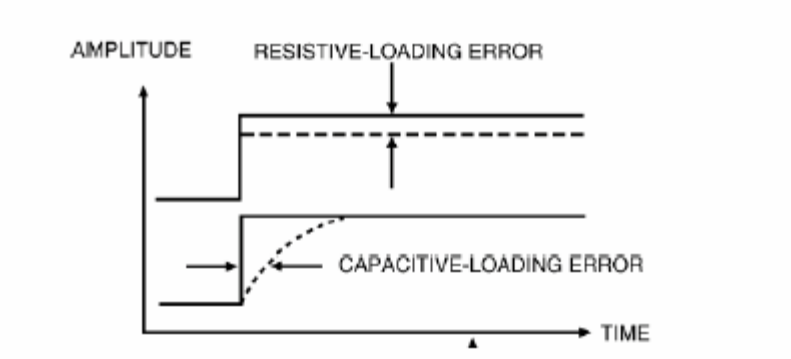

Figura I.14. Efecto de carga de una punta de prueba

El método estándar de conexión presenta ciertas dificultades cuando el número de canales a conectar es grande. Además, en la actualidad se ha popularizado el uso de componentes SMD cuyas dimensiones no permiten, en muchos casos el uso de sondas estándar.

En la figura 1.15 se muestra un nuevo tipo de puntas de pruebas, denominadas “fine-pitch probes” utilizadas para esta tecnología. El proceso de conexión se mejora, entre otros factores, al existir una doble conexión eléctrica en cada pin del circuito integrado con lo cual se aumenta la fiabilidad del sensado.

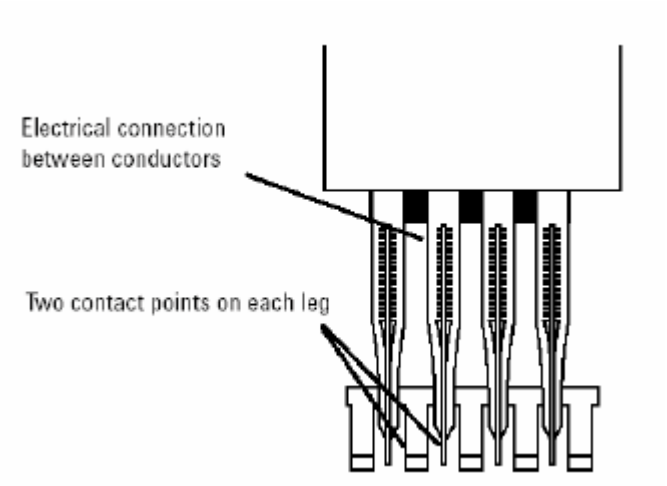

Figura I.15. Punta de prueba para dispositivos SMD

En la figura 1.16 se muestra una imagen de cómo se realiza la conexión de las modernas puntas de prueba sobre un circuito integrado SMD.

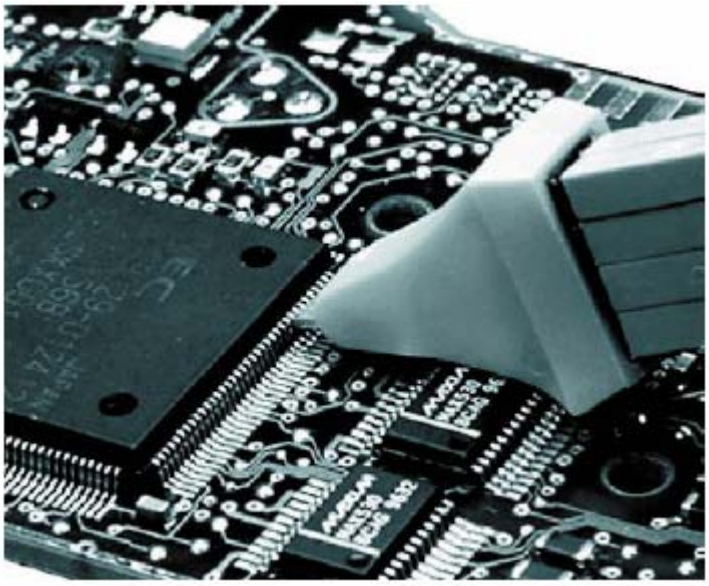

Figura I.16. Ejemplo de conexión de las puntas de prueba para SMD

Cuando el número de puntas de prueba requeridas sobre un mismo circuito integrado es grande resulta más conveniente usar adaptadores específicos como el de la figura I.17. El adaptador se monta sobre el circuito integrado permitiendo la conexión de puntas de prueba estándar. En este caso se muestra el adaptador apropiado para circuitos integrados tipo TQFP.

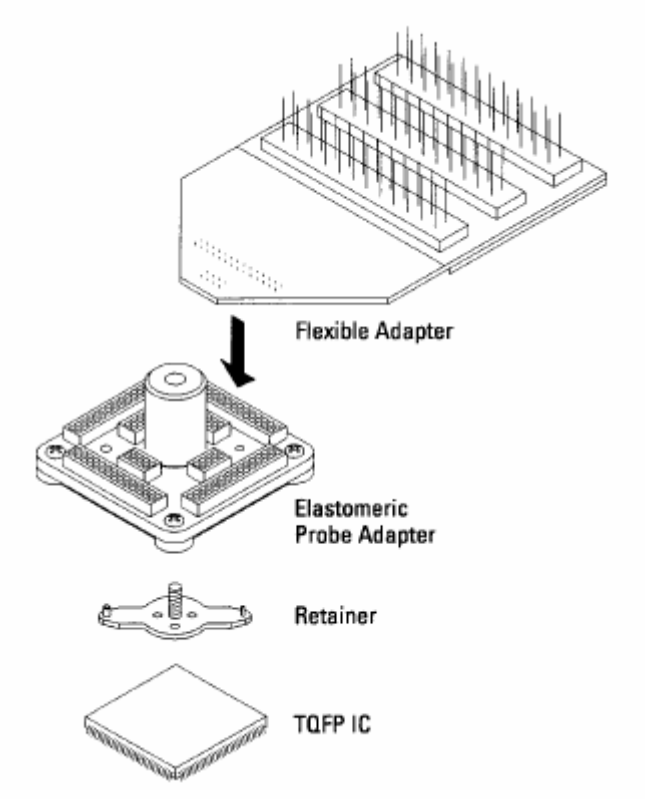

Figura. I.17*.* Puntas de prueba para dispositivos específicos

I.4 Parámetros más importantes de un analizador lógico

I.4.1 Velocidad, el ancho y la profundidad (capacidad de memoria).

El analizador lógico debe poder capturar los eventos más rápidos de un diseño, suministrar suficientes canales y tener abundante memoria. Los diseños actuales suelen demandar un analizador lógico que sólo sea eficiente en una o dos de estas áreas.

A medida que las velocidades de relojes y flancos de señal aumentan, a medida que crece la complejidad, se incrementa la probabilidad de que los tres requisitos resulten cruciales para evaluar y probar con éxito los diseños futuros.

I.4.2 Mayor velocidad

Es absolutamente esencial que el analizador lógico pueda seguir la señal más rápida dentro de un diseño. Para asegurar una clara visibilidad del funcionamiento interno, el analizador lógico debe capturar y analizar aun los eventos más rápidos.

Para los procesadores actuales de alta velocidad, esto significa que el analizador lógico debe brindar una resolución de tiempo inferior a 1 ns. Asimismo, se necesitan ventanas de establecimiento y retención precisas para soportar la adquisición de estados de alta velocidad. Aunque la mayoría de los analizadores lógicos actuales están preparados para manejar velocidades de estado de 100 MHz, no serán capaces de operar con la próxima generación de procesadores de propósito general y su mayor velocidad.

La velocidad por sí sola no garantiza el éxito. El analizador lógico debe examinar millones de datos para hallar rápidamente la información que necesita. Es por eso que el disparo (la forma en que el analizador lógico determina cuándo capturar los datos) es tan importante. Cuando se investiga el problema de un diseño sofisticado, los síntomas visibles son a menudo una intrincada red de eventos de hardware y software interrelacionados.

En consecuencia, el analizador lógico debe monitorear una multitud de señales y procesar gran cantidad de combinaciones de instrucciones de software, accesos de datos, violaciones de temporización, espurios y violaciones de establecimiento y retención. El disparo tradicional, que apenas toma una decisión condicionada por pulso de reloj, a menudo no puede localizar los sutiles eventos asincrónicos que siempre amenazan la integridad de los diseños avanzados.

I.4.3 Número de señales.

Las aplicaciones actuales no son solo más rápidas sino, además, de mayor tamaño. Los diseños con procesadores de alta calidad requieren de 100 a 150

canales para cubrir todas las entradas y salidas del dispositivo. En las aplicaciones complejas, tales como un diseño con múltiples procesadores, el número de canales llega al orden de los centenares. Existen asimismo diseños actuales que, para analizar completamente una aplicación, requieren el monitorizado de más de 1000 señales. Si su diseño aún no requiere centenares de canales, es conveniente disponer de un analizador lógico que se pueda expandirse fácilmente para admitir las demandas de los diseños futuros.

Es también importante que todos los canales sean capaces de sostener la máxima velocidad de adquisición del analizador lógico. Lamentablemente, la mayoría de los analizadores brindan un número limitado de canales de alta velocidad, aunque soportan centenares de conexiones.

Hoy en día, casi ningún diseño tiene pocas señales de alta velocidad. Es bastante común que se requiera visualizar cada entrada y cada salida con resoluciones de tiempo menores que 1 ns.

El software de desensamblaje y los adaptadores de sonda del analizador lógico son otros factores cruciales para un análisis eficaz de procesadores complejos con centenares de conexiones. Con tantas familias de procesadores en el mercado, el analizador lógico debe soportar numerosos adaptadores de sonda. Además, el software de desensamblaje debe proporcionar una clara visión de las instrucciones que se ejecutan y las ramas condicionadas que realmente se toman.

I.4.4 Capacidad de memoria.

Para descifrar la relación entre decenas o incluso centenares de señales a lo largo de muchos ciclos, un analizador lógico debe extraer y almacenar una extraordinaria cantidad de datos.

Si bien la mayoría de los analizadores ofrecen una alta capacidad de memoria, es generalmente a expensas de alguna otra función. Algunos analizadores lógicos pueden brindar la memoria requerida para la depuración avanzada sólo

desactivando una función importante, tal como el estampado de tiempo. Tales compromisos en capacidades de adquisición diluyen los beneficios de la memoria de mayor tamaño.

I.4.5 Simplicidad.

Los tres requisitos básicos de velocidad, número de canales y capacidad de memoria permiten elegir el analizador lógico más adecuado para sus necesidades de diseño. Si bien es esencial determinar en primer lugar cuál de esos criterios es más importante, debe tenerse presente que será muy probable que el próximo diseño necesite un analizador lógico con prestaciones excepcionales en esas tres áreas.

CAPITULO II: Selección y/o desarrollo de un analizador lógico utilizando el puerto paralelo del ordenador.

II.1 Revisión de analizadores lógicos comerciales.

Para cumplir con el objetivo de este capítulo se realizó primeramente una revisión en Internet de las principales firmas que ofertas analizadores lógicos, las facilidades de estos; a partir de las características estudiadas en el capítulo anterior, así como otros factores como precio, acople al PC, etc.

A continuación se resumen algunos de los analizadores lógicos típicos encontrados con un conjunto de facilidades mínimas para el desarrollo de prácticas de laboratorio de Electrónica Digital básica y con una relación funcionalidad-precio aceptables.

II.1.1 ANALIZADOR LÓGICO DE 16 ENTRADAS DIGITALES, LOGIC-3P, REF. KYHE 20-021

- Analizador lógico de ancho de banda de 40 Mhz con 16 entradas digitales de 128 Kbit de memoria por canal. Realiza un máximo de 200 muestras MS/seg.
- Dispone de una entrada de reloj externo y glitch trigger.
- La conexión al PC se realiza mediante el puerto paralelo ó LPT estándar.
- Incluye cable LPT, puntas de prueba, software en castellano compatible con Windows 98/Me/2000/XP, manuales y 16 puntas de prueba.
- Garantía de 1 año
- Precio: 1.536,80 €

Figura 2.1

II.1.2 ANALIZADOR LÓGICO DE 8 ENTRADAS DIGITALES, ANNIE 200P, REF. KYHE 20-022

- Analizador lógico de ancho de banda de 40 Mhz con 8 entradas digitales de 16 Kbit de memoria por canal. Realiza un máximo de 200 muestras MS/seg.
- La conexión al PC se realiza mediante el puerto paralelo ó LPT estándar.
- Incluye cables LPT, puntas de prueba, software en castellano compatible con Windows 3.1/95/98/NT, manuales y 8 puntas de prueba.
- Garantía de 1 año
- Precio: 972, 50 €

Figura 2.2

II.2 Analizador Lógico de 8 canales utilizando LabView 6.0

Como se señala al inicio del presente trabajo, el objetivo fundamental del mismo radica en valorar la factibilidad de implementación o adquisición de un analizador lógico que permita el desarrollo de las habilidades prácticas en el laboratorio de Electrónica Digital.

En el epígrafe anterior se han descrito brevemente dos analizadores gráficos comerciales que utilizan una combinación de hardware-software cuyos precios no son tan excesivos pero aún distantes de las posibilidades económicas del departamento.

Además, como se puede apreciar en ambas descripciones, estos analizadores no cuentan con características que justifiquen su adquisición. Por ejemplo: el

número de canales no es tan grande (hasta 16) y la frecuencia de muestreo es relativamente alta pero no poseen facilidades para diferentes modos de disparo.

Desde hace dos años, el colectivo de Electrónica Digital ha estado intentando el desarrollo de un instrumento virtual que permita convertir, con un mínimo de recursos, cualquiera de nuestros ordenadores de laboratorio en un instrumento real-virtual en el cual los estudiantes puedan comprobar el funcionamiento de los principales circuitos digitales y proyectos que desarrollan en nuestras asignaturas.

En el Trabajo de Diploma de Humberto Eliécer Delgado López, en el Departamento de Telecomunicaciones y Electrónica de la Facultad de Eléctrica de La Universidad Central "Marta Abreu" de Las Villas, del curso 2002-2003, se diseña un instrumento virtual, desarrollado en LabView 6.0, que actúa como un analizador lógico de 8 canales utilizando las facilidades del puerto paralelo.

LabView es un programa para el desarrollo de aplicaciones al igual que otros como el C, BASIC y el National Instruments LabWindows.

LabView al igual que C o BASIC es un sistema de programación de propósito general con un sistema de librerías de funciones y subrutinas para cualquier tarea de programación. LabView también contiene librerías para aplicaciones específicas para la adquisición de datos GPIB y una serie de instrumentos de control, análisis, presentación y almacenamiento de datos.

LabView incluye también un desarrollo convencional de herramientas que permite ver como se pasan los datos a través de un programa de una forma animada.

La programación en lenguaje gráfico (G) constituye el núcleo del LabView. Es un sistema de programación de propósito general pero también incluye librerías de funciones y desarrolla herramientas diseñadas específicamente para la adquisición de datos e instrumentos de control.

Los programas en G son llamados instrumentos virtuales (VIs) debido a que su operación y apariencia puede imitar un instrumento real. Sin embargo los (VIs) son similares a las funciones de un lenguaje de programación convencional.

Los VIs son jerárquicos y modulares, se pueden usar como programas principales o como subprogramas dentro de otro programa. Cuando un VI se utilice dentro do otro VI se 1e llama subVI.

En el trabajo de diploma antes mencionado, el desarrollo del instrumento virtual se cumplió con la exigencia de que, dentro de los requerimientos de hardware, sólo se necesitara la utilización del ordenador sin ninguna tarjeta de adquisición de datos.

Fue por ello que se tomó la decisión de utilizar el Puerto Centronics (Puerto Paralelo) en el modo byte.

Para lograr esto desde el LabView fue necesario utilizar la biblioteca: **RWISAPORT. LLB.**

Esta biblioteca contiene los VIs necesarios para la lectura y escritura, desde y hacia los periféricos de entrada/salida. En este caso el VI: **Read Port NT_Win** permite la lectura desde el puerto paralelo de 8 bits de datos.

Para lograr lo anterior es necesario instalar el driver **wrisaport.sys** que permite el acceso, desde Windows NT, 200P, XP, a los dispositivos físicos de entrada salida.

La secuencia de pasos que permite instalar y registrar este driver es la siguente:

1. Ejecutar el programa REGINI.COM con la secuencia:

   REGINI WRISAPORTINI

2. Copiar wrisaport.sys hacia WI NNT\SYSTEM32\DRIVERS\
3. Reiniciar la máquina.

Los principales bloques con que fue diseñado el analizador lógico son los siguientes VIs:

1. **Read Port NT-Win:** Permite leer desde el puerto 0378H (registro de datos del puerto de impresora), entregando un dato en formato numérico entero.
2. **Number to Boolean Array:** Convierte un valor numérico a un arreglo lógico, si el valor numérico no es entero, primeramente redondea el mismo.

3. **Array to Cluster:** Convierte un arreglo unidimensional en un “cluster” en el cual cada uno de sus elementos es del mismo tipo del arreglo.
4. **Unbundle:** Desagrega un cluster en sus elementos individuales. En este caso cada uno de los elementos es un bit, ordenados de menos significativo a más significativo.
5. **Type Cast:** Operador de conversión de tipo de dato que permite convertir el valor binario (verdadero a falso) en una cadena con el objetivo de que se pueda visualizar en un VI de forma de onda.
6. **Waveform Chart:** De los varios tipos de visualizadores el seleccionado permite mostrar una forma de onda de amplitud contra tiempo que, para este diseño, se ha definido como unidad mínima de tiempo la de 10 ms.
7. **Botón Stop:** Permite detener la ejecución de la aplicación en el momento que el usuario la desee.

Las principales dificultades en el trabajo con este instrumento se encontraron en:

1. Su frecuencia de trabajo es muy baja, limitada a 10 Hz, fundamentalmente por parte del sistema operativo que controla los tiempos asignados a cada tarea.
2. Los intervalos de tiempo de muestreo no son fijos pues dependen del momento en que el sistema operativo le otorgue el tiempo a cada aplicación activa en el ordenador. Esto hace que se pierda en integridad de la forma de onda muestreada, incluso señales que son periódicas se ven aperiódicas.
3. Para una frecuencia de trabajo de 10 Hz, el muestreo de 8 canales obliga a que trabaje con señales de un frecuencia máxima cercana a 1 Hz por lo que, aplicaciones como contadores de 8 bits, registros de desplazamientos de esta longitud o máquinas de estados algorítmico de 4 o más estados requerirían varios minutos para comprobar su correcto funcionamiento.

4. Para trabajar con el analizador en laboratorio tenemos que instalar tres DLL en la computadora que vamos a trabajar y después reiniciar la computadora. Para ello cada estudiante tiene que tener acceso de administrador para instalar las DLLs en la computadora.

Las anteriores deficiencias motivaron continuar con el estudio de desarrollar un analizador lógico de elevado nivel de simplicidad en su instalación y con un mínimo de recursos de hardware. En el siguiente epígrafe se propone la variante desarrollada durante este trabajo de diploma.

II.3 Analizador lógico de 8 canales desarrollado en C y ensamblador.

Para el desarrollo de esta aplicación se continúa con la línea de trabajo de utilizar las capacidades del puerto paralelo de la computadora del PC, que proporcione una entrada de 8 bits para crear un analizador lógico útil y fácil de construir con las funciones básicas de uno real.

La desventaja más grande que la computadora tiene es su razón de muestreo. El programa fue escrito en una computadora Pentium de 75 MHz que proporcionó razones de muestreos de hasta 10 kHz.

Aunque esto no es comparable a las velocidades de los analizadores lógicos reales, esta velocidad es suficiente para muchos proyectos en los laboratorios de las asignaturas de Electrónica Digital.

Los únicos requisitos que se necesitan son la computadora, un display VGA y un puerto paralelo.

No hay ningún requisito de velocidad para la computadora, aunque mientras más alta es la razón de reloj de la computadora, más alta va a ser la razón de muestreo asequible.

La aplicación está desarrollada en lenguaje C y las subrutinas de control de la velocidad de muestreo (la programación del timer), muestreo y visualización en tiempo real se desarrollaron en lenguaje ensamblador. El programa debe ejecutarse en el modo de DOS.

II.3.1 Estrategia de desarrollo de la aplicación.

II.3.1.1 Breve análisis del puerto paralelo.

En la siguientes figuras se muestra la estructura de un puerto paralelo así como la distribución de pines en un conector DB25.

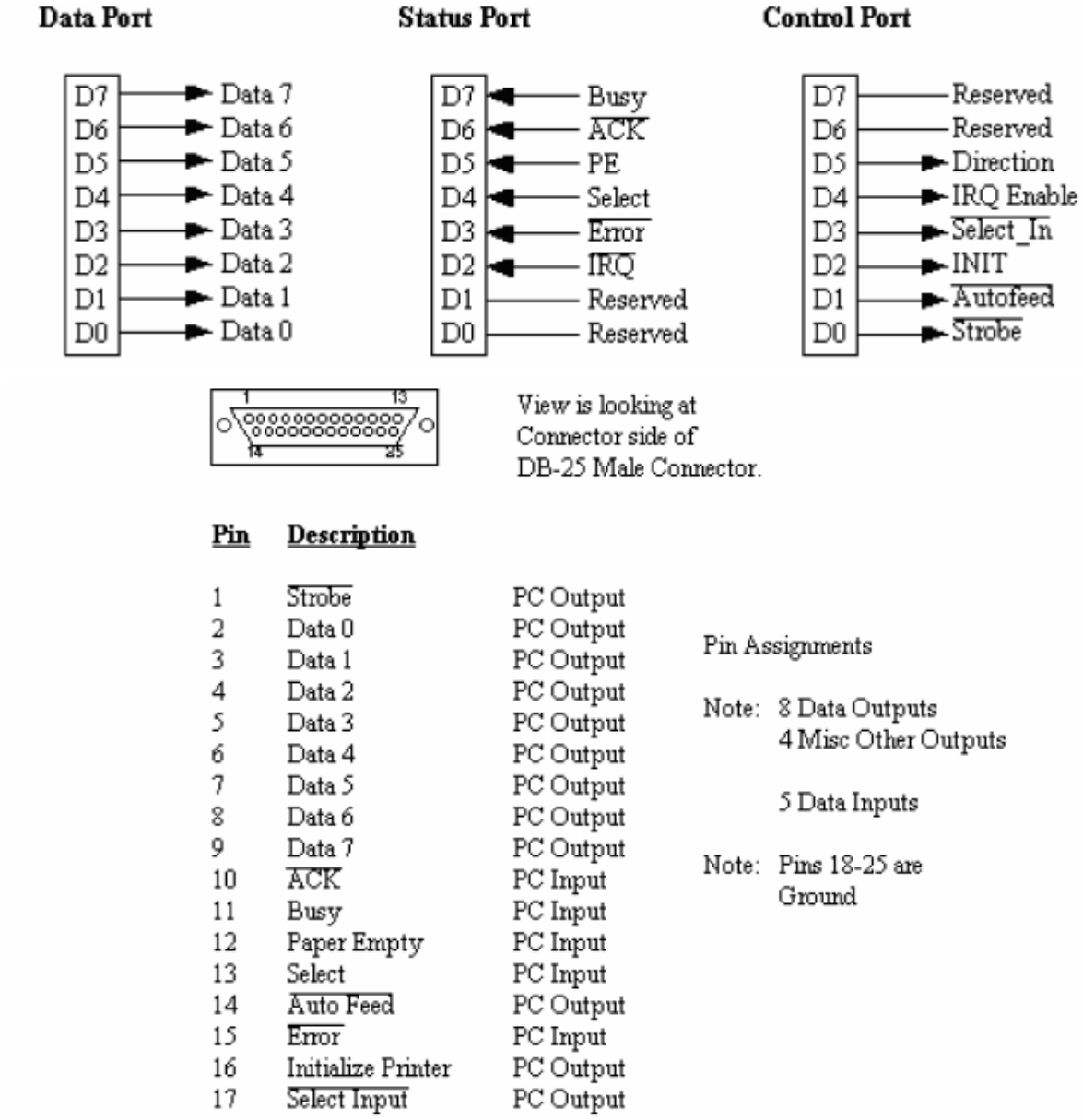

Figura 2.3 Descripción del Puerto y el Conector del puerto paralelo.

Observe que el puerto paralelo tiene 12 líneas de salida (8 líneas de datos, strobe, autofeed, init, y select input) y 5 de entrada (acknowledge, busy, paper

empty, select y error). El estándar IEEE 1284 define cinco modos de operación para este puerto:

1. Modo compatible.
2. Modo nibble
3. Modo byte
4. Modo EPP, puerto paralelo ampliado
5. Modo ECP, puerto de capacidad extendida

El objetivo del estándar es diseñar nuevos dispositivos que sean totalmente compatibles con el puerto paralelo estándar (SPP) definido originalmente por la IBM. Hay tres direcciones de E/S asociadas con un puerto paralelo de la PC. Estas direcciones pertenecen al **registro de datos**, el **registro de estado** y el **registro de control**. El registro de datos es un puerto de lectura-escritura de ocho bits. Leer el registro de datos (en la modalidad unidireccional) retorna el último valor escrito en el registro de datos. Los registros de control y estado proveen la interface a las otras líneas de E/S. La distribución de las diferentes señales para cada uno de los tres registros de un puerto paralelo se muestra en las siguientes tablas:

Dirección	Nombre	Lectura/Escritura	Bit #	Propiedades
Base + 0	Puerto de datos	Escritura	Bit 7	Dato 7
			Bit 6	Dato 6
			Bit 5	Dato 5
			Bit 4	Dato 4
			Bit 3	Dato 3
			Bit 2	Dato 2
			Bit 1	Dato 1
			Bit 0	Dato 0

Tabla 2.1 Registro de datos

Dirección	Nombre	Lectura/Escritura	Bit #	Propiedades
Base + 1	Puerto de estado	Sólo Lectura	Bit 7	Busy
			Bit 6	Acknowledge
			Bit 5	Falta de papel
			Bit 4	Select In
			Bit 3	Error
			Bit 2	IRQ (Not)
			Bit 1	Reservado
			Bit 0	Reservado

Tabla 2.3 Registro de estado

Dirección	Nombre	Lectura/Escritura	Bit #	Propiedades
Base + 2	Puerto de control	Lectura/Escritura	Bit 7	No usado
			Bit 6	No usado
			Bit 5	Permite puerto bidireccional
			Bit 4	Permite IRQ a través de la línea acknowledge
			Bit 3	Selecciona impresora
			Bit 2	Inicializa impresora
			Bit 1	Nueva línea automática
			Bit 0	Strobe

Tabla 2.4 Registro de control

Una PC soporta hasta tres puertos paralelo separados, por tanto puede haber hasta tres juegos de registros en un sistema, en un momento dado. Existen tres **direcciones base** para el puerto paralelo asociadas con tres posibles puertos paralelo: 0x3BCh, 0x378h y 0x278h, que constituyen las direcciones base para el puerto **LPT1**, **LPT2** y **LPT3**, respectivamente. El registro de datos se localiza siempre en la dirección base de un puerto paralelo, el registro de estado aparece en la dirección base + 1, y el registro de control aparece en la dirección base + 2. Por ejemplo, para un puerto LPT2 localizado en 0x378h, ésta es la dirección del registro de datos, al registro de estado le corresponde la dirección 0x379h y su respectivo registro de control está en la dirección 0x37Ah.

Cuando la PC se enciende el BIOS ejecuta una rutina para determinar el número de puertos presentes en el sistema asignando la etiqueta LPT1 al primer puerto localizado, si existen más puertos entonces se asignarán consecutivamente las etiquetas LPT2 y LPT3 de acuerdo a la siguiente tabla:

Dirección inicial	Función
0000:0408	Dirección base para LPT1
0000:040A	Dirección base para LPT2
0000:040C	Dirección base para LPT3
0000:040E	Dirección base para LPT4

Tabla 2.5 Direcciones base en el BIOS

Como se observa en la tabla 2.4, escribiendo en el registro de control del puerto paralelo, un 1 en el bit 5 se puede pasar el registro de datos al modo bidireccional.

Eléctricamente, el puerto paralelo entrega señales TTL y como tal, teóricamente, se le puede conectar cualquier dispositivo que cumpla con los niveles de voltaje específicos de la lógica TTL, sin embargo el hardware del puerto paralelo está muy limitado en cuanto a su capacidad de corriente, por esta razón se debe ser muy cuidadoso con el manejo de las señales del puerto, un corto circuito puede dañar permanentemente la tarjeta madre de la PC.

Para disminuir lo más posible el riesgo de daños al puerto utilizamos un circuito integrado **74LS244** como etapa separadora y al mismo tiempo mejoramos la capacidad de manejo de corriente, aunque se debe señalar que en nuestra aplicación el puerto se programa en modo bidireccional y sólo se utilizará como entrada. El circuito se muestra en la siguiente figura:

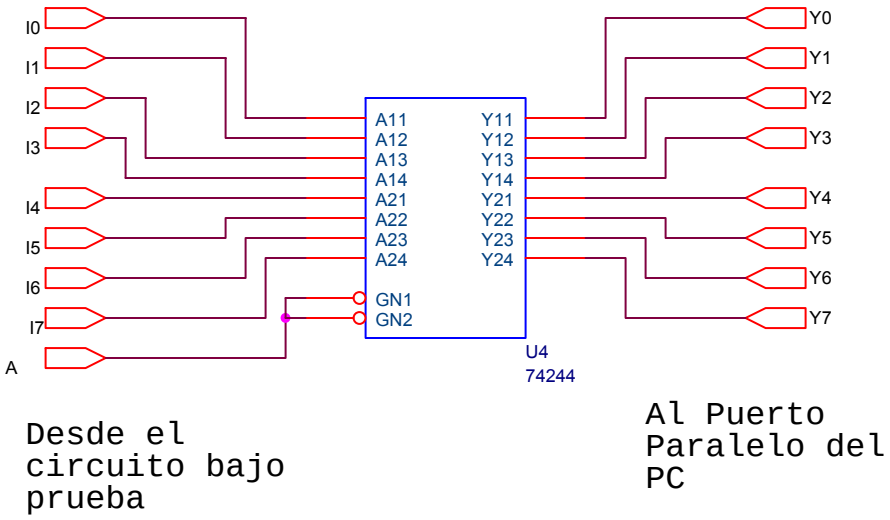

Figura 2.4 Circuito de protección del puerto paralelo.

II.3.1.2 Breve análisis del temporizador.

La razón de muestreo se controla utilizando el 8253 o por temporizadores equivalentes que posee la computadora.

La frecuencia de la entrada a los 8253 es aproximadamente 1.19Mhz. Los 8253 tienen 3 divisores o temporizadores de frecuencia. El temporizador 0 se usa para proporcionar un señal de temporización a 18Hz con la finalidad de cronometrar.

El temporizador 1 se utiliza para proporcionar la señal de refrescamiento de memoria y el temporizador 2 se para controlar el altavoz interior y otros dispositivos.

El temporizador 0 se conecta a IRQ0 al 8259 PIC que a su vez llama a la INT 08, que a su vez llama a la INT 1C. Normalmente la computadora utiliza la interrupción 08 para el reloj de tiempo.

Mediante la reprogramación del temporizador 8253 y rescribiendo la INT 1C es posible controlar la razón de muestreo.

II.3.2 Implementación.

Teniendo en cuenta los análisis de los dos epígrafes anteriores, se realizó un programa en C (Borland C) – ensamblador que, utilizando como señal para el muestreo el temporizador de la computadora y como entrada de datos el puerto

paralelo, permita cumplir con las exigencias de un analizador lógico de 8 canales con muestreo y visualización en tiempo real; así como una posterior visualización de las señales muestreadas almacenadas en un buffer de memoria.

La fuente de dicho programa y sus comentarios se muestran a continuación:

```
#ifdef __TINY__
#error LA8bits will not run in the tiny model.
#endif
#include <dos.h>              // Para leer desde LPT1
#include <math.h>
#include <conio.h>            // Para leer desde teclado
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <graphics.h>         // Para mostrar las forma de onda
#define ESC     0x1b                /* Define la tecla escape       */
#define TRUE    1                   /* Define  constantes  */
#define FALSE   0                   /* Define constantes  */
#define ON      1                   /* Define  constantes  */
#define OFF     0                   /* Define constantes  */
#define MAX 15000
#define NFONTS 11
char *Fonts[NFONTS] = {
"DefaultFont",      "TriplexFont",      "SmallFont",  "SansSerifFont",  "GothicFont",
"ScriptFont",         "SimplexFont",         "TriplexScriptFont",         "ComplexFont",
"EuropeanFont", "BoldFont"
};
char *LineStyles[] = {
  "SolidLn",  "DottedLn",  "CenterLn",  "DashedLn",  "UserBitLn"
};
```

```
char *FillStyles[] = {
"EmptyFill", "SolidFill", "LineFill", "LtSlashFill", "SlashFill", "BkSlashFill",
"LtBkSlashFill", "HatchFill", "XHatchFill", "InterleaveFill", "WideDotFill",
"CloseDotFill"
};
char *TextDirect[] = {
"HorizDir", "VertDir"
};
char *HorizJust[] = {
  "LeftText", "CenterText", "RightText"
};
char *VertJust[] = {
  "BottomText", "CenterText", "TopText"
};
struct PTS {
  int x, y;
};      /* Structure to hold vertex points      */
int    GraphDriver;            /* Driver Gráfico        */
int    GraphMode;              /* Modo Gráfico          */
double AspectRatio;            /* Razón de aspecto de los pixels de pantalla*/
int    MaxX, MaxY;             /* Máxima resolución de pantalla */
int    MaxColors;              /* Máximo número de colores dsiponible    */
int    ErrorCode;              /* Reporte de errores gráficos        */
struct palettetype palette;    /* Información de paleta de colores    */
```

```
void Initialize(void);                          // Inicializa en modo gráfico
void setup(void);                               // Inicializa pantalla, interrupciones, etc...
char read_key(void);                            // Rutina de lectura de teclado
void set_sample_rate(int change);      // Rutina de cambio de razón de muestreo
void sampling_control(int control);         // Habilita/Deshabilita/Cambia muestreo
void view_waveforms(void);                  // Muestra contenido del buffer
void draw_waveforms(float scale,int startwave); // Dibuja las formas de onda
void clear_waveforms(void);           // Limpia las ondas del display
void unsetup(void);                  // Devuelve todo al modo original
void clearmenu(void);                 //Borra el menu del display
void mainmenu(void);                  // Muestra el menu principal
int scancode(void);                  // Funcion para leer el código de barrido

void interrupt sample(...);           // Cambia el temporizador para muestrear dato
void interrupt (far *oldVector)(...);     // Salva la dirección original de INT 0x1c
int huge data[MAX];             // Reserva 15k de espacio de datos
int index;                                      // Indice del arreglo
int printerport;                  // Dirección del Puerto de impresora

void main(void)
{
      Initialize();               /* Inicializa el sistema en modo gráfico */
         char key=0;                     // buffer de la tecla
         if( GraphDriver==EGA || GraphDriver==EGALO || GraphDriver==VGA )
```

```
setup();                    // rutina de inicialización
sampling_control(0);        //Deshabillita el muestreo
while(key!=27) {            // Lazo del programa principal
        key=' ';            // Resetea key
        while(kbhit())          // Chequea por telca pulsada
                key=read_key();         // Devuelve el valor en key
        if (key=='s' || key=='S')       // Cambio de razón de muestreo
                sampling_control(-1);       // -1 cambia muestreo
        if (key=='R')                       // Incrementa razón de muestreo
                set_sample_rate(100);       // en razón de 100
        if (key=='r')               // Decrementa razón de muestreo
                set_sample_rate(-100);      // en razón de 100
        if (key=='v' || key=='V') {     //muestra las formas de onda del buffer
                clearmenu();            // Limpia del display el menú
                view_waveforms();       // pasa el control a view_waveforms()
                clearmenu();            // limpia el menú anterior
                mainmenu();             / muestra el menú principal de nuevo
        }
        if (key=='c' || key=='C') {     // Limpia los datos del buffer
                clear_waveforms();      // Limpia la forma de onda del display
        for(index=0;index<MAX;index++)// Limpia el arreglo completo
                        data[index]=0;
                index=0;
```

```
                }
        }
        unsetup();                          // Restablece todas las condiciones
inciales
}
                                            // y sale
void Initialize(void)
{
  int xasp, yasp;                   /* Utilizadas para leer la razón de aspecto*/
  GraphDriver = DETECT;              /* Para solicitar auto-detection       */
  initgraph( &GraphDriver, &GraphMode, "A:\\BC\\BGI" );
  ErrorCode = graphresult();          /* Lee los resultados de la inicialización*/
  if( ErrorCode != grOk ){            /* Error ocurrido durante la incialización    */
    printf(" Graphics System Error: %s\n", grapherrormsg( ErrorCode ) );
    exit( 1 );
  }
  getpalette( &palette );             /* Lee la configuración de la paleta*/
  MaxColors = getmaxcolor() + 1;       /* Lee el número máximo de colores*/
  MaxX = getmaxx();
  MaxY = getmaxy();                   /* Lee el máximo valor de display          */
  getaspectratio( &xasp, &yasp );      /* Lee razón de aspecto de hardware     */
  AspectRatio = (double)xasp / (double)yasp; /* Lee factor de corrección   */
}
void setup(void)
{
```

```
        setcolor(9);                    //  bright blue
        outtextxy(240,0,"8-bit Logic Analyzer");
        outtextxy(240,10,"Written by Curtis Dodds");
        setcolor(7);                    // Set color back to white
        outtextxy(0,470,"Sample Rate:");
        mainmenu();                     // Muestra menú

// obtiene la dirección del Puerto desde la dirección 0000:0408
        printerport=peek(0,0x408);      // dirección base
// coloca el Puerto en modo bidireccional
        outp(printerport+2, inp(printerport+2)|0x20);
// Establece la razón de muestreo al valor implícito 100Hz
        set_sample_rate(0);             // 0 = 100Hz
// Captura y salva el valor de interrupción  0x1c
        disable();                      // Deshabilita las interrupciones de hardware
        oldVector=getvect(0x1c);        // Obtiene el valor original de la int 0x1c
        setvect(0x1c,sample);   // Cambia esta a que apunte a la subrutina sample
        enable();                       // habilita las interrupciones de hardware
}
// Restaura la interrupción 0x1c y el modo de video
void unsetup(void)
{
        disable();                      // Deshabilita las interrupciones de hardware
        set_sample_rate(18);        // Establece el tick del timer a su valor inicial
```

```
        setvect(0x1c,oldVector);   // Restablece el valor original del vectos int 0x1c
        sampling_control(1);              // deshabilita el muestreo
        enable();                        // Habilita las interrupciones de hardware
        closegraph();                     // Restaura el modo de video original
}

//Lee el carácter ASCII desde teclado
char read_key(void)
{
        char key=0;                       // clear value
        key=getch();                      // get keypress
        return (key);                     // return value
}
// Lee el código de barrido desde teclado
int scancode(void)
{
        char code=0;
                        asm {  mov ah,1              // Check if key is in buffer
                                int 0x16             // keyboard bios interrupt
                                mov code,ah          // else get code
                                mov ah,0             // clear keyboard buffer
                                int 0x16             // keyboard bios interrupt
                        }
                        return(code);               // return scancode
```

```
}
// Ajusta la razón de muestreo
void set_sample_rate(int change)
{
        int number;                     // sample rate per second
        int ratehi,ratelo;              // hi and lo byte to send to timer
        static int rate=100;            // static variable sample rate
        char speed[10];                 // string to hold speed value
        rate=rate+change;               // increment rate by passed parameter
        if (rate<100)                   // impide razones de muestreo inferiors a  100Hz
                rate=100;
        if (rate>5000)                  // Impide rezones de muestreo superiors a  5kHz
                rate=5000;
        number=65536/(rate/18.2);       // calcula el divisor
        ratehi=number&0xff00;           // calcula la parte alta del byte para el timer
        ratehi=ratehi/256;
        ratelo=number&0x00ff;           // calcula la parte baja del byte para el  timer
        if (change==18) {               // si change=18 resetea al tick original
                ratehi=0xff;
                ratelo=0xff;
        }
outportb(0x43,0x36); // programa el 8253 timer: generador de onda cuadrada
outportb(0x40,ratelo);          // byte bajo del divisor
outportb(0x40,ratehi);          // byte alto del divisor
```

```
        itoa(rate,speed,10);          // convierte a entero para mostrar como cadena
        setcolor(0);                  // cambia color de la línea a negro
        disable();                    // deshabilita las interrupciones de hardware
        for(number=468;number<476;number++) // Borra el número que muestra l
                                                // la razón de muestreo
                line(100,number,140,number);     // del display
        setcolor(7);                  // Coloca el color de la línea de nuevo a blanco
        outtextxy(100,470,speed);          // Muestra la sampling rate speed
        enable();                     // Habilita las interrupciones de hardware
}

// Aquí tiene lugar la captura de las señales
void interrupt sample(...)
{
        int y;                        // coordenada y del display
        static int x=0;               // coordenada x
        int port;                     // variable de almacenamiento temporal
        if (index==0)                 // si index=0 , x coordinate=0
                x=0;
        if (index>MAX) {              // si buffer está lleno, x coordinate=0
                x=0;                  // envia mensaje
outtextxy(275,100,"Buffer Full!!");
outtextxy(120,110,"Please press V to view waveforms or C to clear buffer");
                sampling_control(0);                  // Deshabilita el muestreo
```

```
        }

        if (x==640) {                   // si se alcanza el fin del display, coloca
                x=0;                    // de vuelta al inicio izquierda
                clear_waveforms();      // Borra las formas de ondas
        }

// Muestra las trazas en pantalla
no_scroll:      asm {push ax                    // save all registers used
                                push bx
                                push cx
                                push dx
                                push si
                                push di
                                mov y,100               // Lee información
                                mov dx,printerport      // desde el Puerto de datos
                                in al,dx
                                mov port,ax             // salva en la variable port
                                mov cx,8                // Contador para 8 bits
                                mov bl,128              // Máscara  para el MSB primero
                }

back: asm { push cx             // Salva el contador de lazo
                                mov si,0                // Coloca el y (offset) a 0
```

```
                    test al,bl          // Compara el bit con la máscara
                    jz skip             // si está en 0 lo deja en 0
                    mov si,25           // de lo contrario y toma 25
                }
skip: asm { push ax         // Salva el dato tomado del puerto
                    mov ah,0x0c         // función para dibujar un pixel
                    mov cx,x            // coordenada x
                    mov dx,y            // coordena y
                    sub dx,si           // substrae el offset de y
                    mov bh,0            // page 0
                    mov al,7            // color 7
                    int 0x10            // video bios interrupt
                    add y,50            // apunta a la nueva traza
                    shr bl,1            // shift la máscara
                    pop ax              // restaura el dato leído
                    pop cx              // restaura el contador
                    loop back           // cierra el lazo mientras cx <> 0
                    inc x               // incrementa el valor de x
                    inc index           // incrementa el indice del arreglo
                    pop di
                    pop si
                    pop dx              // restaura los valores
                    pop cx
                    pop bx
```

```
                        pop ax
                }
data[index-1]=port;             // guarda el dato muestreado
}

// Esta función controla el muestreo controlando el 8259
void sampling_control(int control)
{
// 0 = disable sampling
// 1 = enable sampling
//-1 = toggle sampling
        int y;
                if (control==0)             // deshabilita el muestreo
                        outport(0x21,inport(0x21)|1);
        if (index<MAX) {                // si el buffer no está lleno entonces
                if (control==1)             // sigue adelante y habilita el muestreo
                        outport(0x21,inport(0x21)&254);  // Habilita el muestreo
                if (control==-1)
                        outport(0x21,inport(0x21)^1); // cambia el muestreo
        }
        setcolor(0);            // Borra todo lo anterior escrito y pone negro display
        disable();                  // deshabilita las interrupciones de hardware
        for(y=468;y<476;y++)            // limpia el mensaje de sampling
                line(214,y,285,y);
```

```
        setcolor(7);                    // vuelve el color a blanco

if ((inport(0x21)&1)==0)    // lee el  status de la máscara del registro del  8259
        outtextxy(150,470,"Sampling Enabled");
        else
                outtextxy(150,470,"Sampling Disabled");
        enable();                       //habilita las interrupciones de hardware
}

// Limpia todas las trazas del display
void clear_waveforms(void)
{
                asm { push ds      // Limpia la memoria de video que almacena las
trazas
                                mov ax,0xa000           // con 0's
                                mov ds,ax
                                mov bx,4800
                                mov ax,0
                                mov cx,0x4000
                        }
repeat: asm {mov [bx],ax
                                add bx,2
                                loop repeat
                                pop ds
                 }
```

```
}
//Esta función muestra las trazas en el display con las facilidades
// de scroll,  zoom in y zoom out
void view_waveforms(void)
{
        char key=0;                     // buffer de teclado
        int y;                          // variable de lazo
        float scale=1;                    // variable de escala para el zoom
        int startwave=0;       // variable índice para iniciar el muestreo de las ondas
        sampling_control(0);          // Deshabilita el muestreo
        clear_waveforms();                // Borra todo
        draw_waveforms(scale,startwave);        // Muestra las trazas en el display
        setcolor(4);                    // pasa a color rojo
       outtextxy(310,427,"ESC-go back.");//muestra el menu de view_waveforms()
        outtextxy(455,457,"Left Arrow-Pan Left.");
        outtextxy(455,470,"Right Arrow-Pan Right.");
        outtextxy(290,458,"Up Arrow-Zoom In.");
        outtextxy(290,470,"Down Arrow-Zoom Out.");
        while(key!=1) {                 // se mantiene en esta función
                if (kbhit()!=0)               // si la tleca ESC key no se pulsa
                        key=scancode();             // toma el código
                if (key==0x48) {             // si key=up arrow entonces hace zoom out
                                            // en un factor 2 (duplica) hasta 16
```

```
                    if (scale<16) {
                            scale*=2;
                            draw_waveforms(scale,startwave);  // dibuja las ondas
                    }
                    key=' ';                    // limpia la variable key
            }
            if (key==0x50) {                // si key=down arrow entonces zoom in
        // en un factor de 2 (divide por 2) hasta 1/4
                    if (scale>.25) {
                            scale/=2;
                            draw_waveforms(scale,startwave);  // dibuja las ondas
                    }
                    key=' ';                    // limpia la variable key
            }
        if (key==0x4b) {                // si  key=left arrow y las ondas
                    if (startwave!=0) {    // pueden aún desplazarse a la izquierda
                            startwave=startwave-50;    // la desplaza
                            if (startwave<0)
                                    startwave=0;
                            draw_waveforms(scale,startwave); // dibuja las ondas
                    }
                    key=' ';                    // limpia la variable key
            }
if (key==0x4d) {                //si key=right arrow y las ondas
```

```
if (startwave!=(index-640)&&(index>640)) { // pueden desplazarse a la derecha
                        startwave=startwave+50;          // las desplaza
                        if (startwave>(index-640))
                                startwave=(index-640);
                        draw_waveforms(scale,startwave);  // dibuja las ondas
                }
                key=' ';                    // limpia la variable key
        }
    if (key==0x73) {              // si key = CTRL+left arrow entonces
                if (startwave!=0) {   // desplaza las ondas 5 veces más rápido
                        startwave=startwave-250;   // a la izquierda
                        if (startwave<0)
                                startwave=0;
                        draw_waveforms(scale,startwave);  // dibuja las ondas
                }
                key=' ';                    // limpia la variable key
        }
if (key==0x74) {              // si key = CTRL+right arrow entonces
if (startwave!=(index-640)&&(index>640)) {//mueve las ondas 5 veces mas rápido
                        startwave=startwave+250;      // a la derecha
                        if (startwave>(index-640))
                                startwave=(index-640);
                        draw_waveforms(scale,startwave); // dibuja las ondas
                }
```

```
                key=' ';                    // limpia la variable key
            }

        }
        clear_waveforms();              //limpia el display
}

//Esta función realiza el dibujo de las formas de onda
void draw_waveforms(float scale,int startwave)
{
        int x;                      // coordenada x
        int xpoint;                   // variable para almacenar el cálculo
        int y;                      // coordenada y
        int offset;                   // offset de la coordenada y al valor de 0 ó 1
        int bitmask=128;                // empieza con el MSB
        int end;                    // fin de la onda
        int done=0;                   // bandera para dibujar el pixel
        clear_waveforms();        // borra las ondas anteriores
        setcolor(7);                  // Establece color de línea a blanco
        setlinestyle(1,0,1);            // Set linestyle to dotted line
        for(y=100;y<=457;y+=50)           // dibuja las líneas horizontales
            line(0,y,639,y);            // representando el valor bajo
            setlinestyle(0,0,1);            // coloca el estilo de línea en sólido
            setcolor(9);                  // coloca el color a azul brillante
```

```
        if ((startwave+639*scale)>index)    // Si se alcanczó el final del display
                end=index;                  // limita el lazo
        else
                end=startwave+(639*scale);      // de lo contrario continúa
        for (x=startwave;x<end;x++) {       // Dibuja las trazas a lo largo del display
                xpoint=(x-startwave)/scale;
                y=100;                              // Comienza con la traza MSB
                for (bitmask=128;bitmask>=1;bitmask/=2) {       // Chequea bit a bit
                        if ((data[x]&bitmask)>0)                // si el bit es 1
                                offset=-25;     //sube el offset de y en el display
                        else
                                offset=0;                       // si no lo deja igual
                if ((data[x]&bitmask)!=(data[x-1]&bitmask))   // si el bit actual es
                        line(xpoint,y,xpoint,y-25);     // diferente al anterior
                                                        // cambia de alto a bajo
                        if(xpoint!=done)
                                putpixel(xpoint,y+offset,9);        // dibuja la traza
                        y+=50;                              // va a la próxima traza
                    }
                done=xpoint;            // actualiza la bandera para asegurar que solo
        }                           // un pixel sea dibujado en cada localización x
}
//Esta función limpia el menu del display
void clearmenu(void)
```

```
{
        int y;
        setcolor(0);
        for (y=460;y<475;y++)
                line(285,y,639,y);
}
void mainmenu(void)
{
        setcolor(4);
        outtextxy(240,100,"ESC-Exit Program");
        outtextxy(240,130,"S-Toggle Sampling");
        outtextxy(240,160,"V-View Still Traces");
        outtextxy(240,190,"R-Increase Sampling Rate.");
        outtextxy(240,220,"r-Decrease Sampling Rate.");
        outtextxy(240,250,"C-Clear Buffer");
}
```

Para la edición, compilación, enlace y ejecución del programa se utilizó el Borland C. Luego se creó un disco sistema DOS al cual se copiaron los siguientes ficheros y/o carpetas:

1. LA8bits.exe Programa para analizador lógico de 8 bits ejecutable
2. LA8bits.cpp Programa fuente
3. Bc (carpeta) Carpeta con los drivers para BGI del C.
4. leeme.txt Fichero texto con las orientaciones para ejecutar el programa.

En la figura siguiente se muestran los resultados de la ejecución del programa bajo DOS. Como se puede observar, los resultados son satisfactorios y permiten garantizar un medio sencillo y de bajo costo para la realización de prácticas de laboratorio de Electrónica Digital.

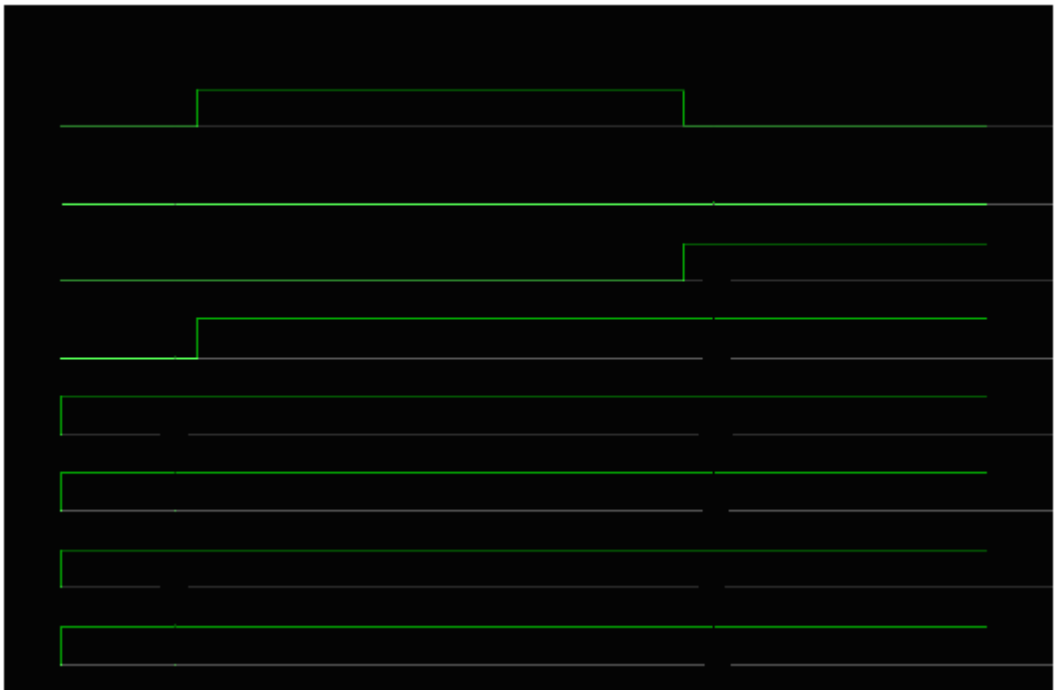

Figura 2.5 Display del analizador lógico

CAPITULO III: Desarrollo de actividades prácticas para Electrónica Digital utilizando el analizador lógico de 8 bits.

III.1 Introducción

Tanto los últimos resultados de las autoevaluaciones de las asignaturas Electrónica Digital I y II, como los de las encuestas del Proyecto MES: "Generalización de experiencias en la enseñanza de la Electrónica", que tenía como objetivos fundamentales fomentar el intercambio entre los diferentes colectivos de profesores que imparten las asignaturas de esta disciplina, así como aportar y/o aplicar las mejores experiencias de cada centro del país; reflejan deficiencias en la base de hardware para el desarrollo de habilidades prácticas en el montaje y comprobación de circuitos digitales combinacionales y secuenciales.

Las tablas que a continuación se muestran (ver indicadores en el Anexo 1) son el resultado de dicha encuesta y, como se puede observar nuestro centro es uno de los que tiene peores resultados en el punto 7, relacionado con el aseguramiento de hardware para los laboratorios reales.

Tabla 3.1 Resumen de los criterios de las demás universidades sobre la UCLV en Digital I.

Centro	1	2	3	4	5	6	**7**	8	9	10
ISPJAE	10	8	10	10	7	10	**7**	8	8	8
UPR	9	10	10	9	10	9	**NE**	10	8	10
UO	9	10	10	10	9	10	**NE**	10	9	10
UC	10	10	10	10	9	8	**NE**	10	10	9
Prom.	**9,5**	**9,5**	**10**	**9,75**	**8,75**	**9,25**		**9,5**	**8,75**	**9,25**

Tabla 3.2 Resumen de los criterios de las demás universidades sobre la UCLV en Digital II.

Sobre la UCLV.

Centro	1	2	3	4	5	6	**7**	8	9	10
CUJAE	7	9	10	7	7	10	**6**	10	10	10
UPR	9	10	10	7	9	10	**4**	8	8	9
UO	9	10	10	8	9	10	**7**	9	7	8
UC	10	10	10	10	9	8	**5**	10	10	9
Promedio	**8.75**	**9.75**	**10**	**8**	**8.5**	**9.5**	**5.5**	**9.25**	**8.75**	**9**

Una encuesta realizada recientemente, al finalizar el 2do. semestre de 3er año de Telecomunicaciones y Electrónica (ver Anexo 2), también muestra un grupo de insatisfacciones relacionadas con ampliar el número de laboratorios reales. Los resultados de la misma se muestran resumidos en la siguiente tabla:

Preguntas	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Sí	30	-	34	-	7	30	-	34	27	34	34	15	27	30	27	34	12
No	-	-	-	34	27	-	15	-	-	-	-	19	-	-	-	-	-
En Cierta Medida.	-	-	-	-	-	4	-	-	7	-	-	-	7	4	7	-	9
Desconozco	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Adecuada	-	30	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
No Adecuada	-	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
No me Interesa	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
No me es Necesario	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Me Molesta	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Sí, pero formando parte de un equipo.	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	13

Tabla 3.3 Resultados de la encuesta

A continuación se propone un conjunto de medios que permitirán solucionar estas deficiencias:

1. Uno (1) ó dos (2) ordenadores personales de prestaciones medias (Pentium, Pentium II) que se ubicarían uno en cada subsección del laborarorio 308. Los requisitos de hardware de estos son mínimos: 64 Mb de RAM, sistema operativo Windows 98 y HDD de 1 Gb o menor, incluso sin HDD; un monitor VGA y teclado, incluso sin mouse.
2. 2 Discos sistema DOS, con el programa LA8bits.exe y los drivers BGI.
3. 2 Circuitos de interface con el 74244 para la protección del puerto de impresora.

4. 2 Contadores de 4 bits para la generación de todas las posibles combinaciones de entrada de forma cíclica.
5. 2 Generadores de frecuencia.
6. En cada puesto de trabajo el estudiante tendrá una breadboard donde montará el circuito bajo prueba a ser comprobado en uno de los puestos de cada subsección del laboratorio.

A continuación se proponen un conjunto de prácticas combinacionales y secuenciales que pueden realizarse con este conjunto mínimo de medios.

III.2 Ejemplos de prácticas con sistemas combinacionales.

Tema III: Diseño de sistemas digitales combinacionales uni y multiterminales con componentes SSSI.

Práctica de Laboratorio: Diseño, montaje y comprobación de decodificadores y multiplexores.

Materiales:

- Circuitos Integrados: 7410 (2), 7404 (1), 7408(3), 7432 (2)

Técnica Operatoria:

1. Diseñe, realice el montaje y compruebe el funcionamiento de un decodificador de 2 a 4 y entrada de habilitación con componentes SSI. Compruebe el funcionamiento del mismo en uno de los dos analizadores lógicos del laboratorio, ajuste la frecuencia del reloj a 1 KHz y tome dos de las salidas del contador (menos significativas) para excitar las dos entradas de decodificación.
2. Diseñe, realice el montaje y compruebe el funcionamiento de un multiplexor de 4 a 1 y entrada de habilitación con componentes SSI. Compruebe el funcionamiento del mismo en uno de los dos analizadores lógicos del laboratorio, ajuste la frecuencia del generador a 1 KHz y tome dos de las salidas del contador (menos significativas) para excitar las dos

entradas de selección. Establezca valores fijos de voltaje en las entradas I0 a I3.

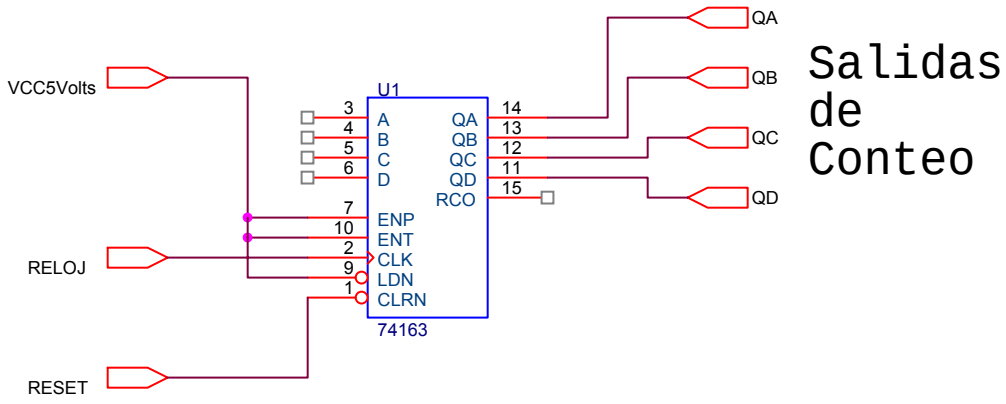

Figura 3.1 Esquema circuital del contador para generar las combinaciones de entrada.

Tema IV: Diseño de sistemas sistemas digitales combinacionales con componentes MSI

Práctica de Laboratorio: Diseño, montaje y comprobación de funciones uni y multiterminales utilizando decodificadores y multiplexores.

Materiales:

- Circuitos Integrados: 74139 (1), 7400 (2), 7420 (1), 74151 (1), 7432 (2)

Técnica Operatoria:

1- Diseñe, realice el montaje y la comprobación de la función F(C, B, A) = ∑ (0, 1, 4, 7) utilizando decodificadores y compuertas lógicas. Compruebe el funcionamiento en el analizador lógico, utilizando como entradas de excitación las 3 salidas menos significativas del contador. Ajuste la frecuencia del generador 1 KHz.

2- Implemente la función anterior utilizando un multiplexor y compuertas lógicas. Compruebe su funcionamiento en el analizador lógico.

III.3 Ejemplos de prácticas con sistemas secuenciales.

Tema I: Diseño de sistemas secuenciales asincrónicos con componentes SSI.

Práctica de Laboratorio: Diseño, montaje y comprobación de aplicaciones típicas de CSA con componentes SSI.

Materiales:

- Circuitos Integrados: 7400 (2)

Técnica Operatoria:

1- Diseñe, realice el montaje y la comprobación del CSA para controlar el encendido y apagado de una motobomba de agua para llenar un tanque de agua que posee dos sensores X1 y X2 en su interior (ver detalles del problema en el sitio: http://digital.fie.uclv.edu.cu).

2- Compruebe el funcionamiento en el analizador lógico, utilice las salidas del contador para evitar el efecto de rebote de los interruptores, tome para las entradas X1 y X2 las salidas QA y QB del contador, explique. Seleccione como frecuencia de reloj del generador 1 KHz.

Tema III: Diseño y análisis de sistemas secuenciales sincrónicos con componentes SSI.

Práctica de Laboratorio: Diseño, montaje y comprobación de CSS con biestables.

Materiales:

- Circuitos Integrados: 7474 (1), 74113 (1), 7408 (1), 7432 (1), 7404 (1)

Técnica Operatoria:

1. Ajuste la frecuencia de reloj del generador a 1 KHz.
2. Realice el montaje y comprobación de un biestable D y un JK.
3. Diseñe, realice el montaje y comprobación de un divisor por 2 de la frecuencia del reloj utilizando un biestable D. Compruebe su funcionamiento utilizando el analizador lógico.

4. Diseñe, realice el montaje y comprobación de un contador módulo 4 utilizando biestables JK.

Tema IV: Diseño y análisis de sistemas secuenciales sincrónicos con componentes MSI.

Práctica de Laboratorio: Diseño, montaje y comprobación de CSS con registros y contadores.

Materiales:

- Circuitos Integrados: 74194 (1), 74162 (1), 7404 (1)

Técnica Operatoria:

1. Ajuste la frecuencia de reloj del generador a 1 KHz.
2. Realice el montaje y la comprobación de un contador de anillo de 4 bits.
3. Realice el montaje y la comprobación de un contador Johnson de 4 bits.
4. Realice el montaje y la comprobación de un contador decimal.
5. Modifique el contador anterior para obtener un contador módulo 5.

CONCLUSIONES

Como conclusiones del presente trabajo se pueden plantear:

1. El objetivo fundamental del mismo fue cumplido, al contarse con un analizador lógico de 8 bits de bajo costo, fácil instalación y manejo que permite desarrollar un amplio conjunto de prácticas de laboratorio en las asignaturas de Electrónica Digital de pregrado.
2. La aplicación del mismo es inmediata y no se requiere de recursos adicionales para su difusión al resto de las universidades cubanas, si se tiene en cuenta que las carreras de Telecomunicaciones-Electrónica y Automática cuentan con un programa de apoyo nacional de medios computacionales, único elemento necesario para la explotación del analizador lógico diseñado.
3. La utilización del lenguaje C y el ensamblador para la creación del programa, ejecutable desde DOS permite un acceso a los recursos de hardware de modo eficiente y elevar la frecuencia de muestreo para obtener mejor integridad de las formas de onda a observar.

BIBLIOGRAFIA

1. Pallás Areny, R. Adquisición y distribución de señales. ISBN 84-267-091 8-4. Editorial Marcombo Barcelona, España 1993. pp 1-24, 27-35, 86-155, 255-283
2. Jones, L. D, Foster Ch, A. "Electronic Instruments and Measurements". ISSN 0422-910X. Editorial John and Wiley& Sons. New York. 1983 Cap II, pp 242-260
3. Data Acquisition Products (ADS 12 entry-level data acquisition card). http://www.humusoft.cz/datacq/ad512.htm. 23/04/02.
4. PCI9111 100K A-D Low Cost Multi-function DAS Card. http://www.technet.co.kr/business/Test&mesurment/industrialcomputer/pcidataIO/pci 911l.htm. 3/04/02
5. National Instruments. Measurement and Automation Catalogue. 1999. pp 258260.
6. National Instruments. Measurement and Automation Catalogue. 1999. pp 249251.
7. Upgrading & Repairing PCs Eighth Edition. Lesson5 Bus Slots and I/O Cards. http://cma.cdnet.com/book/upgraderepair /chO5.htm. 24/04 02.
8. El puerto paralelo de la PC. http://modelo.edu.mx/univ/virtech/circuito/paralelo.htm#uno#uno. 05/05/02.
9. Zamora Orozco,C. EL PUERTO PARALELO DE LA PC.carlos@creaturoides.com.
10. Peter Anderson, H. Use of a PC Printer Port for Control and Data Acquisition pha@eng.morgan edu. Department of Electrical Engineering Morgan State University. http://www. Access.diex.net/~pha. 10/05/02.
11. Parallel (PC). http://www.blackdown.org/~hwb/co-ParallelPC.html. 10/05/02
12. Interfacing the Enhanced Parallel Port. http://www.senet.com.au/~cpeacock.
13. Hardware interno del PC (III). http://www.ucm.es/info/electron/laboratorio/instrumentos/pc.htm. 12 /05/ 02.

14. Parallel Port Complete: Programming, Interfacing, and Using the PC's Parallel Printer Port Jan Axelson 1996, Lakeview Research pp. 343

Anexo 1. Aspectos a tener en cuenta para la evaluación en los tele talleres:

1. Cumplimiento de las indicaciones del plan de estudio.
2. Actualidad de las temáticas tratadas de acuerdo a la evolución de la Tecnología Electrónica.
3. Actualidad de las temáticas tratadas de acuerdo a la evolución de la Electrónica como objeto de estudio. Se toma como referencia lo que se hace en centros universitarios de reconocido prestigio internacional.
4. Orden en que son tratados los temas.
5. Actualidad de la bibliografía utilizada.
6. Utilidad de los recursos de software que se utilizan. Contribución potencial al cumplimiento de los objetivos de la asignatura.
7. Recursos de hardware que se utilizan. Contribución potencial al cumplimiento de los objetivos de la asignatura.
8. Formas organizativas de la docencia que se utilizan. Asignación de tiempo para cada una de ellas.
9. Sistema de evaluación que se utiliza.
10. Correspondencia entre el método de enseñanza que se aplica y los recursos que se utilizan. Vigencia del método.

El tele taller de la asignatura ED I fue coordinado por la CUJE y se desarrolló en el período comprendido entre el 3 de junio y el 17/10 del año 2002. Durante el desarrollo del mismo se valoraron por todos los centros participantes los programas analíticos vigentes para el curso 2001-2002 a través de un instrumento de medición desarrollado por la UCLV que permitió la búsqueda de regularidades a partir de la evaluación de 10 criterios.

El tele taller de la asignatura ED II fue coordinado por la UCLV y se desarrolló en el período comprendido entre noviembre y diciembre de 2002.

Anexo 2. ENCUESTA DE ELECTRONICA DIGITAL

Temática: Estrategias de diseño digital.

Curso: 2003-2004.

Grupo: 3ro de Telecomunicaciones y Electrónica.

La presente encuesta tiene como finalidad que Ud. evalúe cómo la preparación de la asignatura ha permitido el desarrollo de sus habilidades en el diseño de sistemas digitales.

1. Considera Ud. que los temas estudiados en la asignatura están actualizados?.

   No En cierta Medida Sí Desconozco
2. Cómo evalúa Ud. la planificación de las actividades (Conferencias, C.Práct., Labor., Sem, Evaluaciones)?.

   Adecuada No Adecuada Desconozco

   Sugerencias:
3. Considera Ud. que el ordenamiento de los temas es el correcto para desarrollar sus habilidades en el diseño?.

   Sí No En cierta medida Desconozco

   Sugerencias:
4. Considera que algún tema debe ser eliminado?. Cuál?
5. Considera necesario incluir algún tema que mejore sus habilidades de diseño?. Cuál?
6. Considera que el sitio Web digital facilita el desarrollo de dichas habilidades?.

   Sí No En cierta medida Desconozco
7. Qué otras facilidades le sugeriría al mismo?.
8. Facilitan las actividades docentes la comprensión de las estrategias de diseño?.

   Sí No En cierta medida Desconozco

Sugerencias:

9. Ha comprendido Ud. la estrategia de diseño top-down (ascendente-descendente)?.

   Sí No En cierta medida Desconozco

10. Considera de utilidad el desarrollo del proyecto final para la integración de sus habilidades de diseño?.

    Sí No En cierta medida Desconozco

Sugerencias:

11. Le ha resultado útil la herramienta OrCAD en el desarrollo de la estrategia de diseño top-down?.

    Sí No En cierta medida Desconozco

12. Qué otras herramientas y/o facilidades para el diseño necesitaría? (incluso aunque no existan, alguna que Ud. considere le facilite el trabajo)
13. Ha llegado Ud. a familiarizarse con las estructuras fundamentales del VHDL para describir sistemas combinacionales y secuenciales?.

    Sí No En cierta medida No me interesan

14. Le facilita el VHDL al desarrollo de la estrategia para un diseño confiable?

    Sí No En cierta medida No me es necesario Me molesta

15. Facilita el trabajo en grupo, para el proyecto, el desarrollo de las habilidades de diseño?

    Sí No En cierta medida No me es necesario Me molesta

16. Considera Ud. que la forma de evaluación de dicho proyecto es la adecuada?.

    Sí No En cierta medida Propongo otra:

17. Considera Ud. que con los medios y métodos de que dispone (Web, ejemplos, simuladores, materiales de consulta) puede enfrentar diseños futuros de forma independiente?.

    Sí Sí, pero formado parte de un equipo No En cierta medida

18. De no ser positiva la respuesta anterior, qué considera le falta para ello?