Contenido

Editorial

Actualmente en diversas instituciones educativas se está promoviendo la implantación de estudios Doctorales como continuación inmediata a la Licenciatura, en contraposición a mantener el nivel de Maestría como punto intermedio. La comunidad académica del CINTEC considera que esta medida es inadecuada en el campo de la Computación, debido al alto grado de madurez necesaria en los profesionales de esta área, que no es posible alcanzar únicamente con el nivel de Licenciatura en forma satisfactoria.

Como es sabido, el CINTEC ofrece la Maestría en Ingeniería de Cómputo, con especialidad en Sistemas Digitales, teniendo el compromiso permanente de elevar la calidad de este Programa de Posgrado y de la Investigación asociada al mismo. Aquí es forzoso formular una pregunta : ¿Cuáles son las estrategias planteadas en el CINTEC para lograr este objetivo?

Por una parte, el Centro promueve la actualización docente de su Colegio de Profesores, mediante el apoyo a la matrícula en estudios de nivel Doctoral. Complementario a estos estudios se promueven también diversas participaciones en Congresos, Conferencias y Simposia en el área de su interés, en el carácter tanto de asistentes como de ponentes o expositores.

Así mismo, el CINTEC ha planteado la conveniencia de actualizar y mejorar la infraestructura física y tecnológica con que cuenta en la actualidad, de tal forma de mantenerse a la vanguardia tecnológica. En este punto, se cuenta con el apoyo del CONACYT, a través del denominado "Programa de Apoyo a la Ciencia en México" (PACIME). Dentro de dicho programa, el Centro desarrolla el proyecto : "Infraestructura de Laboratorios de Docencia e Investigación", con un presupuesto de N$1,448,278.00. Esta cantidad no tiene precedente en la historia de los apoyos al desarrollo de Tecnología en México, tomando en cuenta que se trata de un solo Centro de Investigación el que recibe casi el 50% del apoyo total otorgado al Instituto Politécnico Nacional por parte de este programa.

Cabe hacer notar que este apoyo no fue otorgado por existir necesidades apremiantes, sino como soporte para ampliar el campo de actividades del Centro mediante una reconversión de su infraestructura. Esto es resultado del reconocimiento que se ha merecido el CINTEC por la labor Docente y de Investigación que desarrolla desde su origen.

La importancia estratégica que ha demostrado poseer la calidad de la Investigación realizada, queda avalada plenamente por dicho apoyo. Con la infraestructura propuesta, y la constante superación de su Personal Académico, el CINTEC aspira a convertirse en un Centro de Investigación líder a nivel nacional, donde el Desarrollo Tecnológico se dé a la par de la Investigación Científica.

Introducción a la computación paralela

M. en C. Antulio Morgado Valle
Jefe del Departamento de Electrónica del CINTEC-IPN.

En este artículo se desea presentar al lector una revisión de conceptos básicos usados en el área de computadores en paralelo, así como algunos de los problemas en este campo que actualmente son de interés para la comunidad científica. Para ello, al final del artículo se presenta una revisión bibliográfica al respecto.

Introducción

La investigación en computadoras de alto rendimiento incluye varios campos que van desde la arquitectura de la computadora por si misma hasta los algoritmos y aplicaciones. **[1.2.3]**. En el campo del diseño de computadoras, en esta década se ha dado más énfasis al desarrollo de arquitecturas de computadoras en paralelo, debido a que éstas tienen el enorme potencial de incrementar sustancialmente la capacidad de cómputo. Esto se debe al hecho de que un número considerable de procesadores trabajando en paralelo reducen en gran medida el tiempo de cómputo, siempre y cuando la computadora tenga la arquitectura adecuada y que los "algoritmos" de los programas sean susceptibles a una división en procesos paralelos.

Actualmente no existen acuerdos en cuanto a la topología ideal para las computadoras con arquitectura en paralelo y es posible que no se llegue a un consenso a corto plazo, ya que las aplicaciones y algoritmos son muy diferentes y diversos. Además de la limitante tecnológica que impide realizar físicamente muchas de las topologías propuestas.

Sin embargo, el diseño de procesadores de alto desempeño suple las limitantes anteriores, y en un futuro próximo veremos cuajados los esfuerzos en esta área que se traducirán en computadoras paralelas cuyos costos sean accesibles a la mayoría de los usuarios que requieran hacer uso de la supercomputación.

Antecedentes

M. J. Flynn [4] ha clasificado las arquitecturas para computadoras paralelas en 4 tipos, dependiendo del flujo de datos y/o del flujo de instrucciones:

* En la arquitectura SISD una sola instrucción es procesada o actúa sobre cada dato a la vez; en esta arquitectura están basadas todas las máquinas monoprocesador y computadoras personales que existen en la actualidad (Modelo de Von Newman).

Las instrucciones se ejecutan secuencialmente pero pueden estar solapadas ("pipelined"). Un computador con esta arquitectura contiene varias unidades funcionales, bajo la supervisión de una unidad de control principal.

* En la arquitectura SIMD la misma instrucción es ejecutada por todos los procesadores a la vez; así cada procesador ejecuta la misma operación sobre diferentes datos (Máquinas de procesamiento matricial).

* En la arquitectura MISD existen N procesadores, cada uno ejecutando diferentes instrucciones sobre el mismo flujo de datos y sus derivados. Los resultados de un procesador pasan a ser la entrada

Nombre	Significado
SISD	Single Instruction Stream Single Data Stream.
SIMD	Un Solo flujo de Instrucciones Un Solo Flujo de Datos. Single Instruction Stream Multiple Data Stream.
MISD	Un Solo flujo de Instrucciones Múltiple Flujo de Datos. Multiple Instruction Stream Single Data Stream.
MIMD	Múltiple flujo de Instrucciones Un Solo Flujo de Datos. Multiple Instruction Stream Multiple Data Stream.

del siguiente procesador en el macrocauce. Este es un modelo teórico, ya que no existe ningún desarrollo práctico de esta arquitectura.

* La arquitectura MIMD permite ejecutar diferentes instrucciones sobre diferentes conjuntos de datos a la vez. Esta arquitectura es la más flexible de todas, ya que permite el procesamiento de datos que pueden o no estar relacionados entre sí. Cabe aclarar que en esta arquitectura se han basado la mayoría de las computadoras paralelas comerciales que existen en la actualidad.

Una máquina multiprocesadora es aquella en donde una serie de procesadores se interconectan a través de una red con un bloque de memoria global, dividida en pequeñas secciones, lo cual hace aparecer M bloques de memoria contra N procesadores. La comunicación en estos procesadores se efectúa a través de "pase de variables" en "memoria compartida".

Dependiendo de la distribución física de la memoria, los sistemas multiprocesadores se dividen en tres tipos:

* UMA "Uniform Memory Access" (Acceso Uniforme a Memoria). Este nombre se debe a que los retardos al accesar a cualquier localidad de memoria son similares. Ver **figura 1**.

* NUMA "Non Uniform Memory Access" (Acceso No Uniforme a Memoria). En este modelo el bloque de memoria se coloca en forma local al procesador por lo que cuando un procesador accesa a una memoria remota (que se encuentra asignada a otro procesador) el tiempo de acceso es mayor al tiempo de acceso a su memoria local. Ver **figura 2**.

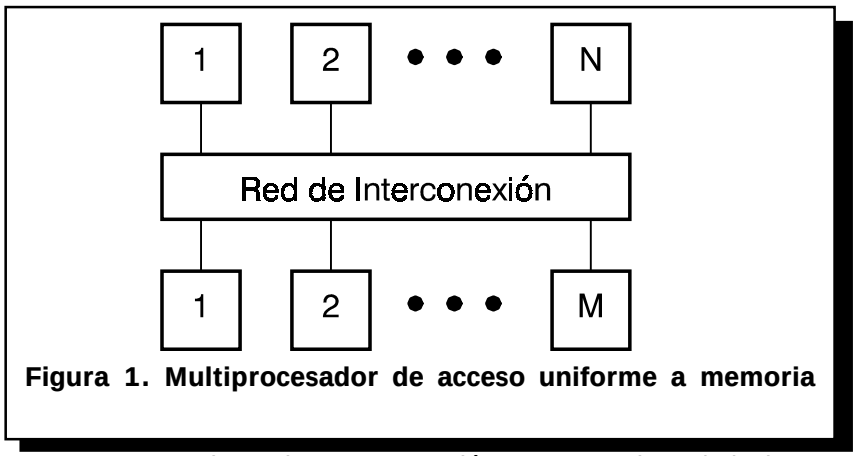


Figura 1. Multiprocesador de acceso uniforme a memoria

* COMA "Cache Only Memory Access" (Acceso Solo a Memoria Caché). En este modelo la memoria compartida se sustituye por memoria caché, comportándose como el modelo NUMA con una mayor velocidad de acceso. Se emplean directorios distribuidos para localizar direcciones en cachés remotas. Ver **figura 3**.

Un sistema multicomputador consiste de un número de nodos de proceso interconectados a través de una red, donde cada nodo de proceso contiene un procesador, un módulo de memoria local y una interface de interconexión a la red.

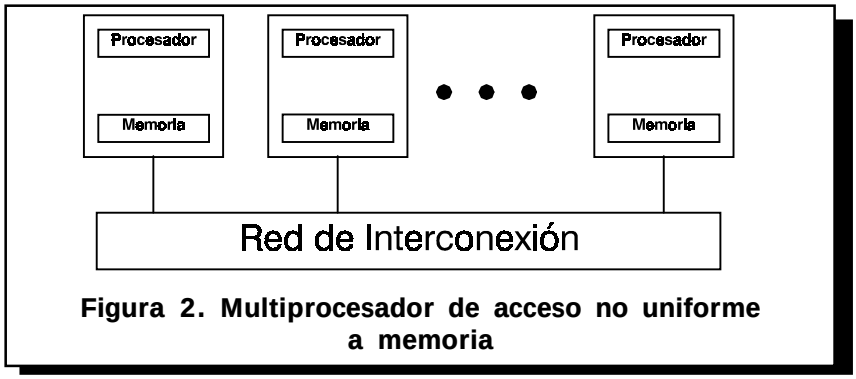


Figura 2. Multiprocesador de acceso no uniforme a memoria

Todas las comunicaciones entre procesadores se realizan por medio de "envío de mensajes".

El rendimiento de un sistema en paralelo depende en alto grado de la velocidad de comunicación que permita la red de intercomunicación. Por esto la red de interconexión, en su conjunto, constituye uno de lo factores más importantes para construir un sistema en paralelo.

Normalmente una red de interconexiones se evalúa tomando en cuenta varios elementos. Entre otros se mencionan la forma de interconexión de los nodos, la configuración misma de éstos y el algoritmo de control de rutas.

La arquitectura de sistemas multiprocesadores cuenta básicamente con tres elementos principales que son: el procesador, la memoria y una red de interconexión, quedando clasificados los sistemas UMA, NUMA y COMA dentro de estos sistemas.

La arquitectura de sistemas multicomputadores adiciona un elemento más, el ruteador, **figura 4**.

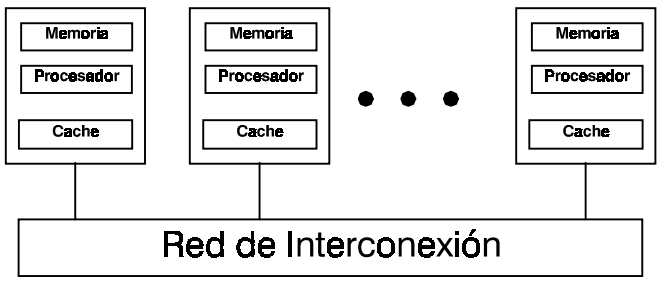


Figura 3. Multiprocesador de acceso solo a memoria Caché.

Dentro de este esquema se tienen dos bloques funcionales: La unidad de proceso ("Process Unit", PU) y la red de interconexión. La PU está constituida por un procesador, memoria local y la unidad de control de rutas ("Control Data Routing", CDR) que incorpora el ruteador y la interface a la red de interconexión. El procesador sigue siendo la unidad clásica compuesta por una unidad aritmética y lógica ("Arithmetic Logic Unit", ALU) y registros. El bloque de memoria consta de un manejador de memoria ("Memory Managment Unit", MMU) y unidades de memoria. La unidad de control de rutas CDR consta de un área de memoria ("buffer"), una unidad de control de flujo de información y la interface a la red. También pueden incluirse otras funciones (Dispositivos Periféricos, E/S, etc). Las E/S pueden adicionarse como un subconjunto en una o todas las unidades de proceso.

En esta arquitectura cada procesador tiene asignada una parte de la memoria global. Un procesador no tiene acceso directo a la memoria local de otro procesador o nodo; debido a esto, toda la coordinación y sincronización entre múltiples procesadores en un sistema multiprocesador se efectúa por medio de un protocolo de "pase de mensajes".

El CDR **[15]**, presente solamente en sistemas paralelos, tiene la función de controlar los mensajes que se transmiten entre los procesadores, desde su origen hasta su destino. Un mensaje puede pasar por varios nodos si es que no existe una conexión directa entre el procesador que originó el mensaje y el procesador destino. Dependiendo de la topología con que se conecten los procesadores, el C.D.R buscará la o las rutas más idóneas para enviar el mensaje que se esté transmitiendo.

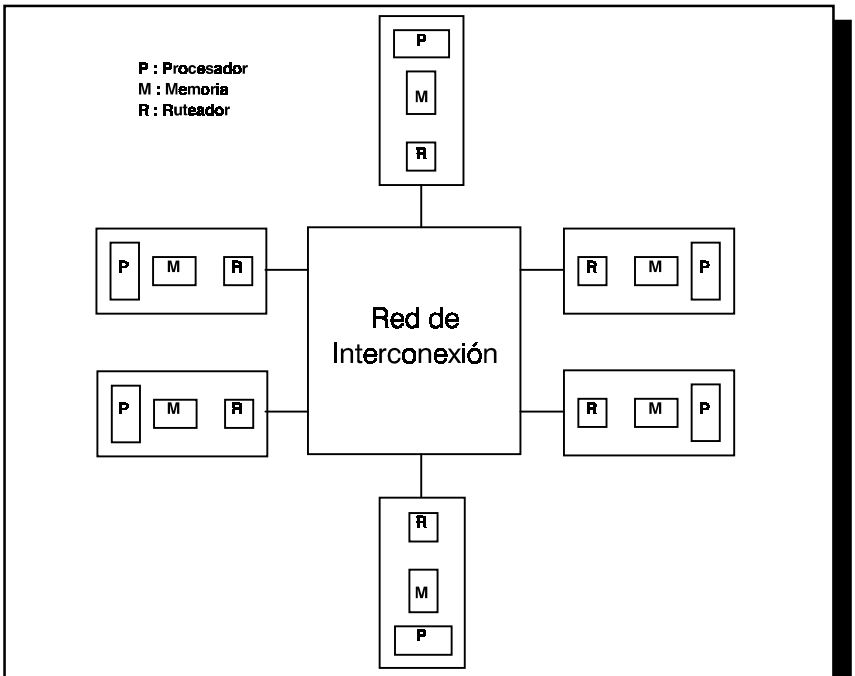


Figura 4. Multicomputador con "envío de mensajes".

Normalmente existen dos tipos de CDR con respecto a la topología:

El primer tipo se relaciona con sistemas de topología estática en donde, para cada sistema el CDR ejecuta un solo algoritmo conforme al tipo de topología definida. Ejemplo de esto son los sistemas "TORUS" **[5]**, los sistemas "IPSC/2" **[6]** y el cubo "COSMIC" **[7]**.

El segundo bloque de CDR se relaciona con sistemas con topología dinámica. Para cada uno de estos sistemas el CDR es capaz de ejecutar varios algoritmos o un algoritmo que se adapta a diferentes topologías. Ejemplo de esto son el "INTEL/CMUS'S iWARP" **[8]** o el "INMOS" transputer **[9]**.

Para ambos tipos de CDR existen a su vez dos tipos de protocolos de ejecución. El primero está basado en la tecnología "SOURCE

ROUTING", en donde la decisión de la ruta a tomar pertenece directamente al nodo. En este caso el paquete de información a procesar tiene un encabezado ("Header") que almacena toda la información necesaria y en base a ésta, junto con la topología del sistema y con la condición del nodo procesador, se toma la decisión de la ruta a seguir para llegar al nodo destino.

En el segundo tipo de protocolo la información de los nodos se almacena en una "TABLA" que reside en la memoria global del sistema. Las condiciones particulares de cada nodo en el momento de la ejecución son tomadas en cuenta junto con las condiciones generales de la "TABLA", para determinar el nodo destino.

Los dos tipos de CDR mencionados son comerciales. Ambos tienen ventajas y desventajas, ya sea a nivel protocolo o topológico.

Las desventajas más importantes entre ellos son: los "ruteadores" utilizados en topologías fijas son de uso específico, poco flexibles y se vuelve costoso y de difícil manejo a nivel fabricante, ya que es necesario desarrollar un CDR para cada topología. Los "ruteadores" utilizados en topologías dinámicas son de uso general y difíciles de implementar lo cual eleva más sus costos, presentando poca flexibilidad en ciertas topologías. Aunque en realidad con un solo tipo de CDR es suficiente, siempre y cuando sea lo suficientemente flexible. Entonces el problema se transfiere al control central y a la memoria principal para su manejo a nivel de protocolo.

Como podemos notar existen serios problemas para ambos tipos de "ruteadores". A nivel protocolario, en el "SOURCE ROUTING" es necesario aumentar la lógica del CDR para permitir que cada paquete de información lleve consigo todo lo necesario. En el segundo, el problema principal reside en el tamaño de la "TABLA", siendo esta proporcional a la cantidad de nodos en el sistema, pudiendo esto ser también una limitante en la cantidad de nodos posibles.

Como se observa, en la actualidad el problema de implementación de cualquier topología paralela radica principalmente en la intercomunicación entre nodos procesadores, y es aquí en donde se esperan aportaciones que ayuden a resolver dichas limitantes.

Una de ellas sería el implementar algoritmos eficientes y compactos para establecer los protocolos de comunicación entre nodos. Otro sería implementar estos algoritmos como parte del CDR y por último encontrar una red de conmutación de alto desempeño que se adapte bien a la mayoría de las topologías propuestas, todo esto bajo las limitantes que impone el uso de tecnología planar en VLSI para la realización de dispositivos semiconductores.

Bibiliografía

[1] G.S. Almasi y A.Gottlieb. *"Highly parallel computer"*. The Benjamin Cummins. 1989.

[2] R.W. Hockney y C.R. Yesshope. *"Parallel computer"*. Adam Hilger LTD. 1983.

[3] Al Kernek, *"Massively parallel systems - The revolution has begun"*. Super Computing Magazine, pp. 27-28, Otoño 1988.

[4] M. J. Flynn. *"Some computer organizations and their effectivness"*. IEEE Transactions on Computers, Vol 21, Nº 9, pp. 948-960. September 1972,

[5] S. Park S. Vassiliadis y J. G. Delgado-Frias. *"Flexible oblivius router architecture"*. IBM Technical Report Trol. C749, IBM Endicott, Nueva York 13760, Sept. 1993.

[6] W. J. Dally and C. L. Seitz. *"The TORUS routing chip"*. Distributed Computing, 1: 187-196, 1986.

[7] S. Borkar. *"iWRAP: An integrated solution to High-speed parallel computing"*. In proceding of Supercomputing Conference, p.330-338 1988.

[8] C.L. Seitz. *"Cosmic cube"*. Communication of ACM, 28(1): 22-33 1988.

[9] N. Santoro y R. Khitib. *"Routing without routing tables"*. Technical report SCS-TR-6, Sschool of Computer Science.

[10] I. Van Leeuwen y R.B. Tan. *"Routing with compact routing tables"*. Technical report RUU-CS-83-16, Departament of Computer Science, University of Utrecht, 1983.

[11] R. Payne and J. G. Delgado-Frias. *"MPU: A Matching-Processing Unit".* IEEE Int. Conf. on Computing Design, Cambridge, Mass. Oct 1991.

[12] J. Park, S. Vassiliadis and J. G. Delgado-Frias. *"Packet flow controller whit self*

compacting buffer". IBM 1993.

[13] Harold D. Johnson, José Delgado-Frias, S. Vassiliadis and Douglas M. Green. *"A Petri net technique for accesing performance of loosely couple multiprocesor machine architecture"*. Int. Journal on microcomputers applications, Vol 11, Nº 1 pp 6-40, 1992.

[14] José Delgado-Frias, S. Vassiliadis, G Pecha-Nek, Harold D. Johnson and Douglas M. Green. *"A processing unit for flexible multiprocessor machine organizations"*. Conf. on Computer Design, October 1992.

[15] K. Hwang. *"Advanced Computer architecture: Paralelism, Scalability, Programability"*. Mc Graw-Hill, 1981.

Arreglos de compuertas programables en campo, FPGA's. (compendio)

M. en C. Juan Carlos Gonzáles Robles.
Jefe del Departamento de Producción y Adecuación de Tecnologías del CINTEC-IPN.
Ing. Eduardo Vega Alvarado.
Jefe del Departamento de Laboratorios Ligeros del CINTEC-IPN.

El presente artículo pretende continuar con una serie de publicaciones relacionadas con el tema del diseño en base a la lógica programable. En esta ocasión se describe a los Arreglos de Compuertas Programables en Campo ("Field Programmable Gate Array", FPGA).

Introducción

Las arquitecturas FPGA's surgen como respuesta a la necesidad del diseñador de obtener mayor funcionalidad en un solo dispositivo programable. A pesar de ser dispositivos programables, los FPGAs se distinguen de los Dispositivos de Lógica Programable ("Programmable Logic Devices",PLD), en el hecho de que requieren estrategias de implementación completamente distintas, así como por el uso de arquitecturas que no se basan en Arreglos Lógicos Programables ("Programmable Logic Array", PLA).

Características de los FPGA's

Esencialmente, los FPGA's están compuestos por elementos con recursos no comprometidos que pueden ser seleccionados, configurados e interconectados; el PLD no requiere del proceso de interconexión dado que sus elementos internos (registros, arreglos lógicos y macroceldas de E/S) son fijos en relación de unos con otros.

Los FPGA's actualmente en uso presentan algunas características fundamentales, entre las que destacan:

- Todos estos dispositivos están compuestos de un cierto número de módulos lógicos relativamente independientes entre sí, que pueden interconectarse para formar un circuito mayor; estos módulos pueden ser grandes bloques configurables o pequeños elementos de función fija formados por algunas compuertas.

- El tamaño óptimo de los módulos lógicos y los requerimientos de interconexión son completamente dependientes del tipo de aplicación que se desea implementar en el dispositivo. Por ejemplo, una aplicación compuesta de estructuras regulares o semi-independientes puede implementarse en forma más eficiente en un FPGA con módulos lógicos grandes, mientras que si se tiene un diseño en el que predominen funciones lógicas aleatorias interrelacionadas se puede emplear un dispositivo con menos módulos y más recursos de interconexión.

- Los módulos en un FPGA se interconectan por medio de canales configurables, mediante un proceso conocido como enrutamiento. El enrutamiento consiste básicamente en determinar, ya sea en forma manual o a través de herramientas de cómputo, una estrategia de interconexión eficiente.

- El rango en tamaño de los FPGA's abarca desde 1,200 hasta 20,000 compuertas equivalentes, mientras que en los PLD's es desde algunos cientos hasta 2,000 compuertas. Dado que los FPGA's se están incrementando constantemente en cuanto a densidad, se puede esperar que sus aplicaciones típicas serán en orden de magnitud más compleja que para un simple PLD.

Familias de FPGAs.

Existen diferentes tipos de FPGA's; cada una de estas familias presenta una arquitectura única, y también difieren en sus tecnologías básicas de fabricación. La aparición de mayores densidades de componentes en los FPGA's a motivado al desarrollo de nuevas tecnologías de programación para estos dispositivos, mientras que en todos

los PLD's se han adoptado los métodos de programación de PROM's y EPROM's (fusibles bipolares programables una sola vez o celdas CMOS borrables). A continuación se describen algunas de las familias más representativas de estos dispositivos:

Arreglo de Celdas Lógicas (LCA's).

La primer familia de FPGA's fue anunciada en 1984 por Xilinx Corporation. Los dispositivos, llamados Arreglos de Celdas Lógicas ("Logic Cell Array", LCA) , se manufacturan usando un proceso de producción CMOS de alta velocidad. Como todos los FPGA's, los LCA's no se basan en un simple arreglo lógico programable, sino que se componen de pequeños módulos, los cuales son elementos lógicos programables conocidos como Bloques Lógicos Configurables ("Configurable Logic Block", CLB). La **figura 1** muestra el diagrama a bloques de la arquitectura LCA, ejemplificando al dispositivo Xilinx XC2064, que está compuesto de 64 CLB's de propósito general, cada uno de los cuales es capaz de implementar ya sea dos funciones lógicas de tres entradas, o una función simple de cuatro o cinco entradas. Estos CLB's son similares a una pequeña PROM conteniendo además un flip-flop y lógica de retroalimentación. Cada CLB contiene un elemento de memoria en la forma de un flip-flop configurable que puede utilizarse como un latch transparente o un flip-flop tipo D disparado por flanco.

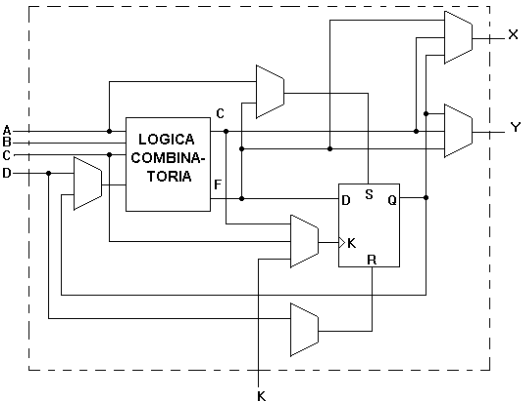


Figura 2.- Bloque Lógico Configurable XC2064.

Las celdas lógicas de propósito especial, llamadas bloques de E/S ("I/O Block", IOB), son arreglos colocados alrededor del dispositivo, cada uno de los cuales puede ser configurado para usarse como una entrada, una salida en tercer estado, o una terminal de E/S bidireccional. Los IOB's pueden configurarse además con flip-flops y ser accesados por los CLB's internos para funciones distintas a E/S. Estas celdas configurables se arreglan tal como se muestra en la **figura 1**, y pueden ser interconectadas a través del uso de canales de enrutamiento programable. Como cada CLB del LCA XC2064 es capaz de implementar funciones de complejidad limitada, los recursos de interconexión requeridos son consecuentemente grandes (como se muestra en la **figura 2**). La ubicación de las funciones lógicas en los CLB's y la determinación del enrutamiento de interconexión son los problemas principales de diseño para los usuarios de LCA's.

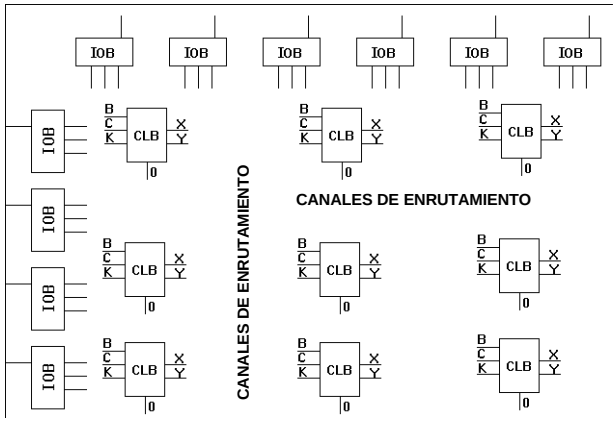


Figura 1.- FPGA XC2064 de Xilinx.

Los métodos de programación usados en el LCA también difieren a los del tradicional PLD; su esquema de programación se basa en la tecnología RAM estática. El dispositivo se carga con un patrón de configuración al instalarse en el circuito. Mediante esto, las configuraciones nuevas se pueden programar fácilmente en el dispositivo mientras está en operación en el sistema. En la mayoría de los casos, el patrón de programación del LCA se almacena en una configuración PROM instalada en la tarjeta simultáneamente con el dispositivo LCA. La configuración PROM es leída automáticamente por el LCA al encender el sistema.

Los LCA presentan la más alta velocidad en sus flip-flops (hasta 100 MHz en el cambio de estado), pero generalmente no son adecuados para aplicaciones de alta velocidad debido al retraso inherente en la matriz de interconexión, dado que incluyen elementos no metálicos, resultando en señales retardadas. El retardo total del circuito es difícil de predecir, siendo dependiente del enrutamiento de las señales de interconexión.

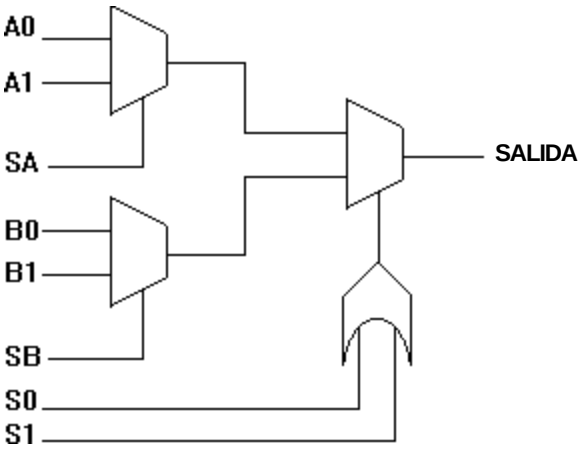


Figura 4.- Módulo Lógico de Actel.

Dispositivos ACT

En 1988, la compañía Actel liberó una nueva familia de dispositivos basados en tecnología CMOS, cuya arquitectura se sitúa en el rango de 3,000 a 6,000 compuertas equivalentes. Estos FPGA's, denominados como ACT, están compuestos por renglones de bloques lógicos no comprometidos separados por canales de enrutamiento, tal como se muestra en la **figura 3**.

Los módulos lógicos consisten de tres multiplexores de dos a uno y una compuerta OR, formando un arreglo como el que se muestra en la **figura 4**. Debido a su simplicidad, estos módulos son más generales que los CLB's que conforman a los LCA's, teniendo cada uno ocho entradas y ningún registro interconstruido. Ya que los registros se construyen a partir de los módulos lógicos básicos (uno o dos módulos pueden teóricamente implementar un flip-flop de cualquier tipo), pueden aplicarse únicamente donde sean necesarios, por lo que no hay flip-flops sobrantes o desperdiciados.

A diferencia de los módulos lógicos LCA, los cuales son internamente configurables, los elementos de Actel se configuran conectando los módulos lógicos de entrada a los canales de interconexión y fijando los módulos restantes de entrada a niveles altos o bajos. Esta arquitectura es muy similar a los arreglos lógicos programados por máscara, y permite al dispositivo el implementar una gran variedad de circuitos.

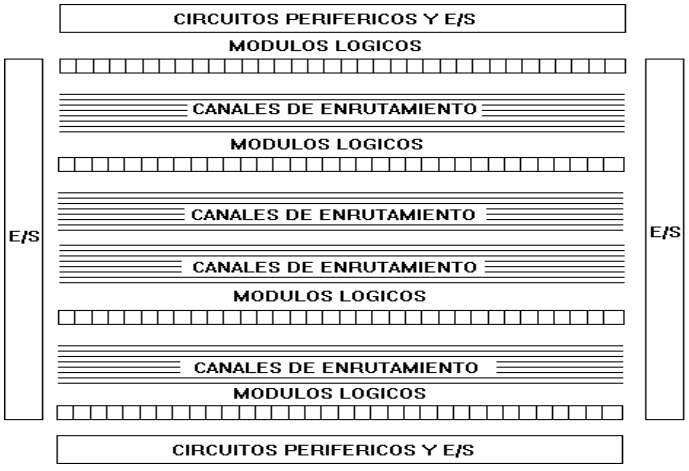


Figura 3.- FPGA ACT de Actel.

Sin embargo, el uso de módulos lógicos menos complejos tiene un precio; dado que los módulos de Actel son de un tamaño relativamente pequeño y son configurados exclusivamente a través de sus interconexiones con otros módulos, forzosamente deben dedicar una gran parte de su área para interconexiones internas. Para resolver este problema, se utiliza un solo elemento compacto de programación, el cual se denomina Elemento Programable de Circuito de Baja Impedancia ("Programmable Low Impedance Circuit Element", PLICE). A diferencia de los fusibles establecidos en las PROM's y PLD's, éste no está conectado en su estado original, sino hasta el momento en que es programado, por lo que se conoce también como antifusible.

La tecnología antifusible presenta una alternativa importante, dado que su elemento de programación es extremadamente pequeño siendo aproximadamente 50 veces menor que los elementos RAM estáticos establecidos en los LCA's y 10 veces en relación con una celda CMOS EPROM. Como desventaja se tiene el hecho de que no son borrables y por lo tanto no pueden probarse antes de ser programados.

La naturaleza no reprogramable de este dispositivo, aunado con su relativo alto precio, se ha constituido en una barrera muy importante para la comercialización del mismo. Para algunas aplicaciones, la velocidad es el primer factor que afecta la decisión y, en esta área, los dispositivos de Actel tienen la delantera; ya que las trayectorias de interconexión son todas metálicas y los retardos inherentes son menores que las trayectorias de las señales de los LCA's.

Arreglo Reconfigurable Eléctricamente

Un reciente ingreso al campo de los FPGA's es el circuito Arreglo Reconfigurable Eléctricamente ("Electrically Reconfigurable Array", ERA), desarrollado por Plessey Semiconductors; este dispositivo es similar en algunos aspectos al Arreglo de Celdas Lógicas. El primer dispositivo ERA contenía 10,000 compuertas equivalentes, y actualmente se hallan disponibles circuitos de hasta 100,000 compuertas.

Como los LCA's, el ERA está basado en tecnología RAM estática, pudiendo ser reprogramados al momento de operación. Esto significa que un segmento de lógica dentro del ERA puede, teóricamente, controlar la reconfiguración dinámica de un segmento distinto del dispositivo; así mismo, éste se encuentra formado por un gran número de compuertas NAND de dos entradas, en lugar de más bloques lógicos complejos. Si la aplicación requiere más funciones complejas, tales como flip-flops, estas funciones deben ser construidas fuera de la estructura básica NAND.

El ERA tiene, además de su arreglo central de compuertas NAND, una serie de 84 celdas configurables para fines de E/S. Cada uno de los módulos de arreglos de compuertas puede usarse ya sea para propósitos lógicos o para enrutamiento de interconexión. Para proporcionar mayor eficiencia en el enrutamiento de las señales en el arreglo, el dispositivo tiene un bus configurable de 10 líneas rodeando al arreglo principal; este puede usarse para enrutar las señales de una sección a otra del arreglo, o hacia y desde las celdas de configuración de E/S o los bloques fijos de E/S.

La simplicidad del dispositivo ERA permite un uso más eficiente del dispositivo que el disponible en el LCA, pero esta ventaja en eficiencia puede ser una pérdida debido a la mayor necesidad de interconectar circuitos en el dispositivo. Las interconexiones en el ERA se enrutan a través de los mismos módulos lógicos, por lo que para mayores requerimientos de interconexión se dispone de un menor número de compuertas NAND.

Transición de los diseños en PLD's a FPGA's

Las altas densidades de los dispositivos FPGA's, aunado con su programación directa por el usuario, hace de ellos una alternativa importante para aplicaciones de ingeniería basadas en diseños de PLD's; sin embargo, existen algunas diferencias entre PLD's y FPGA's que deben tomarse en cuenta.

Lo primero es entender las limitaciones de cada arquitectura y cómo dichas limitaciones afectan a los diseños que son originalmente pensados para una arquitectura. El mayor limitante cuando se diseña con PLD's es el número relativamente bajo de salidas y flip-flops disponibles. Se puede pensar en los PLD's generalmente como dispositivos carentes o con pocos registros, y con una cantidad considerable de entradas lógicas (aunque sí el diseño requiere lógica multinivel, los PLD's sencillos son bastante limitados incluso en este aspecto). Un FPGA, en cambio, puede tener un mayor número de registros, mientras que la lógica de entrada para estos registros es limitada; esto es particularmente cierto en los dispositivos LCA. Mientras que una salida de un PLD puede tener la capacidad de implementar una ecuación con 20 o más entradas, esta misma ecuación representada en un LCA puede requerir hasta 7 de los CLB's del dispositivo, lo que en pequeños LCA's puede representar más del 10% de los CLB's disponibles.

Esto significa que un diseño que tiene lógica de entrada compleja, un número considerable de entradas y relativamente pocos registros puede implementarse perfectamente con un número reducido de PLD's, y en forma bastante ineficiente en un dispositivo LCA. Por otra parte, los PLD's y los FPGA's difieren enormemente en cuanto a velocidad, por lo que si se requiere alta velocidad de operación quizás sea imposible utilizar un FPGA para sustituir un diseño implementado por PLD's.

Bibliografía

[1] David Pellerin / Michael Holley. *"Practical design using Programmable Logic"*. Prentice Hall, 1991.

[2] Edwards J. Mc. Cluskey. *"Logic design principles whit emphasis on testable semicustom circuits"*. Prentice Hall, 1986.

[3] Di Giacomo. *"Designing whit high performance ASIC's"*. Prentice Hall, 1991.

Sistema de control remoto de parámetros

M. en C. Miguel A. Partida Tapia.
Subdirector Académico y de Investigación del CINTEC - IPN.
Ing. Ma. Elena Aguilar Jáuregui.
Ing. Enrique Jarquín León.
Alumnos de Maestría del CINTEC.

El presente artículo muestra el diseño de un sistema digital que controla la excitación de los actuadores existentes en el laboratorio del sistema de transporte colectivo "Metro", de tal manera que las señales de vibración (aceleración, velocidad, desplazamiento, etc.) que se registran en pruebas de campo sobre diferentes tipos de vehículos, se puedan reproducir en el laboratorio a fin de simularlas, y así poder aplicar determinado número de ciclos de vibración al vehículo bajo prueba. Adicionalmente se tiene la capacidad de editar y sumar señales para que posteriormente sean enviadas a los respectivos excitadores.

Descripción del sistema

En la **figura 1**, se muestra un diagrama de bloques del sistema integral requerido para efectuar las pruebas de simulación a vehículos o estructuras diversas.

En la consola de control de los actuadores se cuenta con un generador de señales las cuales, una vez programadas, excitan las electroválvulas de los actuadores, permitiendo así el paso del fluido proveniente del sistema hidráulico y lográndose un desplazamiento en dichos actuadores proporcional a la amplitud de la señal. En la misma consola, se cuenta con un circuito analógico que forma una malla de realimentación que sensa el desplazamiento de los actuadores y corrige contra la señal programada, teniéndose así un sistema de lazo cerrado que proporciona a la electroválvula la señal requerida.

Por otra parte, cuando la señal de excitación es externa, (grabadora) producto de un registro de prueba de campo, ésta es "inyectada" en la consola de control. En este caso no existe realimentación para verificar el punto de prueba de campo (acelerómetro del 1 al 8); entonces se genera la misma vibración que la obtenida en prueba de campo. Por tal motivo es necesario agregar un control digital que forme un lazo cerrado con el sistema, de tal manera que los acelerómetros 1 al 8 se comporten de manera similar a las señales obtenidas en campo.

Características de las señales

Las señales provenientes de los transductores pueden ser de aceleración, velocidad, desplazamiento, esfuerzo y temperatura. Para propósitos de control, las señales pueden ser adquiridas por el sistema digital para su análisis, contándose con ocho canales de salida.

Configuración del sistema

El sistema cuenta con la siguiente configuración: una tarjeta de reproducción de ocho canales de salida y dos tarjetas de adquisición con 8 canales de entrada. Estas tres tarjetas tienen una interface a un "*bus PC*", lo cual permite usar una computadora PC/XT o PC/AT compatible. Las tarjetas son autónomas en su operación ya que cuentan con rutinas en "firme" y memoria suficiente para adquirir y reproducir las señales.

Descripción del hardware

Tarjeta de reproducción con 8 canales de salida

Esta tarjeta cuenta con 4 ConvertidoresDigital Analógico ("Digital Analogic Converter",DAC) dobles del tipo DAC 1248 de "Precision Monolithics Inc.", a 12 bits de resolución, con un rango dinámico de +/-5V y 100 ns de acceso de escritura, todos estos controlados por un microprocesador 80C188EB @ 16 MHz de Intel, 128 Kb de memoria de solo lectura para el "firmware" y 872KB de memoria

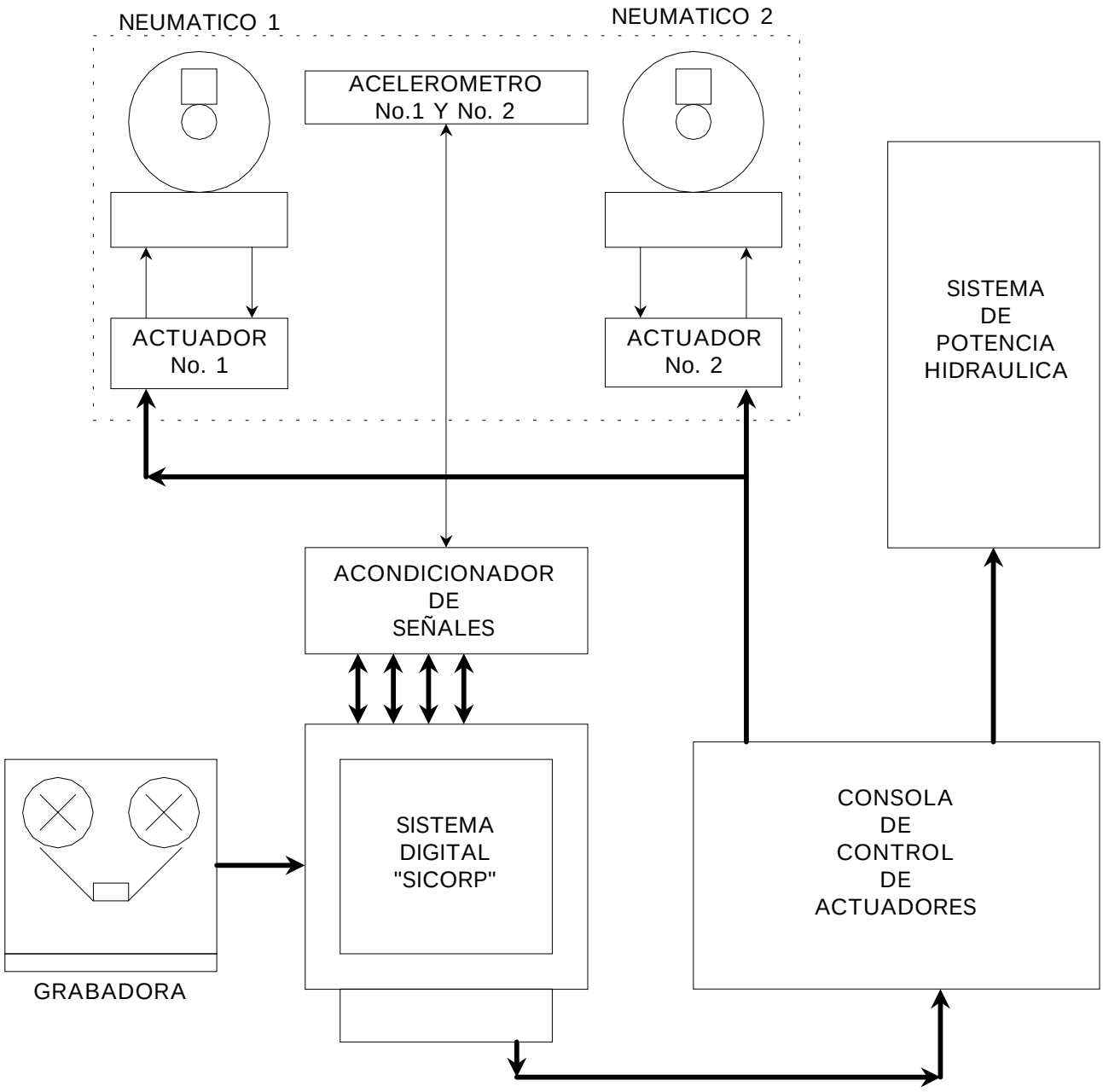


Figura 1. Sistema de control remoto de Parámetros.

estática para cargar de 1 a 8 señales. En operación, si en dicha tarjeta se carga solo una señal al sistema, este podrá soportar hasta 2 horas de señal a 100 Hz y, en 8 canales con 32 "KWords" de señal, un total de 5 minutos a 100 Hz. La reproducción se realiza de manera continua en base a un temporizador programable por comando de usuario a la tarjeta, generando una interrupción para la escritura a los DAC's, y se concluye por solicitud del usuario escribiendo un comando de "paro", que la tarjeta muestrea cada vez que ha concluido un ciclo de escritura de la señal. Esta tarjeta reproduce una señal en modo síncrono usando dos "*Timers*" en cascada, pudiendo programar por software sincronía de 1Hz a 10 KHz. En las **figuras 2** y **3** se muestran los diagramas esquemáticos de esta tarjeta.

Area de datos de Bios y Stack

El área de trabajo para los DAC's se distribuye como se muestra en la **Tabla 1**.

Protocolo de interface con el anfitrión

La interface por software con la tarjeta se realiza por protocolo de encuesta o *"Polling*", donde se muestrean dos bits de estado del dispositivo 8255A (usado como interface). Dichos bits, */OBF* (Output Buffer Full) e *IBF* (Input Buffer Full), determinan las condiciones de lectura o escritura a la tarjeta de reproducción.

La escritura de un*"byte"* al puerto de E/S de la interface define al comando de solicitud, donde el *nibble-alto* es el comando base y el *nibble-bajo* es el subcomando o atributo del comando en su caso. Después de emitir el comando, en la mayoría de los casos, se requiere de datos adicionales para completar éste. A continuación se muestran los comandos típicos usados.

Comando 00H.

Este comando requiere de tres datos para iniciar la transferencia de una señal al área de trabajo de cada uno de los DAC's, como se muestra a continuación. **AL** es el registro de 8 *bits* del 80C188EB para la transferencia tanto de comandos como de subcomandos y datos.

Primera transferencia: el contenido de **AL** tiene el comando y No. del DAC.

Segunda transferencia: **AL** contiene el *byte* alto del No. de datos a transferir.

Tercera transferencia: **AL** contiene el *byte* bajo del No. de datos a transferir.

Las siguientes transferencias son los datos que la tarjeta tomará y depositará en el área de trabajo del DAC solicitado, concluyendo con 2xN transferencias. La máxima transferencia será de 64Kb por DAC.

Comando 10H al 80H.

Los comandos 10 al 80 habilitan de manera individual la operación de cada DAC. Este comando inicia la escritura de los datos a los 8 DAC's o individualmente a cada DAC; el microprocesador de la tarjeta verifica el puerto de control por cada ciclo completo de transferencia de señal para verificar si el usuario desea concluir la generación de la señal, dicha conclusión afecta a todos los DAC's.

Comando B0H.

Este comando programa la frecuencia interna, de tal modo que proporciona una señal de sincronía para la escritura a cada DAC y que es tomada como base para los comandos 10 al 80.

TABLA 1
DISTRIBUCION DE LA MEMORIA PARA LA TARJETA DE REPRODUCCION ROM's

FFFFFH	-	E0000H	128 Kb	BIOS_TARJETA
DFFFFH	-	C0000H	128 Kb	AREA TRABAJO NO INSTALADA
BFFFFH	-	A0000H	128 Kb	AREA TRABAJO NO INSTALADA
9FFFFH	-	90000H	64 Kb	AREA TRABAJO STACK
8FFFFH	-	80000H	64 Kb	AREA TRABAJO DAC_8
7FFFFH	-	70000H	64 Kb	AREA TRABAJO DAC_7
6FFFFH	-	60000H	64 Kb	AREA TRABAJO DAC_6
5FFFFH	-	50000H	64 Kb	AREA TRABAJO DAC_5
4FFFFH	-	40000H	64 Kb	AREA TRABAJO DAC_4
3FFFFH	-	30000H	64 Kb	AREA TRABAJO DAC_3
2FFFFH	-	20000H	64 Kb	AREA TRABAJO DAC_2
1FFFFH	-	10000H	64 Kb	AREA TRABAJO DAC_1
10000H	-	02000H	56 Kb	AREA TRABAJO APLICACION
01FFFH	-	00000H	8 Kb	VECTORES DE INTERRUPCION Y DATOS.

Modo de uso:

1.- Evocar comando B0
2.- Se envía byte alto del valor de la frecuencia a programar.
3.- Se envía byte bajo del valor de la frecuencia a programar.
4.- Retorna conclusión de co-

mando 0B, indicando que éste fue correctamente ejecutado.

Comando C0H.

Este comando verifica los bancos de memoria de RAM estática y reporta si algún banco se encuentra dañado:

Banco 1: solo se reporta 64Kb altos (1)
Banco 2: 64Kb Bajo (2) ; 64Kb Alto (3)
Banco 3: 64Kb Bajo (4) ; 64Kb Alto (5)
Banco 4: 64Kb Bajo (6) ; 64Kb Alto (7)
Banco 5: 64Kb Bajo (8) ; 64Kb Alto No reportado

Si algún banco se encuentra mal, la rutina reporta al usuario el número del banco en el puerto de lectura/escritura y se mantiene en un ciclo infinito y solo se sale con RESET físico a la tarjeta de reproducción. De no encontrar banco dañado, limpia dicha área de datos con cero virtual (07FFH).

Comando E0H.

Este comando limpia el área de trabajo de los DAC's con cero virtual.

Comando F0H.

Este comando permite al usuario poder generar sus propios programas de aplicación y enviarlos a ejecución a la tarjeta, en una área que no está siendo usada por los DAC's. El procedimiento es compilar el programa con un macroensamblador para microprocesadores 8086 u 80186 y transferirlo vía el puerto de lectura/escritura de la interface. El microprocesador transfiere a RAM estática el programa de aplicación a ser ejecutado por éste. El comando transfiere el programa de aplicación al área de aplicaciones y posteriormente hace un salto largo ("JUMP FAR"). Los segmentos quedan inicializados como sigue: CS = DS = ES = 100H.

Nota: Para frecuencias de 1 a 30 Hz. el valor es el equivalente a su valor Hexadecimal, ejem. 30Hz. :

byte-alto = 00,
byte-bajo = 1E.

Para frecuencias mayores es :

1/Frecuencia * 1,000,000.

ejemplo:
2000 Hz = 1/2000 * 1000000 = 500 ; 01F4 Hexadecimal.

donde:
byte-alto = 01,
byte-bajo = F4.

Tarjeta de adquisición de 4 canales de entrada

Cada tarjeta se encuentra formada por cuatro Convertidores Analógico Digital ("Analogic Digital Converter", ADC) ADC912 de Precision Monolithics Inc. de 12 bits de precisión y 18 microsegundos de conversión, ganancia digitalmente programable por canal en rangos de 1, 1.25, 2,5, 10, 20, 50 y 100, separación de voltaje DC y rango dinámico de +/-5V con acoplamiento capacitivo. Cada canal contiene un circuito para realizar 1a. y 2a. integración de la señal para los casos de transductores del tipo acelerómetros y de velocidad, ajustados a una frecuencia de corte de 20Hz, así las señales se encuentran conectadas a un multiplexor analógico 1 de 4, permitiendo con esto medir la señal base, el offset, 1a. integral y 2a. integral. El acoplamiento es capacitivo por cada canal y tiene una protección de sobrevoltaje por arreglo de diodo a +/- 12V.

La supresión del "offset" de la señal puede darse por la acción de medir la señal y el voltaje DC en un lapso de 40 microsegundos y sumarse o restarse, segun el caso, de cada adquisición y obtener una señal "limpia". La tarjeta esta diseñada para operar para un rango de 1Hz a 10Khz. Los ADC's son controlados por un microprocesador 80C188EB @ 16 MHz que tiene una memoria de solo lectura de 128 Kb para Firmware y 512 Kb para "buffers" de adquisición. En las **figuras 3**, **4**, **5** y **6**, se muestran los diagramas esquemáticos del diseño de esta tarjeta.

Area de datos, Bios y Stack

La distribución del área de trabajo queda definida en la **Tabla 2**, la cual define también una área de trabajo para aplicaciones como se mostró en el diseño de la tarjeta de reproducción.

Protocolo de interface con anfitrión

El protocolo de comunicación entre las tarjetas de adquisición y la máquina anfitriona se realiza de forma similar a la expuesta con anterioridad para la tarjeta de re-

Tabla 2
DISTRIBUCION DE MEMORIA PARA 188EB

ROM's		
FFFFFH - E0000H	128 Kb	BIOS_SAA
SRAM's		
7FFFFH - 70000H	64 Kb	AREA TRABAJO DISPONIBLE (No Instalada)
6FFFFH - 60000H	64 Kb	AREA TRABAJO DISPONIBLE (No Instalada)
5FFFFH - 50000H	64 Kb	AREA TRABAJO DISPONIBLE
4FFFFH - 40000H	64 Kb	AREA TRABAJO ADC_4
3FFFFH - 30000H	64 Kb	AREA TRABAJO ADC_3
2FFFFH - 20000H	64 Kb	AREA TRABAJO ADC_2
1FFFFH - 10000H	64 Kb	AREA TRABAJO ADC_1
10000H - 02000H	56 Kb	AREA TRABAJO APLICACION
01FFFH - 00000H	8 Kb	VECTORES DE INTERRUPCION, AREA DE DATOS DE BIOS Y STACK.

producción, donde solo cambia el mapa de puertos definidos para la interface.

Comando 00H.

Para el uso de este comando se requiere que previamente se haya programado la frecuencia de adquisición (comando B0H) ya que requiere que los "*Timers*" estén inicializados y sus interrupciones respectivas habilitadas.

Este comando ofrece al usuario la posibilidad de adquirir con o sin "*offset*", donde en el caso de medición de *"offset"* requiere de una adquisición adicional y una operación de resta o suma según sea el caso, todo esto tendrá como detrimento la pérdida de tiempo en el ancho de banda para la adquisición.

El comando puede adquirir todos los canales a un tiempo o por canal individual como se muestra en la estructura de los subcomandos a continuación.

Primer *byte* será un 00H.
Segundo *byte* será una de las siguientes elecciones.

00: Todos los canales sin OFFSET
01: Todos los canales con OFFSET
10: Solo canal 1 sin OFFSET
11: Solo canal 1 con OFFSET
20: Solo canal 2 sin OFFSET
21: Solo canal 2 con OFFSET
30: Solo canal 3 sin OFFSET
31: Solo canal 3 con OFFSET
40: Solo canal 4 sin OFFSET
41: Solo canal 4 con OFFSET

El 3er. *byte* indica el *byte-alto* del No. de datos a adquirir.
El 4o. *byte* indica el *byte-bajo* del No. de datos a adquirir.

Así se tiene la cuenta total de datos a adquirir.

Al concluir la adquisición y llenar el buffer (que no exceda de 64 o 32 Kb) la rutina reporta a la máquina anfitriona con un 55H que tiene datos disponibles. La rutina realiza 16 escrituras para que la máquina anfitriona reporte con otro 55H si requiere la transferencia de datos, o solicita abortar la adquisición, con la escritura de cualquier otro valor. De no tener ninguna escritura el proceso de adquisición sobreescribirá en el área del ADC hasta concluir la cuenta total definida con anterioridad y nuevamente esperará aviso de transferencia de datos. A continuación se muestra el puerto de la tarjeta definido para monitorear el bit de estado que indica que el ADC se encuentra en proceso de conversión.

Port_BUSY = 140H

D7	D6	D5	D4	D3	D2	D1	D0
B4	B3	B1	B2	0	0	0	0

Comando 10H.

Este comando permite tener acceso a los cuatro dispositivos que generan la ganancia a cada uno de los canales, el comando puede colocar la ganancia igual para todos los canales o de manera individual. El protocolo se muestra a continuación.

Primer *byte* define al comando. Segundo *byte* es la elección de una de las siguientes opciones, donde el *"nibble-alto"* define una opción, el *"nibble-bajo"* define el valor de la ganancia, donde este valor se encuentra entre 0 y 7.

Ganancia :

0X: Todos con la misma ganancia
1X: Ganancia Canal 1
2X: Ganancia Canal 2
3X: Ganancia Canal 3
4X: Ganancia Canal 4

donde X: toma un valor de la ganancia 0 a 7H. de este modo se tiene:

0 = 1X
1 = 1.25X
2 = 2X
3 = 5X
4 = 10X
5 = 20X
6 = 50X
7 = 100X

Los puertos asociados para definir las ganancias se muestran a continuación:

Port_GANAN_1 = 180H

D7	D6	D5	D4	D3	D2	D1	D0
X	X	C4	B4	A4	A3	B3	C3

Port_GANAN_2 = 1C0H

D7	D6	D5	D4	D3	D2	D1	D0
X	X	C4	B4	A4	A3	B3	C3

Los *bits* asociados se definen en los registros que se muestran a continuación:

GAN_SIGNAL_12
Registro de control canales 1 (A1,B1,C1) y 2 (A2,B2,C2).

GAN_SIGNAL_34
Registro de control canales 3 (A3,B3,C3) y 4 (A4,B4,C4).

Comando 20H.
Este comando asigna la señal que el usuario desea adquirir y observar. Permite observar de manera individual cada una de las siguientes opciones:

Primer *byte* es el comando.

Segundo *byte* es una de las siguientes elecciones, donde el *nibble-alto* es la opción a observar y el *nibble-bajo* el canal referido.

0X: Todos con la misma señal
1X: Señal Canal X
2X: 1a. Integral Canal X
3X: 2a. Integral Canal X
4X: Offset Canal X
X: Canal a observar 0 a 3.

A continuación se muestra el puerto que define los *bits* a definir para elegir la opción.

Port_BUS_SIG = 144H

D7	D6	D5	D4	D3	D2	D1	D0
B3	B4	A3	A4	B1	A1	B2	A2

A1 Y B1 para BUS_ANA_C_1.
Registro de señal observada.

A2 y B2 para BUS_ANA_C_2.
Registro de señal observada.

A3 y B3 para BUS_ANA_C_3.
Registro de señal observada.

A4 y B4 para BUS_ANA_C_4.
Registro de señal observada.

Comando B0H.
Este comando opera de manera similar a la tarjeta de reproducción descrita anteriormente.

Comando C0H.
Este comando opera de manera similar a la tarjeta de reproducción descrita anteriormente.

Comando D0H.
Primer *byte* valor de comando. Este comando genera un *Reset* por software realizando un salto largo (JUMP FAR) a la dirección E000:1000.

Comando E0H.
Este comando limpia el área de trabajo de los ADC's con cero virtual (07FFH).

Comando F0H.
Este comando opera de manera similar a la tarjeta de reproducción descrita anteriormente.

Plataforma de trabajo

La plataforma de trabajo se realiza en base a la microcomputadora, una máquina PC/XT o PC/AT usando 3 ranuras de expansión donde se instalan 2 tarjetas de adquisición y 1 tarjeta de reproducción de señal.

Estas tarjetas son operadas por un software de adquisición y reproducción de señales. La adquisición y reproducción es realizada de manera independiente y transparente para el usuario.

Plataforma de software

El software desarrollado se constituye en cuatro fases:

1.- Firmware para tarjeta reproductora de señales.

2.- Firmware para tarjetas de adquisición de señales.

3.- Programa de adquisición y reproducción de señales.

4.- Programa de edición de señales y generación de espectros.

La estructura de los datos en los archivos de señal tiene la siguiente representación:

Los ADC's tienen un rango de 0 a 4095, dentro del cual se representa una escala de +/-5V (10Vpp). La representación de -5V es de 0 a 2047 o 000h a 7FFH y +5V de 2048 a 4095 u 800H a FFFH, por lo cual el signo es ubicado por el usuario por comparación de escalas. Ejemplo: Una onda cuadrada podrá ser representada por la siguiente estructura de datos en un archivo:

00 08 FF 0F FF 0F 00 00

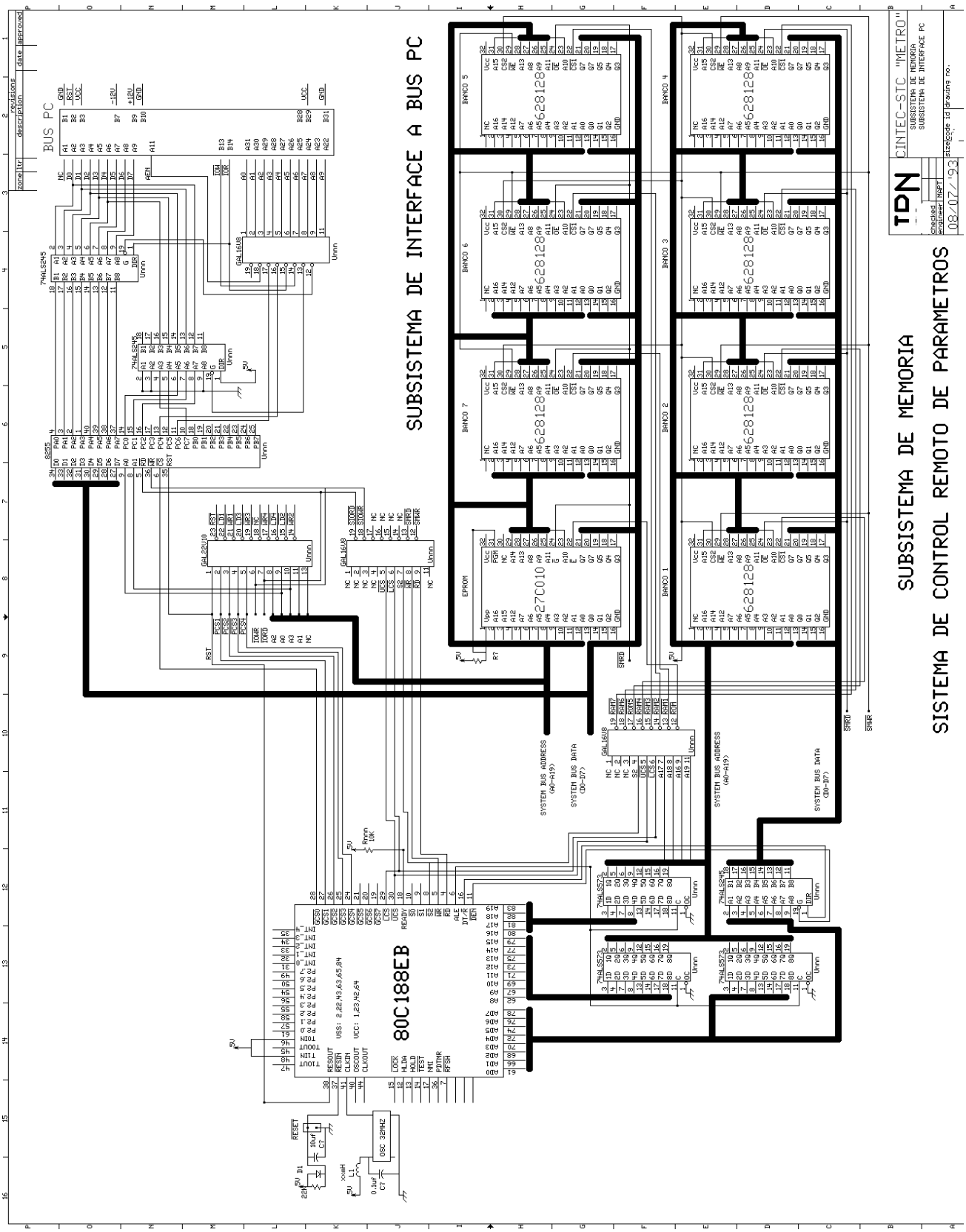

SUBSISTEMA DE INTERFACE A BUS PC
SUBSISTEMA DE MEMORIA
SISTEMA DE CONTROL REMOTO DE PARAMETROS
BUS PC
80C188EB
EPROM
BANCO 1
BANCO 2
BANCO 3
BANCO 4
BANCO 5
BANCO 6
BANCO 7
SYSTEM BUS ADDRESS (A0-A19)
SYSTEM BUS DATA (D0-D7)
CINTEC-STC "METRO"
SUBSISTEMA DE MEMORIA
SUBSISTEMA DE INTERFACE PC

Figura 2

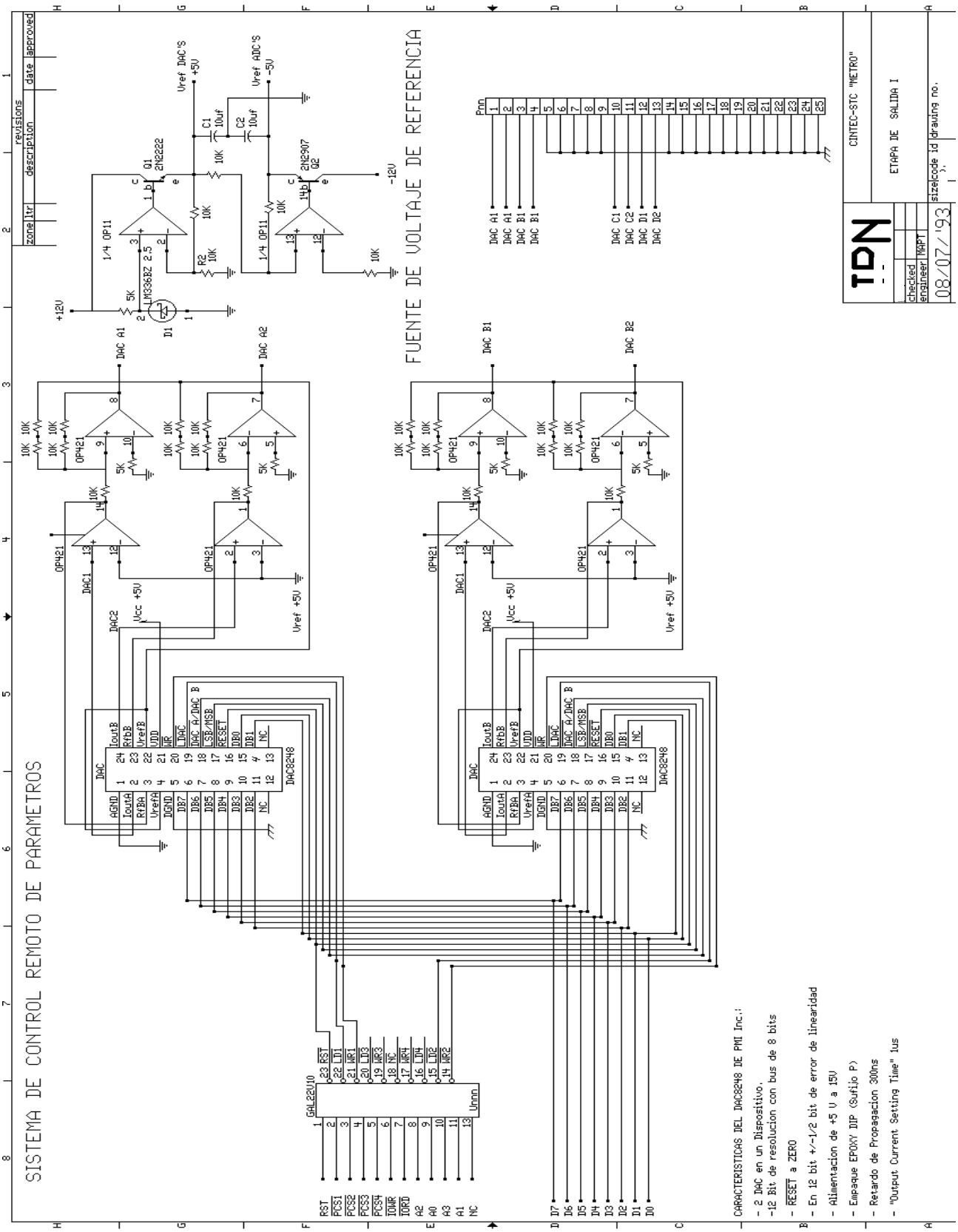

SISTEMA DE CONTROL REMOTO DE PARAMETROS
FUENTE DE VOLTAJE DE REFERENCIA
CINTEC-STC "METRO"
ETAPA DE SALIDA I
TDN
08/07/'93
DAC A1
DAC A2
DAC B1
DAC B2
DAC8248
OP421
GAL22V10
Vref +5V
Vcc +5V
-12V
+12V
CARACTERISTICAS DEL DAC8248 DE PMI Inc.:
- 2 DAC en un Dispositivo.
-12 Bit de resolucion con bus de 8 bits
- RESET a ZERO
- En 12 bit +/-1/2 bit de error de linearidad
- Alimentacion de +5 U a 15U
- Empaque EPOXY DIP (Sufijo P)
- Retardo de Propagacion 300ns
- "Output Current Setting Time" 1us

Figura 3

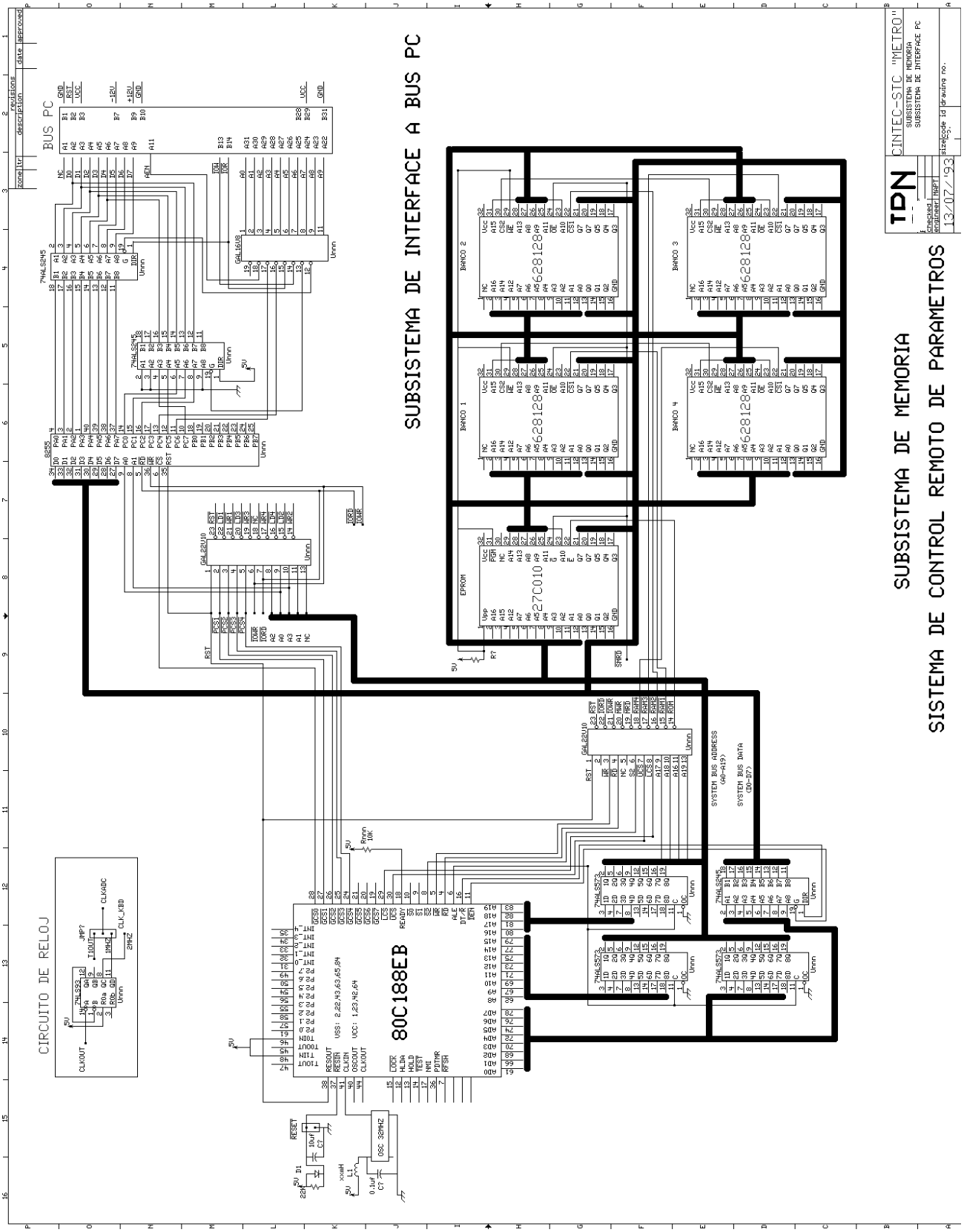

SUBSISTEMA DE INTERFACE A BUS PC
SUBSISTEMA DE MEMORIA
SISTEMA DE CONTROL REMOTO DE PARAMETROS
CINTEC-STC "METRO"
BUS PC
80C188EB
CIRCUITO DE RELOJ
EPROM
BANCO 1
BANCO 2
BANCO 3
BANCO 4
RESET

Figura 4

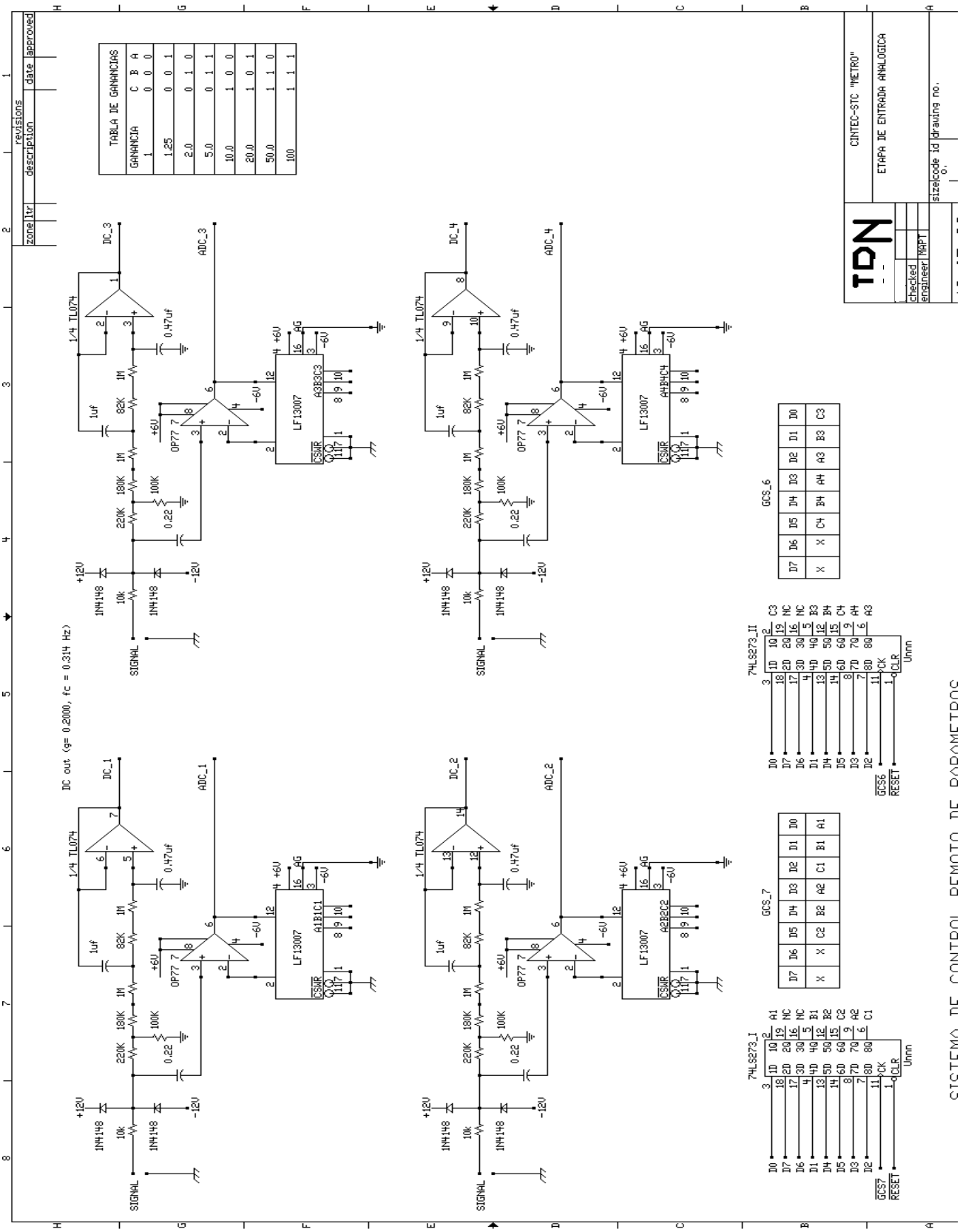


TABLA DE GANANCIAS
GANANCIA C B A
1 0 0 0
1.25 0 0 1
2.0 0 1 0
5.0 0 1 1
10.0 1 0 0
20.0 1 0 1
50.0 1 1 0
100 1 1 1
CINTEC-STC "METRO"
ETAPA DE ENTRADA ANALOGICA
DC out (q= 0.2000, fc = 0.314 Hz)
SISTEMA DE CONTROL REMOTO DE PARAMETROS

Figura 5

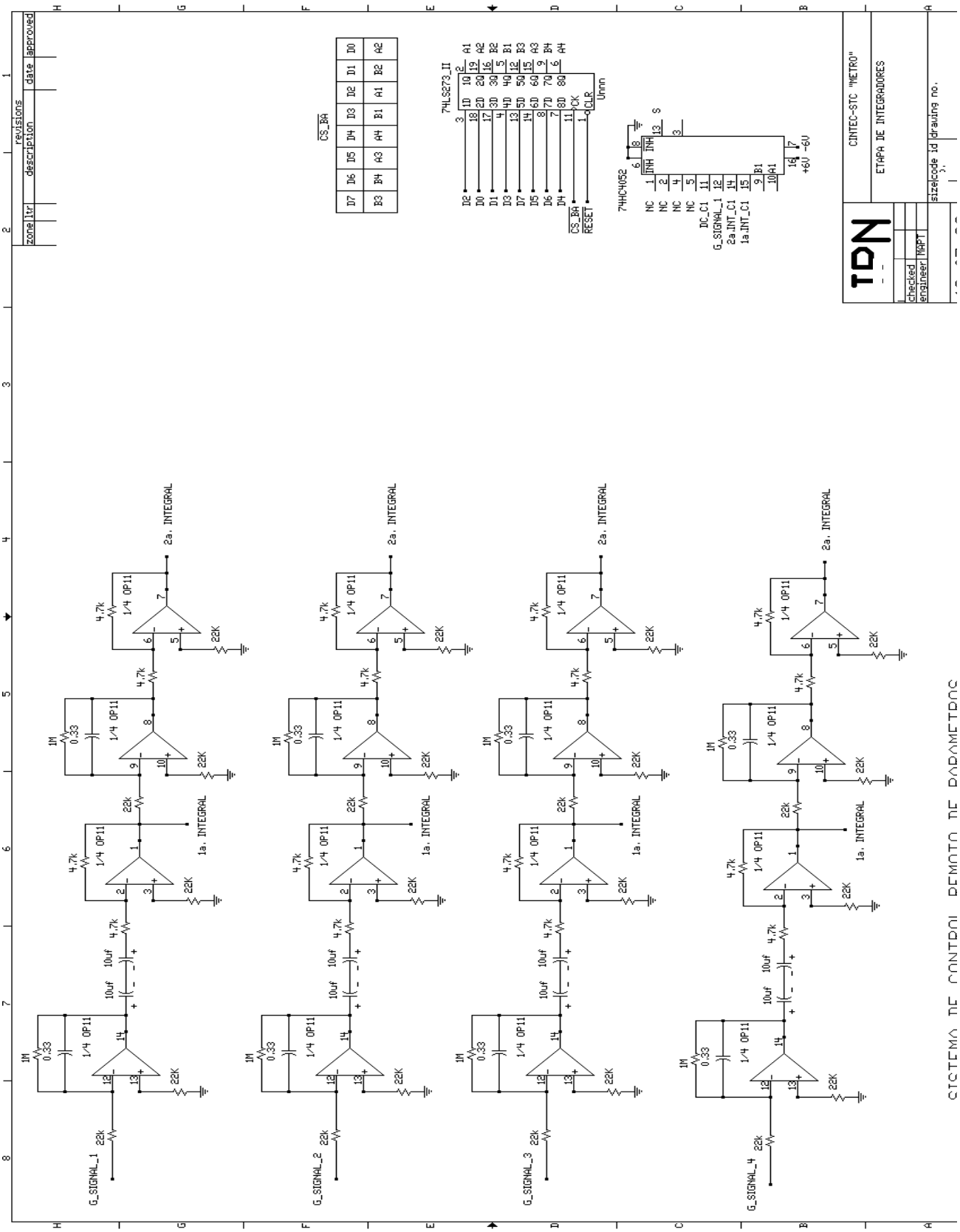

SISTEMA DE CONTROL REMOTO DE PARAMETROS
CINTEC-STC "METRO"
ETAPA DE INTEGRADORES
TDN
2a. INTEGRAL
1a. INTEGRAL
G_SIGNAL_1
G_SIGNAL_2
G_SIGNAL_3
G_SIGNAL_4
74LS273_II
74HC4052
CS_BA
RESET

Figura 6

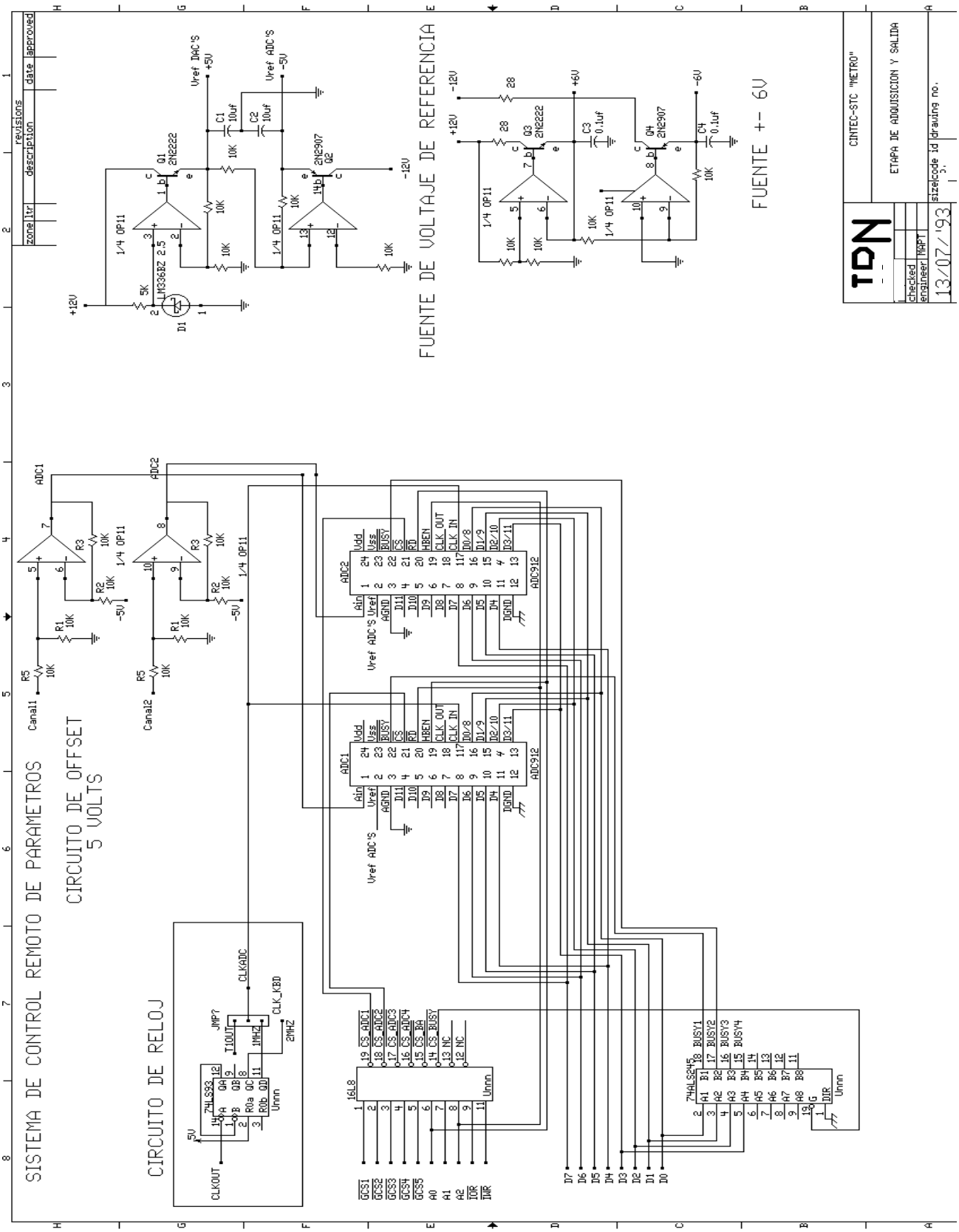

SISTEMA DE CONTROL REMOTO DE PARAMETROS
CIRCUITO DE OFFSET
5 VOLTS
CIRCUITO DE RELOJ
FUENTE DE VOLTAJE DE REFERENCIA
FUENTE +- 6V
CINTEC-STC "METRO"
ETAPA DE ADQUISICION Y SALIDA
TDN
13/07/'93
Vref DAC'S
Vref ADC'S
ADC1
ADC2
Canal1
Canal2
CLKADC
CLK_KBD
CLKOUT
JMP7
1MHZ
2MHZ
ADC912
1/4 OP11
LM336BZ 2.5
2N2222
2N2907

Figura 7

Conclusiones.

La aplicación aquí mostrada define un sistema cerrado que ofrece al usuario prácticamente todas las posibilidades que un sistema de adquisición/reproducción requiere, de tal forma que su empleo podría darse prácticamente dondequiera que el análisis de señales se haga necesario. En **[7]** se presenta una aplicación para calcular la Transformada Rápida de Fourier; dicha aplicación se realizó en lenguaje ensamblador, prevista para ser integrada en *firmware*, de tal forma que pueda quedar integrada. También representa una invitación a los programadores a poder realizar rutinas de filtrado en lenguaje ensamblador para ser integrado al *"firmware"* y tener la gama de aplicaciones completa. Un elemento adicional no contemplado en este diseño es la incorporación de un circuito de sincronía externo. Este circuito ya se tiene diseñado y probado, por lo que haría falta realizar un rediseño de la tarjeta de adquisición e incorporarlo.

Así mismo, el proyecto permite ser estructurado como parte de un sistema para laboratorio de sistemas digitales o procesamiento digital de señales, donde cursos en estas áreas o en diseño de software de interface pueden llevarse en conjunto en una Licenciatura en Ingeniería en Electrónica, Ingeniería de Cómputo o Licenciaturas afines.

Bibliografía.

[1] *"Embedded Microcontrollers and Processors"*. Volume I, Intel Inc. 1992.

[2] *"Analog IC DataBook"*. Precision Monolithics Inc. , 1988.

[3] *"Linear Aplications Specific IC's Databook"*. National Semiconductors, 1993.

[4] Miguel A. Partida Tapia. *"Sistema de Adquisición Autónoma"*. Revista POLIBITS No. 10 Oct-Dic 1992, 11 pags.

[5] Miguel A. Partida Tapia. "*Una tarjeta Controladora Basada en el 80188"*. Revista POLIBITS No.5 Ene-Mar 1990, 14 pags.

[6] Miguel A. Partida Tapia. *"Descripción del BUS de la microSEP 8088 e IPNe16"*. Revista POLIBITS No. 5 Ene-Mar 1990, 5 pags.

[7] Miguel Lindig Bos. *"Rutinas para análisis y procesamiento de señales"*. Revista POLIBITS No.10 Oct-Dic, 1992, 24 págs.

Una arquitectura para procesamiento paralelo

M. en I. Miguel Lindig Bos
Director del CINTEC-IPN

El procesamiento paralelo posee una innegable importancia en el campo de la computación técnica y científica. El paralelismo limitado (esto es, máquinas de un número reducido de procesadores), constituye la base de prácticamente todas las estaciones de trabajo, así como de muchas máquinas "grandes" ("mainframes") y supercomputadoras. El paralelismo masivo se ha aplicado con eficacia a muchos problemas computacionales específicos. Sin embargo, es válido afirmar que las máquinas paralelas han tenido poco éxito como máquinas de "propósito general", en el sentido de proporcionar una razonable eficiencia computacional para cualquier tipo de problema paralelizable.

En un número previo de esta revista se publicaron una serie de artículos relacionados con procesamiento paralelo. En particular, el artículo *"Procesamiento en Paralelo"* **[1]** distingue diferentes niveles de paralelismo en función de la **granularidad** del algoritmo. De manera informal, podemos definir este concepto como la relación entre tiempos de ejecución, y el tiempo requerido para el intercambio de información entre procesadores, demandado por el algoritmo. La granularidad es tanto más fina mientras la cantidad y frecuencia de intercambio de información crecen con relación al tiempo de ejecución. Consecuentemente, una granularidad fina impone mayores demandas a la estructura de intercomunicación de la máquina, que una granularidad más gruesa.

Para los propósitos de este trabajo, aceptemos por "procesamiento paralelo" la operación simultánea de una cantidad arbitraria de procesadores con base en información compartida. Esto implica que todos los procesadores involucrados en un proceso deben tener acceso simultáneo a la misma memoria, una condición que no puede ser satisfecha en la práctica. Máquinas paralelas se diseñan con base en procesadores que poseen memoria propia, y mecanismos de intercambio de información entre procesadores. De manera simplista, el problema de diseño de una arquitectura paralela de propósito general consiste, entonces, en determinar aquella geometría de intercomunicación que se adapte al mayor número de problemas computacionales posible, en forma independiente a la granularidad del algoritmo utilizado, y dentro de un marco de restricciones de tipo físico y/o económico preestablecido.

Una máquina de paralelismo limitado que posee gran importancia práctica es el llamado multiprocesador de bus común. Esta arquitectura, tal vez la más frecuentemente utilizada en las estaciones de trabajo, satisface el principio de generalidad en el sentido de que no impone restricciones a la geometría de intercomunicación requerida por un algoritmo en particular. Esta característica es de fundamental importancia. Estructuras de interconexión fija, tales como procesadores de arreglo e hipercubos, así como estructuras basadas en redes de conmutación, constituyen soluciones óptimas para ciertas clases de problemas, siendo menos eficientes para otros tipos de algoritmos. Pero, por otra parte, en virtud de establecer esta intercomunicación por medio de un solo bus, la arquitectura es altamente sensible a la granularidad del algoritmo. Esto es, la arquitectura posee un "cuello de botella" intrínseco que limita tanto la velocidad de ejecución, como el número máximo de procesadores con que puede contar la máquina. El presente trabajo pretende dar una respuesta preliminar a las siguientes preguntas:

1.- ¿Con una cantidad de líneas físicas de intercomunicación similar al del multiprocesador de bus común, es posible construir una máquina de mayor rendimiento computacional?

2.- ¿Dentro de las limitaciones tecnológicas del momento, es posible rebasar el número máximo de procesadores utilizados en arquitecturas de bus común?

Para ello, se presenta un breve resumen del modelo abstracto de la máquina paralela de acceso aleatorio, y se establece la concordancia entre este modelo y los microprocesadores modernos. A continuación se discute el multiprocesador de bus común y se propone una arquitectura alternativa, basada en el multiprocesador de memoria compartida. Esta arquitectura es caracterizada por un modelo matemático basado en el acceso aleatorio, simultáneo de todos los procesadores, a memoria. Finalmente, se discute la factibilidad de construcción de una máquina basada en esta arquitectura y se presentan estimaciones de rendimiento computacional.

El modelo PRAM

La máquina paralela de acceso aleatorio ("Parallel Random Access Machine", PRAM), es un modelo idealizado de multiproceso (**figura 1**)**[2]**. En su forma original **[3]**, el modelo consta de una colección infinita de procesadores, cada uno de los cuales tiene acceso a una memoria global, utilizada para comunicación entre procesadores. Cada procesador cuenta con una memoria local en forma de un conjunto ilimitado de registros. El conjunto de instrucciones incluye las transferencias a memoria (tanto local como global) LOAD, STORE y operaciones tales como suma y resta. Registros globales no pueden ser usados como base para direccionamiento indirecto. Tanto la memoria local, como la global, tienen capacidad para almacenar enteros de longitud arbitraria.

El modelo descrito en la referencia **[3]** incluye dispositivos de entrada/salida, así como las instrucciones correspondientes. Todos los procesadores ejecutan un mismo programa, que forma parte de los procesadores. Esto es, existen tantos modelos PRAM como programas. El modelo PRAM respeta el principio de **costo unitario**, en el sentido de que el tiempo de ejecución de cada una de las instrucciones definidas para el modelo es el mismo. Lo anterior incluye accesos a memoria y operaciones de entrada/salida. Se han propuesto una serie de modificaciones al modelo original. En su forma más aceptada, el modelo PRAM está basado en:

1.- Un número de identificación, **PIN** (processor identification number), único para cada procesador. Cada procesador puede leer su PIN en un registro de lectura dedicado.

2.- Todos los procesadores ejecutan la misma instrucción. Si esta instrucción es función de un PIN en particular, aquellos procesadores que no poseen este PIN ejecutan lazos de sincronización.

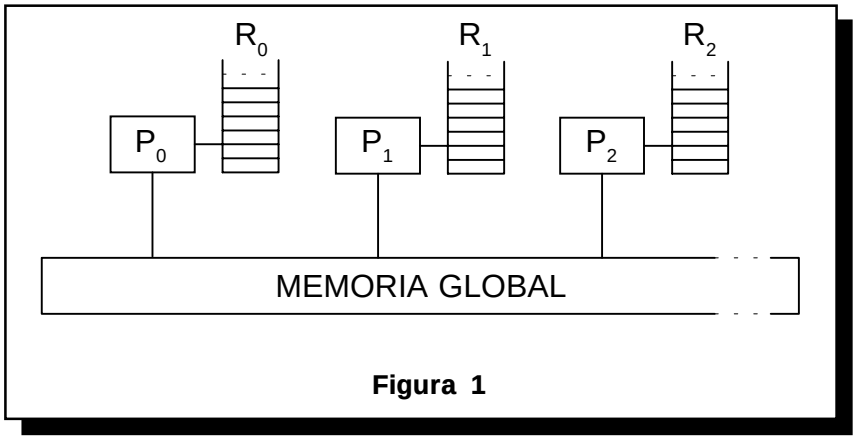


Figura 1

Desde el punto de vista del conjunto de instrucciones, en la actualidad se acepta un modelo que incluye la multiplicación, división y desplazamientos de datos sobre un número de bits arbitrario. Por otra parte, existen subdivisiones de este modelo en función a que la lectura y/o escritura concurrente (por más de un procesador) a la misma localidad de memoria global es permitida, o no.

El microprocesador como elemento de procesamiento

Definamos como procesador el conjunto formado por microprocesador, controlador de memoria *caché* y dispositivos de memoria *caché* (**figura 2**).

Como elemento de procesamiento de una PRAM, este procesador ejecuta un superconjunto del juego de instrucciones definido como completo. Por otra parte, la memoria caché puede, en principio, desempeñar las funciones de la memoria local, si se omite el requisito de tamaño ilimitado. La condición relativa al tamaño arbitrario de los datos no es satisfecha. Sin embargo, dispositivos modernos aceptan formatos de hasta 64 bits. En consecuencia, para la gran mayoría de los problemas computacionales, esta limitación no es relevante.

En materia de tiempos de ejecución, el principio de costo unitario puede considerarse aplicable (particularmente en procesadores tipo

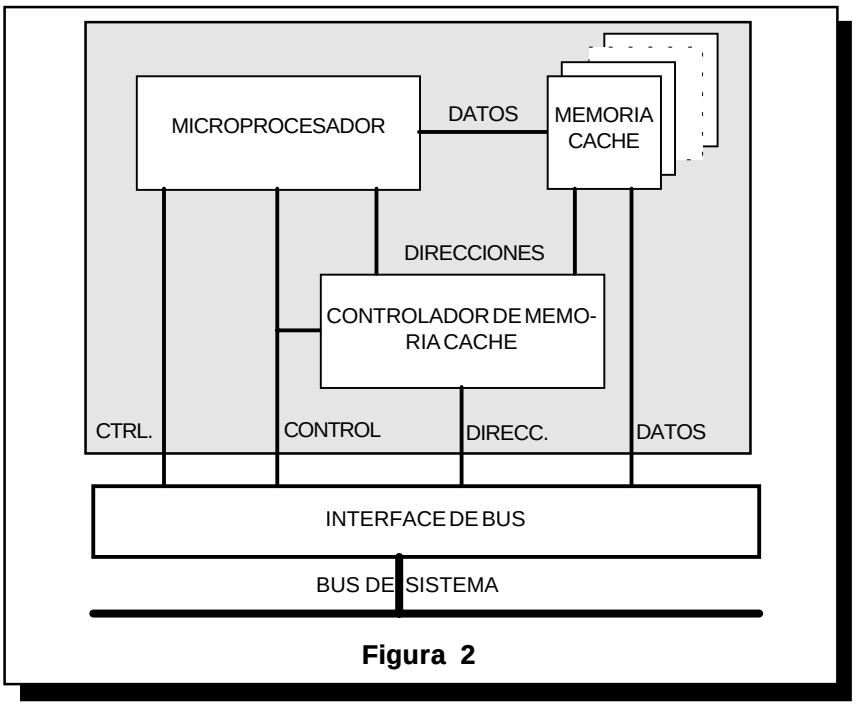


Figura 2

RISC), con la posible excepción de las instrucciones de desplazamiento, multiplicación y división. El principio de costo unitario no es aplicable a la memoria global. No solamente es más lenta que la memoria caché por razones de costo, sino por la mayor separación física entre este subsistema y los procesadores. Esta separación requiere dispositivos de transmisión / recepción, cuyos retardos de propagación son significativos en términos de tiempos de ciclado de los procesadores. De este hecho se desprende una consecuencia importante. La necesidad de intercomunicación de datos entre procesadores involucra procesos que son más lentos que los accesos a memoria caché. En consecuencia, un procesador en una arquitectura paralela tiene un rendimiento individual menor, comparado con su rendimiento en un esquema uniprocesador, independientemente de la arquitectura de que se trate.

El multiprocesador de bus común

Esta arquitectura es, tal vez, la más sencilla desde el punto de vista conceptual (**figura 3**). La comunicación entre procesadores y memoria global se efectúa por medio de un solo bus (de aquí el nombre de la arquitectura). El número de accesos a memoria global se reduce a un mínimo, permitiendo la comunicación directa entre procesadores. Esta comunicación se establece mediante un segundo bus, el llamado **bus de indagación** ("snoop bus") **[4]**.

En rigor, el multiprocesador de bus común posee características tanto del modelo PRAM, como del modelo de interconexión fija. Sin

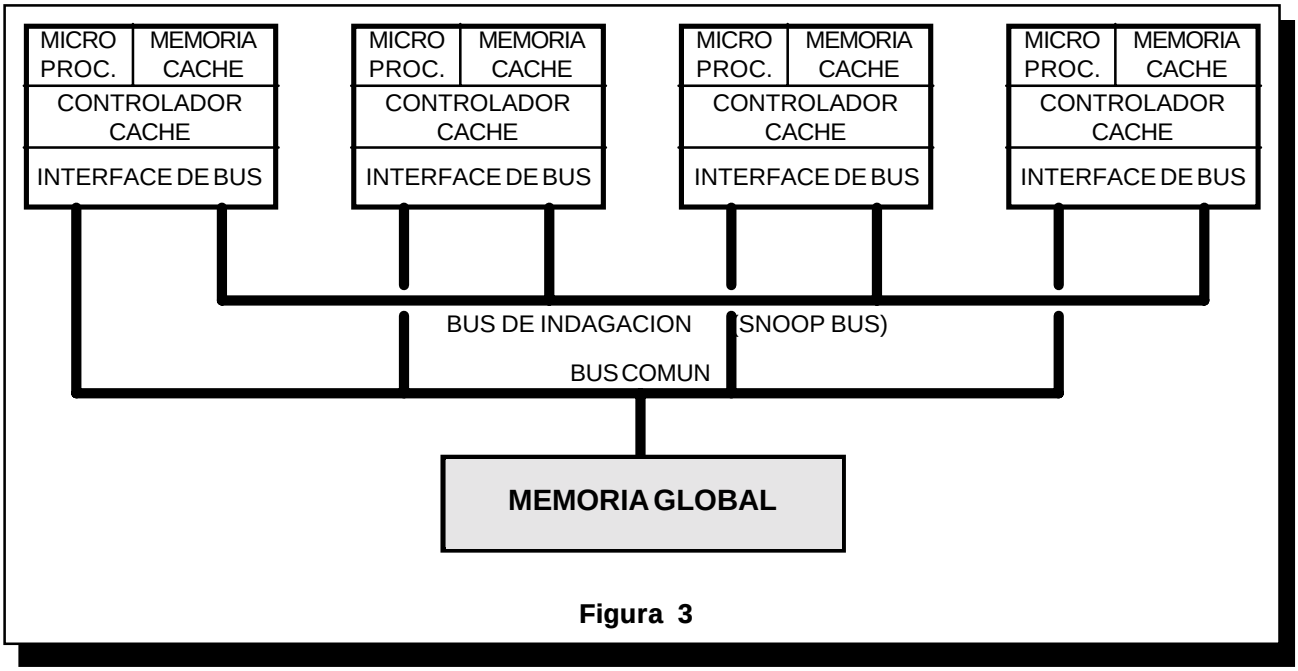


Figura 3

embargo, es posible visualizar esta arquitectura como una PRAM, en virtud de que la función del bus de indagación consiste en simular una memoria global más rápida que, la de hecho, existente. En efecto, la comunicación directa entre procesadores puede emular la función de la escritura de actualización, sin afectar a la memoria global.

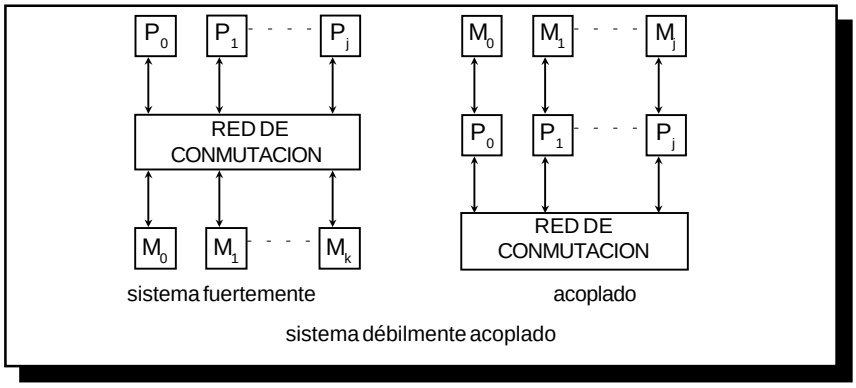
La comunicación directa entre procesadores puede efectuarse, en principio, con un costo unitario, si se garantiza la operación síncrona de los procesadores. Sin embargo, el uso de un bus común obliga a la implantación de un protocolo, para evitar la utilización simultánea del bus por más de una pareja de procesadores. Cualquiera que sea este tiempo, es evidente que la cantidad total de tiempo requerida crece linealmente con el número de intercomunicaciones que deben establecerse en un momento dado. Se puede probar que, para una granularidad dada, la máquina se "satura" después de un cierto número de procesadores, esto es, el rendimiento computacional ya no crece al aumentar el número de procesadores.

La diferencia fundamental entre el modelo PRAM y el multiprocesador de bus común consiste en el mecanismo de intercomunicación. Para la PRAM el medio de comunicación es la memoria global. La intercomunicación entre dos procesadores requiere dos accesos a memoria, uno de escritura por un procesador dado, seguido de uno de lectura por otro procesador. Para el multiprocesador de bus común, el medio de comunicación es el bus de indagación. Un procesador dado escribe a su memoria local (memoria caché), y la lectura por parte de otro procesador es efectuada a esta misma memoria local, en forma "transparente" a la operación del procesador que efectuó la escritura. Este proceso es, desde luego, más rápido que la comunicación por medio de la memoria global. Nótese que lo anterior es aplicable a cualquier esquema de interconexión fijo.

El multiprocesador de memoria compartida

El término **multiprocesador** abarca todas las arquitecturas posibles, integradas por varios procesadores y una, o más, unidades de memoria. Desde el punto de vista de comunicación de datos entre procesadores, se distingue entre dos modelos, el de una memoria global, compartida, y el de memoria distribuida.

En el modelo de memoria distribuida, la memoria global es dividida en tantos módulos como procesadores existan en la estructura. Cada procesador posee un módulo, y el intercambio de información se produce a través de los procesadores.



De acuerdo con **[5]**, se trata de un multiprocesador débilmente acoplado (**figura 4**).

En el modelo de memoria compartida, los procesadores tienen acceso a toda la memoria global. En el caso del multiprocesador de bus común, esta memoria constituye un solo módulo. En general, esta memoria puede ser dividida en ***M*** módulos. Cada uno de ***N*** procesadores es unido mediante un bus de datos a una red de conmutación. Lo mismo se cumple para los ***M*** módulos de memoria, esto es, la red se encuentra entre procesador y memoria (modelo fuertemente acoplado) **[5]**. Nótese que ***N*** puede ser diferente a ***M***.

Con este esquema, la cantidad de accesos simultáneos posibles a memoria aumenta proporcionalmente al número de módulos de memoria. Sin embargo, existe la posibilidad de que se generen colisiones, esto es, el intento de dos o más procesadores por acceder al mismo módulo. Tal vez por esta razón, el modelo de memoria compartida ha recibido poca atención en la literatura reciente. Aquí se analiza este modelo, por lo siguiente:

1.- El modelo de memoria compartida es el más cercano al modelo abstracto. Todos los modelos de memoria distribuida pueden ser mapeados en este modelo, por lo que es, intrínsecamente, el de mayor generalidad.

2.- El modelo de memoria compartida no considera la existencia de memorias locales. Esto no corresponde a la definición

de procesador para una PRAM. Bajo ciertas consideraciones, la memoria caché puede operar como memoria local, por lo que conviene re-examinar el modelo.

3.- Un multiprocesador de memoria compartida puede constituir un grupo ("cluster") de procesadores, en una arquitectura híbrida de memoria distribuida **[6]**. Por lo tanto, sus características definen las características globales de la arquitectura.

La discusión subsecuente se limita al caso de un número de módulos de memoria igual al de procesadores. También, consideramos una memoria de (**NL**) localidades, dividida en **N** módulos iguales, donde cada localidad almacena el número de bits correspondiente al formato de datos.

En el Centro de Investigación Tecnológica en Computación (CINTEC) del Instituto Politécnico Nacional se está evaluando la factibilidad de construcción de una máquina con el siguiente cambio en la estructura descrita. La red de conmutación es substituida por un conjunto de buses, tal que cada procesador está conectado a un bus propio. Estos buses son comunes a los módulos de memoria, esto es, cada módulo se conecta al conjunto de buses, por medio de dispositivos de tercer estado (**figura 5**).

Este esquema permite la interconexión de cualquier procesador a todos los módulos de memoria, así como establecer tantas comunicaciones simultáneas, como procesadores existentes en el esquema. Llamaremos a este modelo **multiprocesador de memoria compartida de buses múltiples (MCBM)**.

Con relación al multiprocesador de bus común, parece evidente que este modelo posee un rendimiento relativo mayor. Pero, en virtud de que el modelo no excluye la posibilidad de colisiones, también es evidente que no puede ser tan eficiente como el modelo teórico de la PRAM. Por otra parte, accesos simultáneos deben ser atendidos secuencialmente, esto es, el esquema requiere arbitraje.

La probabilidad de que ocurra una colisión en un acceso dado depende de la naturaleza del algoritmo, esto es, del tipo de simetría inherente a la gráfica del problema. Un caso límite puede constituir el acceso aleatorio, esto es:

1.- La probabilidad asociada a la dirección A_k es la misma para toda k.
2.- La ocurrencia de la dirección A_j es independiente de A_k.

Supóngase que el tiempo de acceso sea la unidad. El acceso simultáneo de dos procesadores a un mismo módulo debe efectuarse secuencialmente, esto es, requiere 2 unidades, y así sucesivamente. Para N procesadores, la cantidad total de intercambios posibles de información es N^N. Durante un conjunto de accesos dado, se pueden producir colisiones a más de un módulo de memoria. Sea n_j el número de colisiones de *j* procesadores, donde *j* corresponde al conjunto de más elementos. Entonces, el tiempo medio de acceso, τ_m, está dado por:

$$\tau_m = (1n_1 + 2n_2 + \ldots + Nn_N)/N^N \quad \ldots\ldots 1$$

Se puede probar que el número de accesos que involucran n colisio-

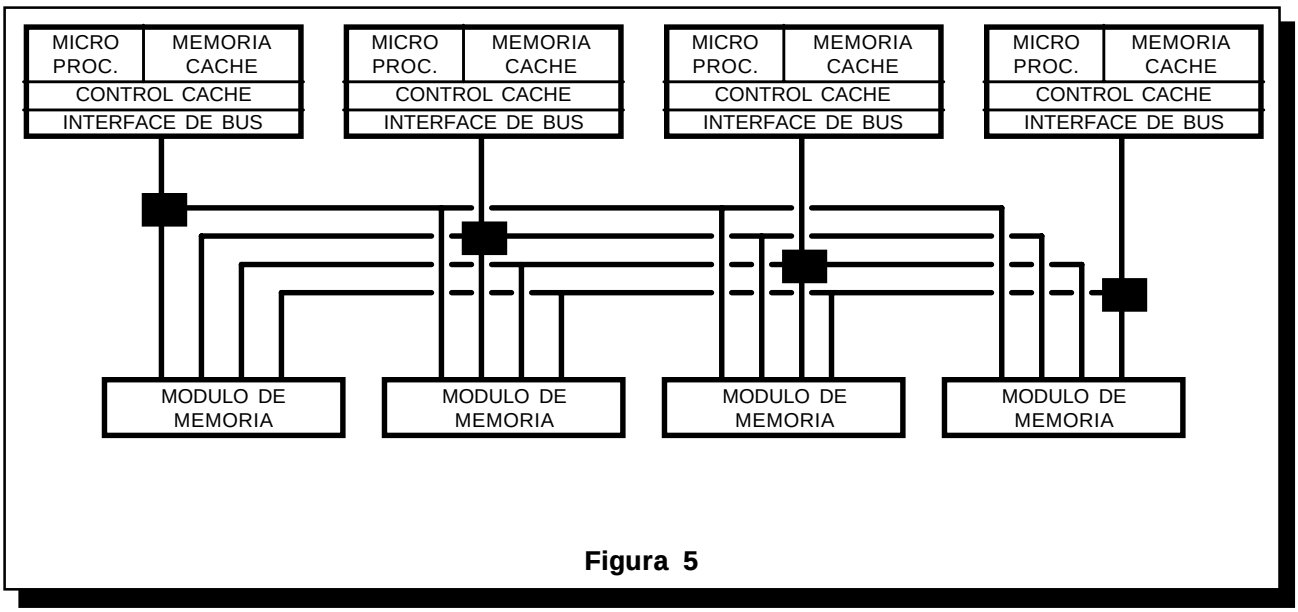


Figura 5

nes es numéricamente igual a la cantidad de términos cuyo exponente máximo es n, de la expansión de la siguiente expresión:

$$P = (p_1 + p_2 + ... + p_N)^N \qquad .....2$$

donde p_m^n significa que **n** procesadores solicitan el acceso al módulo de memoria **m**.
Ejemplo. Sea N = 3. Entonces:

$$\begin{aligned} P &= [p_1+p_2+p_3]^3 \\ &= p_1^3 + p_2^3 + p_3^3 \\ &+ 3[p_1^2(p_2+p_3) + p_2^2(p_1+p_3) + p_3^2(p_1+p_2)] \\ &+ 6\, p_1p_2p_3 \end{aligned}$$

Aquí, **P** posee 3 términos de orden 3, 18 de orden 2 y 6 de orden 1. Esto es, en 3^3 accesos se producen 3 que requieren 3 tiempos de ejecución, 18 que requieren 2 y 6 que requieren 1 tiempo de acceso cada uno. El tiempo medio de acceso es, de acuerdo con (1):

$$\tau_m = [6 + 2(18) + 3(3)]/27 = 1.889$$

La **tabla 1** reproduce el tiempo medio de acceso para hasta 16 procesadores. Para un multiprocesador de bus común, el tiempo medio de acceso es numéricamente igual al número de procesadores de la estructura. Sin embargo, el tiempo requerido para intercambiar información entre procesadores es menor que el tiempo de acceso a memoria (el costo unitario es menor). Por lo tanto, se puede concluir, con base en la **tabla 1**, que existe un valor inferior para el número de procesadores N, tal que, para cantidades mayores, el rendimiento relativo del multiprocesador tipo MCBM es superior al del bus común.

N	τ_m(N)	N	τ_m(N)
1	1.000	9	2.676
2	1.500	10	2.749
3	1.889	11	2.815
4	2.125	12	2.876
5	2.286	13	2.932
6	2.408	14	2.985
7	2.509	15	3.033
8	2.597	16	3.078

Tabla 1

Los valores dados en la **tabla 1** se ajustan, aproximadamente, a la siguiente expresión :

$$\tau_m = (1 + \log_2 \mathbf{N})^{0.7} \qquad .....3$$

Esto es, el tiempo medio de acceso crece en forma menor al logaritmo del número de procesadores.

Conviene hacer una observación final con relación a este modelo. El número de buses de interconexión crece linealmente con N. Esto impone un límite práctico a la escalabilidad del modelo, en tanto que todos los buses conectan a todos los módulos de memoria.

Consideraciones sobre la realización física

Se hizo un estudio preliminar de factibilidad de construcción del multiprocesador tipo MCBM, basado en el procesador PENTIUM de INTEL **[7,8]**, con un tiempo de ciclado de 16 nanosegundos (una frecuencia de operación de 66 MHz), y una memoria caché secundaria de 256 Kbytes, operando sin estados de espera. El bus de datos de este procesador es de 64 bits, y el tamaño de línea de la memoria caché, de 256. El diseño del procesador permite la operación de subsistemas externos a la frecuencia del procesador, o submúltiplos de esta frecuencia. También soporta la operación síncrona, como asíncrona, de dispositivos externos. Se consideró que la memoria opera a la mitad de la frecuencia del procesador.

El número máximo de procesadores que puede ser utilizado en un multiprocesador tipo MCBM depende, esencialmente, del número de líneas físicas, esto es, de buses de interconexión, que pueden ser proporcionadas por una estructura de interconexión dada ("backplane"). Considerando reglas de diseño de 25 milésimas de pulgada, cada capa de circuito impreso soporta 40 líneas por pulgada, o 256 en poco más de 6 pulgadas. En un circuito multicapa, en el que se separan las líneas de comunicación por planos de tierra, es factible proporcionar 1024 líneas con tecnología convencional. Esto equivale a 32 buses de 32 bits, o 16 de 64 bits. Para el diseño de una máquina, la línea de comunicación debe ser vista como un recurso computacional, que posee un costo asociado. Para el multiprocesador de bus común, y para el procesador considerado, el total de líneas de intercomunicación es del orden de 200, incluyendo señales de control y de direccionamiento del bus de indagación. A este valor habría que sumar las líneas utilizadas para arbitraje de acceso, que depende del número de procesadores. Esto es, el multiprocesador de bus común requiere por el orden de 250 líneas físicas.

Para los módulos de memoria, se consideró una tecnología basada en dispositivos dinámicos de 60 nanosegundos de tiempo de acceso. En el modo de operación más sencillo, las memorias dinámicas almacenan secuencialmente la dirección de fila y la de columna para, posteriormente, proporcionar o aceptar el dato seleccionado. Esto es, los módulos de memoria pueden ser diseñados con base en un bus multiplexado, por medio del cual se transmite, primero, la dirección y comando, y posteriormente el dato.

Se consideró que los módulos de memoria soportan accesos en modo página para transferencias secuenciales de más de una palabra. Adicionalmente, se consideró el caso de módulos provistos con memorias tipo FIFO ("first in - first out") para el último dato, o línea, leída/escrita.

Los resultados de este estudio, expresados en el número de ciclos de procesador requeridos para diferentes tipos de transferencias de datos, se reproducen en las **tablas 2** y **3**. La **tabla 2** corresponde al intercambio aleatorio de líneas de 4 palabras de 64 bits (a) y de datos de 64 bits (b) para 4, 8 y 16 procesadores. Las cantidades sombreadas corresponden a 256 líneas físicas de intercomunicación, esto es, a un número de líneas comparable a las requeridas por el multiprocesador de bus común. Los valores restantes corresponden a configuraciones de 512, y 1024, líneas físicas. La tabla 3 reproduce los resultados correspondientes al intercambio sin colisiones para los mismos casos descritos arriba.

Conclusiones

El multiprocesador tipo MCBM es la aproximación más rápida posible al modelo ideal de una PRAM. Bajo la hipótesis de costo unitario, el tiempo medio de intercambio aleatorio de información sin colisiones es 2, y de $1+\tau_m$, para el caso de intercambios aleatorios con colisiones.

Desde el punto de vista de factibilidad de construcción, es posible construir máquinas de hasta 32 procesadores y buses de 32 bits, con base en tecnología convencional. Lo anterior equivale a más de 3,000 millones de instrucciones por segundo para el procesador considerado (PENTIUM), y por el orden de 4,800 millones para el procesador i860 **[9]**, ambos de INTEL. La complejidad de diseño, así como el costo, es básicamente comparable al de un multiprocesador de bus común. Si bien el esquema de buses múltiples requiere dispositivos de tercer estado para activar un bus en particular, no exige, en cambio, buses de muy alta velocidad, esenciales para el modelo de bus común. Por otra parte, el protocolo de escritura de actualización da origen a controladores de memoria, por lo menos un orden de magnitud más complejos, que los correspondientes a escritura transparente **[10]**.

Una de las características más importantes del modelo MCBM, es la baja degradación que sufre el rendimiento de un procesador, comparado con su rendimiento individual. El único retardo asociado a la arquitectura es el tiempo de tránsito

Tipo de dato:	-a-			-b-		
Cant. de procesadores:	4	8	16	4	8	16
Multiproc. de bus común:	112	224	448	48	96	192
MCBM, 32 bits, sin FIFO:	98	119	142	43	52	62
MCBM, 32 bits, con FIFO:	85	104	123	30	36	43
MCBM, 64 bits, sin FIFO:	64	78	92	38	47	55
MCBM, 64 bits, con FIFO:	51	62	74	26	31	37

Tabla 2

Tipo de dato:	-a-			-b-		
Cant. de procesadores:	4	8	16	4	8	16
Multiproc. de bus común:	112	224	448	48	96	192
MCBM, 32 bits, sin FIFO:	46			22		
MCBM, 32 bits, con FIFO:	40			16		
MCBM, 64 bits, sin FIFO:	30			18		
MCBM, 64 bits, con FIFO:	24			12		

Tabla 3

por los dispositivos de tercer estado. Este retardo es comparable al asociado a dispositivos alimentadores de bus (buffer), que forman parte de cualquier diseño tipo uniprocesador. En consecuencia, el rendimiento global depende, básicamente, de la cantidad de estados de espera introducida por los módulos de memoria. Dependiendo del rendimiento deseado, el tiempo de acceso puede reducirse a dos ciclos de memoria, con técnicas de diseño bien establecidas.

El modelo aquí propuesto es universal, en tanto que admite cualquier gráfica de problema, y no requiere algoritmos de tránsito. La granularidad de los algoritmos que pueden ser eficientemente ejecutados es mucho más fina a la de otros multiprocesadores. Con ello, se abren nuevas perspectivas para la investigación de algoritmos. Por otra parte, no existe limitación al formato de datos, en el sentido de que la fragmentación del mapa de memoria puede ser programada, y hasta dinámicamente reconfigurada, durante la ejecución del programa.

Una de las ventajas del protocolo de escritura transparente, es la implícita coherencia de memoria. Esto simplifica la programación y sincronización de procesos considerablemente. Por otra parte, evita el tener que concentrar los resultados de un programa en memoria global al término de su ejecución ("flushing"). Finalmente, permite que el multiprocesador constituya un elemento de procesamiento en una arquitectura de paralelismo masivo. En este sentido, el modelo propuesto amplía los límites del paralelismo físicamente realizable en, por lo menos, un orden de magnitud.

Bibliografía

[1] Lindig, B. Miguel . *"Procesamiento en paralelo"*. Polibits, Año V, Número 11 Oct-Dic. 1993

[2] Codenotti, B., Leoncini,M. *"Introduction to Parallel Processing"*. Addison-Wesley International Computer Science Series, 1993.

[3] Fortune, S., Wyllie, J. *"Parallelism in Random Access Machines"*. Proc. of 10th Annual ACM Symp. on Theory of Computing, 1978, pp. 114-118.

[4] Cache Tutorial. INTEL Corp.,1991.

[5] Blasi, M. *"Computer Architecture"*. Addison-Wesley International Computer Science Series, 1990.

[6] Lenoski, D. et al. *"The Stanford Dash Multiprocessor"*. IEEE Computer, vol.25, no.3, pp.63-79, marzo de 1992.

[7] Pentium Processor User's Manual (3 tomos). INTEL Corp., 1993.

[8] Reilly, J. *"Designing with the PentiumTM Processor, 82496 Cache Controller and SRAM CPU-Cache Chip Set"*. Application Note AP-481, INTEL Corp., julio de 1993.

[9] i750, i860, i960 Processors and Related Products. INTEL Corp., 1994

[10] Taufik T. *"Cache and Memory Design Considerations for the Intel486TM DX2 Microprocessor, Microprocessors"*. Vol.1, pp. 2-1137-1173, INTEL Corp., 1993.

Diseño y construcción de una tarjeta convertidora de 8 canales A/D y 4 canales D/A

M. en C. Romeo Urbieta Parrazales
Profesor e Investigador del CINTEC-IPN
Ing. Ignacio Minjares Tarazena
Profesor e Investigador del CINTEC-IPN

El artículo trata del diseño y construcción de una tarjeta de expansión para PC con ocho canales de entrada, con conversión de señales analógicas a señales digitales, basado en el Convertidor Analógico Digital ("Analogic Digital Converter", ADC) ADC 0809 de la empresa National Semiconductor, funcionando a una frecuencia de reloj de 640 KHz, y ventana de 0 a 5 voltios; cuatro canales de salida con conversión de señales digitales a analógicas, con el Convertidor Digital Analógico ("Digital Analogic Converter", DAC) DAC0808, también de National Semiconductor, funcionando en una ventana de 0 a 5 voltios, con amplitudes 20 mV por bit. La codificación de los convertidores de señales digitales a analógicas se efectuó con un dispositivo lógico programable. La entrada y salida de datos de los convertidores se llevó a cabo con la interface paralela de Intel, el 8255, con puertos de direccionamiento 100h y 101h.

Introducción

La computadora digital tipo PC se empleará para controlar las variables analógicas de un proceso industrial; para esto, forzosamente se requiere de una tarjeta convertidora de señales A/D y D/A (ver "Definiciones" al final del artículo), con una amplitud de voltaje, corriente, y velocidad específica, determinados por estas variables.

Generalmente, los procesos son multivariables, es decir, son de más de una variable, y se clasifican como: variables independientes, y variables dependientes. Las variables independientes entran al A/D y las variables dependientes se obtienen del D/A. El diseño de la tarjeta integra un solo ADC con 8 entradas, y 4 ADC de una salida cada uno.

Los parámetros de los circuitos de la tarjeta, son: la amplitud de voltaje y la corriente de entrada - salida de los ADC's y DAC's, siendo en este caso de 0 a 5 voltios @ 20 mA.; la velocidad de adquisición de los datos en los ADC's, los que pueden muestrear señales hasta diez muestras por milisegundo, es decir, sirven para controlar procesos con tiempos de restablecimientos de 1 milisegundo ("Settling time"), dando períodos de muestreo de 1/10 de milisegundo (parámetro de interés en el análisis de la teoría de control). Los DAC's pueden manipular niveles de voltaje de 20 mV por bit. La precisión, y la no linealidad de los ADC's y los DAC's, es de 1/2, y 1 bit menos significativo, respectivamente.

La tarjeta está construida con 15 bases para circuitos integrados (C. I.), dos conectores DB9, un bus PC dorado, 380 orificios y líneas. Se empleo el paquete de diseño Smartwork para realizar el diseño del circuito impreso. Los 15 C. I. son de tecnología TTL y CMOS.

El programa de prueba empleado tiene básicamente dos subrutinas: Una que maneja a los D/As y otra a los A/Ds, usando el lenguaje C.

1. Descripción general

La tarjeta convertidora de señales A/D y D/A consiste de los siguientes circuitos, ver **figura 1**.

- **o** Un Controlador de puertos paralelos 8255.
- **o** Un amplificador digital bidireccional de datos 74LS245.
- **o** Un decodificador de selección de puerto 74LS138.
- **o** Un decodificador para la transferencia de datos de entrada y salida de los convertidores A/D y D/A, PAL16L8.
- **o** Cuatro arreglos octales de flip flop tipo D, 74LS373.
- **o** Cuatro convertidores digital a analógico DAC0808.
- **o** Un divisor de frecuencia 74LS93.
- **o** Un amplificador analógico con cuatro amplificadores operacionales LM348.

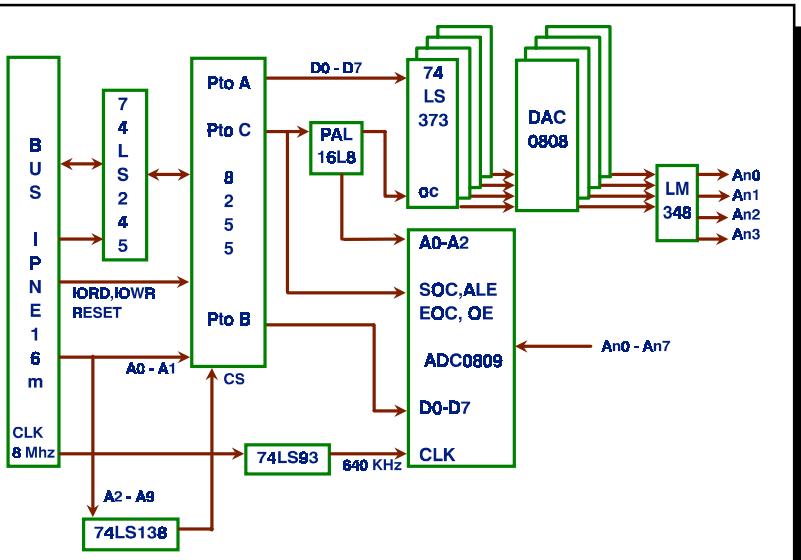


Figura 1 Diagrama a bloques de la tarjeta convertidora A/D, D/A

1.1 Entradas del Controlador de Puerto Paralelos

Las líneas de entrada son trece: ocho de datos y cinco de control. Las líneas de datos provenientes de la computadora son acopladas por medio de un amplificador bidireccional, el 74LS245, el cual es accionado por dos líneas de control: selección de chip, y lectura de puerto. Las líneas de control provienen del bus de la computadora y entran directamente, excepto la de selección de dispositivo CS. Esta se genera a través de un decodificador de direcciones, el 74LS138.

1.2 Salidas del Controlador de Puertos Paralelos

El controlador 8255 tiene tres puertos de ocho líneas cada uno, llamados A, B, y C. Es aquí donde se concentra la recepción y transferencia de datos. Al puerto A se le conectaron en forma paralela cuatro dispositivos 74LS 373, para transferir o escribir el dato en uno de ellos. A la salida de estos amplificadores se conectan los cuatro convertidores D/A, los DAC0808, los cuales a su vez se conectan a amplificadores analógicos integrados en un dispositivo, el LM348. Estos amplificadores acondicionan la señal analógica a un nivel de voltaje de 0 a 5 voltios @ 25 mA.

El puerto B tiene conectadas las líneas de salida de datos del convertidor A/D, para leer o recibir la información.

Por último, el puerto C, usado para controlar dispositivos, tiene conectadas cinco líneas al convertidor A/D y dos líneas al PAL16L8. De las cinco líneas que van al convertidor A/D, dos de ellas son el control de comienzo y final de conversión, y las otras tres líneas seleccionan los ocho canales de entrada de las señales analógicas. Las dos líneas que van conectadas a las entradas del PAL16L8 sirven para seleccionar los cuatro dispositivos 74LS373.

1.3 Operación

1.3.1 Controlador de Puertos Paralelos

En el modo de operación 0, estos tres puertos se configuran por programa, en dieciséis formas posibles para transferir o recibir los datos. En este caso, se usará el comando 82H para decirle al dispositivo, que los datos saldrán por los puertos A y C, y que por el puerto B los datos serán recibidos; con el dato 8BH los datos serán recibidos en los puertos B y C, y por el puerto A serán enviados (ver programa de prueba).

1.3.2 El Convertidor Analógico Digital

La frecuencia de reloj máxima de operación del ADC es de 640 KHz, dando un período para los pulsos de 1/640 KHz = 1.56 µseg. Este tiempo se obtuvo por circuitería a partir de los 8 MHz de la computadora, con un divisor de frecuencia por 16, (500 KHz), que es aproximadamente la frecuencia de trabajo del ADC0809.

Se observa en la **figura 2** que la configuración del canal analógico a emplear es primero. El ADC cuenta con 8 canales analógicos seleccionados por tres líneas de entrada A0, A1 y A2. Para seleccionar el canal a emplear se envía, por medio del puerto C, alguno de los datos siguientes:

dato = 0, canal analógico 1.
dato = 2, canal analógico 2.
dato = 4, canal analógico 3.
dato = 6, canal analógico 4.
dato = 8, canal analógico 5.
dato = a, canal analógico 6.
dato = c, canal analógico 7.
dato = e, canal analógico 8.

Una vez hecho esto, se configu-

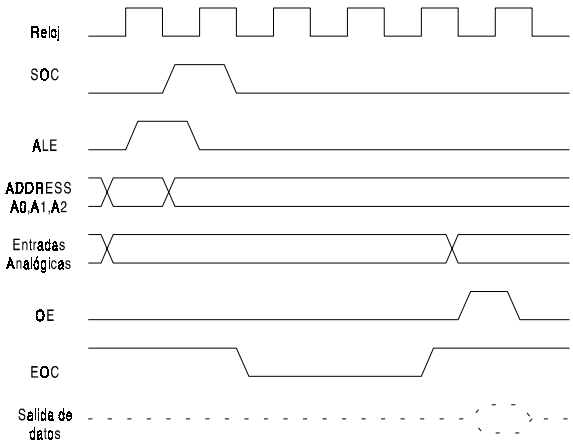


Figura 2 Diagrama de tiempos del ADC0809

ran las líneas de conversión de datos. El convertidor analógico digital ADC0809 tiene dos líneas importantes: comienzo de conversión (SOC), y fin de conversión (EOC). En este caso, la señal SOC se conecta a ALE debido a que casi comienzan al mismo tiempo.Por la misma razón, EOC se conecta a la señal de habilitación de salida OE.

La subida y bajada de la señal SOC tienen aproximadamente la misma duración (ver **figura 2)**. La señal SOC se genera por programa y no necesita retardos de tiempo, porque la instrucción de lenguaje C utilizada tiene un retardo de ejecución de este orden de tiempo.

El tiempo de conversión ocurre cuando la señal SOC baja y la EOC sube. Este tiempo es de 100 µseg., lo cual implica tener diez muestras por cada milisegundo.

1.3.3 Los Convertidores Analógico - Digital

La tarjeta cuenta con 4 canales de conversión de datos de digital a analógico, como se puede observar en la **figura 1**. El dato digital se envía a uno de los 4 registros octales 74LS373, cuando una de las líneas de habilitación (G) provenientes del PAL16L8 se activa en alto. Las entradas de este PAL provienen de dos líneas del puerto C, y se activan con los datos siguientes: 00H para el DAC1, 20H para el DAC2, 40H para el DAC3, y 60H para el DAC4.

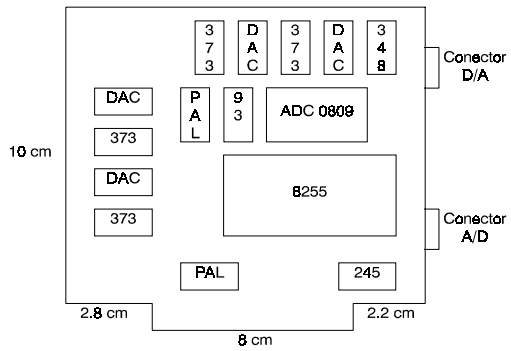


Figura 3 Diagrama de dimensionamiento y ubicación de dispositivos de la tarjeta

2. Diseño y construcción de la tarjeta.

La tarjeta tiene como conector de entrada y salida de señales digitales (datos, direcciones y control) un peine de 62 dientes, que se inserta en un conector de bus tipo PC. Como salida y entrada de señales analógicas tiene dos conectores tipo DB9, los cuales comunican al proceso con la tarjeta, como se ve en la **figura 3**.

El material empleado para el circuito impreso es una pieza de fibra de vidrio de tamaño de 13 x 10 cm, con ambas caras cobrizadas. En el lado de componentes se conectan 15 C. I. de tecnología CMOS y TTL. La tarjeta tiene 380 orificios que conectan las dos superficies: componentes y soldadura. Para evitar el "ruido", cada uno de los componentes tiene un condensador de 0.1 µf en la terminal de alimentación y tierra. La tarjeta se alimenta con tres voltajes diferentes: +5V, +12V y -12V, provenientes del bus PC. El diseño del circuito impreso, el ensamblado y las pruebas de factor calidad fueron realizados en el CINTEC, solamente la construcción del circuito impreso se realizó de forma externa.

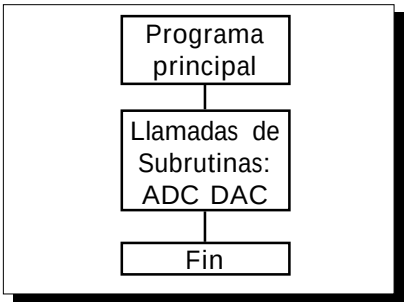


Diagrama de flujo: prueba de funcionamiento de la tarjeta

```
/* ADCDAC.C Programa de prueba para la tarjeta convertidora de
8 canales de entrada analógica a digital por 4 canales de salida digital
a analógica.

 Prof. Investigadores:
 M. en C. Romeo Urbieta Parrazales
 Ing. Jesús González Aguilar.
 Ing. Teodoro Alvarez Sánchez.
 Ing. Ignacio Minjares Tarazena
 Depto: Electrónica
 Area: Automatización de procesos por computadora.
 Escuela: Centro de Investigación Tecnológica en Computación
 Dirección: Av. Té # 950
      Col. Granjas México
      Del. Iztacalco
      Tel. 657-7453, 654-3932, 649-5036
 */
#include <stdio.h>#include <dos.h>
/* Constantes de direccionamiento del chip 8255 */
#define PA    0x100     /* Dirección Puerto A    */
#define PB    0x101     /* Dirección Puerto B    */
#define PC    0x102     /* Dirección Puerto C    */
#define PCONTROL 0x103     /* Dirección control 8255  */

/* Constantes de entrada/salida de datos del chip 8255 */
#define PALABRA1  0x82      /* A = sal b = ent c = sal */
#define PALABRA2  0x8b      /* a = sal b = ent c = ent */

/* Constantes globales */
#define BAJO    0 #define ALTO    55000 #define RETAR 0
void adc(void); void dac(int);
main()
 {
  adc();
 }
void adc()
 {
   int dato, voltaje;
   double i;



 for (i = BAJO; i <= ALTO; i++)
  {
   outportb(PCONTROL,PALABRA1); /* pc = sal          */
   outportb(PC,0x02);      /* Seleccionó canal 2 analógico  */
                /* Dato = 00, canal analógico 1  */
                /* Dato = 02, canal analógico 2  */
                /* Dato = 04, canal analógico 3  */
                /* Dato = 06, canal analógico 4  */
                /* Dato = 08, canal analógico 5  */
                /* Dato = 0a, canal analógico 6  */
                /* Dato = 0c, canal analógico 7  */
                /* Dato = 0e, canal analógico 8  */

   outportb(PC,0x12);      /* Subida de SOC y sel. canal   */
                /* Analógico de entrada      */
   outportb(PC,0x00);      /* Bajada pulso SOC        */
   outportb(PCONTROL,PALABRA2); /* PC = ent.          */

      do
       {
       dato = inportb(PC); /* Lee fin de conversión     */
       dato &= 0x01;
       }
      while(dato != 0x01);  /* Ya terminó conversión     */

      voltaje = inportb(PB); /* Captura muestra        */

       dac(voltaje);     /* Acciona dac          */
  }
outportb(PA,0x00);         /* Reset             */
}
void dac(int r)
{
   outportb(PCONTROL,PALABRA1);    /* Pto a y c = sal    */

   outportb(PA,r);
   delay(RETAR);
   outportb(PC,0x60);        /* Acciono DAC4      */

                /* dato = 0x00 DAC1    */
                /* dato = 0x20 DAC2    */
                /* dato = 0x40 DAC3    */
                /* dato = 0x60 DAC4    */
}
```

3. Programa de prueba.

Para probar el funcionamiento de la tarjeta se diseñó un pequeño programa en lenguaje C, con dos subrutinas: ADC y DAC. La subrutina ADC lee señales analógicas de 0 a 5 Voltios @ 20 mA., y las convierte a digital. La otra subrutina DAC toma el dato digital capturado y lo traduce analógico, otra vez, con el mismo nivel de voltaje.

Conclusiones

La tarjeta captura señales analógicas de 0 a 5V de amplitud en los 8 canales de entrada de conversión analógica digital, en un rango de frecuencias de 20 Hz a 300 KHz, perfectamente. Se pudo manipular voltajes de 20 mV a 4.98 V en la salida de los DAC's. El diseño de la tarjeta tiene como salidas 4 canales digitales y como entradas 8 canales analógicos, los cuales se dedican a interfazar sensores y actuadores de sistemas de control digitales directo, para procesos tanto industriales como de laboratorio.

La operación de la tarjeta puede realizarse a partir de cualquier lenguaje de programación. Tiene como direcciones la 100Hex para manipular o controlar el proceso y la 102Hex para leer las señales de entrada. Con esta tarjeta se puede realizar cualquier lazo de control digital directo, usando la computadora anfitriona para ejecutar los algoritmos de control.

Definiciones

A/D
Conversión de señal, analógica a digital.

ALE
Habilitador o indicador de dirección válida.

CMOS
Tecnología de semiconductores de óxidos metálicos complementarios.

CS
Selector o habilitador de dispositivo.

D/A
Conversión de señal, digital a analógica.

EOC
Señal de fin de conversión.

LSB
Bit menos significativo.

IORD
Señal de lectura de puerto.

IOWR
Señal de escritura en puerto.

OE
Habilitación de salida disponible.

PAL
Arreglo lógico programable.

PC
Computadora personal basado en el diseño IBM-PC.

RESET
Restablecedor.

SOC
Señal de comienzo de conversión.

TTL
Tecnología de lógica de transistor - transistor.

Bibliografía

[1] Data Adquisition Linear Devices Databook. National Semiconductor. 1990.

[2] Peripherals. Intel. 1990.

[3] Circuit Interfaces. Texas Instrument. 1990.

[4] Programmable Logic. Texas Instrument. 1990

[5] Ramakant and Sokolof. *"Digital Control"*. Ed. Prentice Hall. 1989.

OBSERVACIONES

Si te interesa contribuir con algún comentario o sugerencia, el Consejo Editorial de la Revista Polibits agradecerá utilices este espacio.

Y lo remitas a :

Centro de Investigación Tecnológica en Computación
Consejo Editorial Revista POLIBITS
Av. Té No. 950, Edificio de Graduados 2o. piso.
Colonia Granjas México. C.P. 08400
México D. F.
☎ 6-54-39-32, 6-49-50-36, 6-49-03-66 ext. 327. Fax : 6-57-74-53.

Si tu contribución es en forma de un artículo, favor de anexar los siguientes datos:
Nombre del autor, Grado Académico e Institución de procedencia.