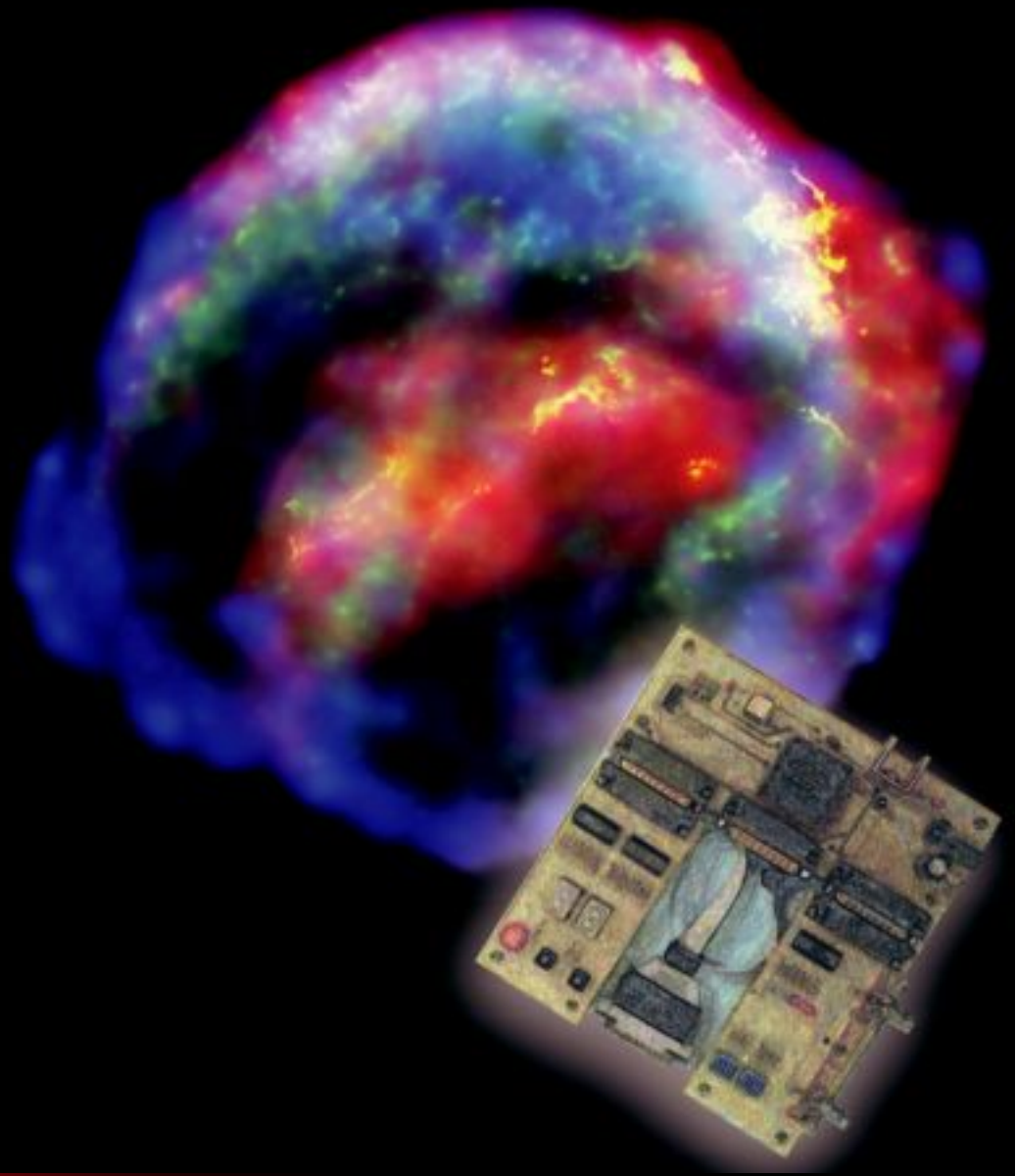
Educación técnico-profesional

Entrenador en lógica programada



λ Θ α Ψ δ Σ σ Ω φ

Serie: Recursos didácticos

Serie: Recursos didácticos

Tapa:
Imagen combinada de la Supernova Remnamt captada
por el telescopio Hubble - NASA.

autoridades

Presidente de la nación
Dr. Néstor Kirchner

Ministro de Educación, Ciencia y Tecnología
Lic. Daniel Filmus

Secretario de Educación, Ciencia y Tecnología
Prof. Alberto E. Sileoni

Directora Ejecutiva del Instituto Nacional de Educación Tecnológica
Lic. María Rosa Almandoz

Director Nacional del Centro Nacional de Educación Tecnológica
Lic. Juan Manuel Kirschenbaum

Entrenador en lógica programada

Sergio Noriega

Colección Serie “Recursos didácticos”.
Coordinadora general: Haydeé Noceti.

Industria Argentina.

ISBN 950-00-0520-4

Noriega, Sergio
Entrenador en lógica programada / Sergio Noriega ; coordinado por Juan Manuel Kirschenbaum.
- 1a ed. - Buenos Aires : Ministerio de Educación, Ciencia y Tecnología de la Nación. Instituto Nacional de Educación Tecnológica, 2005.
212 p.; 22x17 cm. (Recursos Didácticos; 12)

ISBN 950-00-0520-4

1. Electrónica-Circuitos. 2. Lógica Programada.
I. Kirschenbaum, Juan Manuel, coord.
II. Título

CDD 621.381 5

Fecha de catalogación: 3/11/2005

Impreso en Gráfica Pinter S. A., México 1352 (C1097ABB), Buenos Aires, en noviembre 2005

Tirada de esta edición: 3.000 ejemplares

Instituto Nacional de Educación Tecnológica
Centro Nacional de Educación Tecnológica
CeNET-Materiales

Serie: "**Recursos didácticos**"

1 Invernadero automatizado
2 Probador de inyectores y motores paso a paso
3 Quemador de biomasa
4 Intercomunicador por fibra óptica
5 Transmisor de datos bidireccional por fibre óptica, entre computadoras
6 Planta potabilizadora
7 Medidor de distancia y de velocidad por ultrasonido
8 Estufa de laboratorio
9 Equipamiento EMA -Características físicas de los materiales de construcción-
10 Dispositivo para evaluar parámetros de líneas
11 Biodigestor
12 Entrenador en lógica programada
13 Entorno de desarrollo para programación de microcontroladores PIC
14 Relevador de las características de componenetes semiconductores
15 Instalación sanitaria de una vivienda
16 Equipamiento para el análisis de estructuras de edificios
17 Cargador semiautomático para máquinas a CNC de accionamiento electroneumático
18 Biorreactor para la producción de alimentos
19 Ascensor
20 Pila de combustible

Ministerio de Educación, Ciencia y Tecnología.
Instituto Nacional de Educación Tecnológica.
Saavedra 789. C1229ACE.
Ciudad Autónoma de Buenos Aires.
República Argentina.

Las metas, los programas y las líneas de acción del Instituto Nacional de Educación Tecnológica

El Instituto Nacional de Educación Tecnológica -INET- enmarca sus líneas de acción, programas y proyectos, en las metas de:

- Coordinar y promover programas nacionales y federales orientados a fortalecer la educación técnico-profesional, articulados con los distintos niveles y ciclos del sistema educativo nacional.

- Implementar estrategias y acciones de cooperación entre distintas entidades, instituciones y organismos –gubernamentales y no gubernamentales-, que permitan el consenso en torno a las políticas, los lineamientos y el desarrollo de las ofertas educativas, cuyos resultados sean considerados en el Consejo Nacional de Educación-Trabajo –CoNE-T– y en el Consejo Federal de Cultura y Educación.

- Desarrollar estrategias y acciones destinadas a vincular y a articular las áreas de educación técnico-profesional con los sectores del trabajo y la producción, a escala local, regional e interregional.

- Diseñar y ejecutar un plan de asistencia técnica a las jurisdicciones en los aspectos institucionales, pedagógicos, organizativos y de gestión, relativos a la educación técnico-profesional, en el marco de los acuerdos y resoluciones establecidos por el Consejo Federal de Cultura y Educación.

- Diseñar y desarrollar un plan anual de capacitación, con modalidades presenciales, semipresenciales y a distancia, con sede en el Centro Nacional de Educación Tecnológica, y con nodos en los Centros Regionales de Educación Tecnológica y las Unidades de Cultura Tecnológica.

- Coordinar y promover programas de asistencia económica e incentivos fiscales destinados a la actualización y el desarrollo de la educación técnico-profesional; en particular, ejecutar las acciones relativas a la adjudicación y el control de la asignación del Crédito Fiscal –Ley Nº 22.317–.

- Desarrollar mecanismos de cooperación internacional y acciones relativas a diferentes procesos de integración educativa; en particular, los relacionados con los países del MERCOSUR, en lo referente a la educación técnico-profesional.

Estas metas se despliegan en distintos programas y líneas de acción de responsabilidad de nuestra institución, para el período 2003-2007:

Programa 1. Formación técnica, media y superior no universitaria:

1.1. Homologación y validez nacional de títulos.

1.2. Registro nacional de instituciones de formación técnica.

1.3. Espacios de concertación.

1.4. Perfiles profesionales y ofertas formativas.

1.5. Fortalecimiento de la gestión institucional; equipamiento de talleres y laboratorios.

1.6. Prácticas productivas profesionalizantes: Aprender emprendiendo.

Programa 2. Crédito fiscal:

2.1. Difusión y asistencia técnica.

2.2. Aplicación del régimen.

2.3. Evaluación y auditoría.

Programa 3. Formación profesional para el desarrollo local:

3.1. Articulación con las provincias.

3.2. Diseño curricular e institucional.

3.3. Información, evaluación y certificación.

Programa 4. Educación para el trabajo y la integración social.

Programa 5. Mejoramiento de la enseñanza y del aprendizaje de la Tecnología y de la Ciencia:

5.1. Formación continua.

5.2. Desarrollo de recursos didácticos.

Programa 6. Desarrollo de sistemas de información y comunicaciones:

6.1. Desarrollo de sistemas y redes.

6.2. Interactividad de centros.

Programa 7. Secretaría ejecutiva del Consejo Nacional de Educación Trabajo –CoNE-T–.

Programa 8. Cooperación internacional.

Los materiales de capacitación que, en esta ocasión, estamos acercando a la comunidad educativa a través de la serie "Recursos didácticos", se enmarcan en el Programa 5 del INET, focalizado en el mejoramiento de la enseñanza y del aprendizaje de la Tecnología y de la Ciencia, uno de cuyos propósitos es el de:

- Desarrollar materiales de capacitación destinados, por una parte, a la actualización de los docentes de la educación técnico-profesional, en lo que hace a conocimientos tecnológicos y científicos; y, por otra, a la integración de los recursos didácticos generados a través de ellos, en las aulas y talleres, como equipamiento de apoyo para los procesos de enseñanza y de aprendizaje en el área técnica.

Estos materiales didácticos han sido elaborados por especialistas del Centro Nacional de Educación Tecnológica del INET y por especialistas convocados a través del Programa de las Naciones Unidas para el Desarrollo –PNUD– desde su línea "Conocimientos científico-tecnológicos para el desarrollo de equipos e instrumentos", a quienes esta Dirección expresa su profundo reconocimiento por la tarea encarada.

María Rosa Almandoz
Directora Ejecutiva del Instituto Nacional de Educación Tecnológica.
Ministerio de Educación, Ciencia y Tecnología

Las acciones del Centro Nacional de Educación Tecnológica

Desde el Centro Nacional de Educación Tecnológica –CeNET– encaramos el diseño, el desarrollo y la implementación de proyectos innovadores para la enseñanza y el aprendizaje en educación técnico-profesional.

El CeNET, así:

- Es un ámbito de desarrollo y evaluación de metodología didáctica, y de actualización de contenidos de la tecnología y de sus sustentos científicos.
- Capacita en el uso de tecnología a docentes, profesionales, técnicos, estudiantes y otras personas de la comunidad.
- Brinda asistencia técnica a autoridades educativas jurisdiccionales y a educadores.
- Articula recursos asociativos, integrando a los actores sociales involucrados con la Educación Tecnológica.

Desde el CeNET venimos trabajando en distintas líneas de acción que convergen en el objetivo de reunir a profesores, a especialistas en Educación Tecnológica y a representantes de la industria y de la empresa, en acciones compartidas que permitan que la educación técnico-profesional se desarrolle en la escuela de un modo sistemático, enriquecedor, profundo... auténticamente formativo, tanto para los alumnos como para los docentes.

Una de nuestras líneas de acción es la de diseñar y llevar adelante un sistema de capacitación continua para profesores de educación técnico-profesional, implementando trayectos de actualización. En el CeNET contamos con quince unidades de gestión de aprendizaje en las que se desarrollan cursos, talleres, pasantías, conferencias, encuentros, destinados a cada educador que desee integrarse en ellos presencialmente o a distancia.

Otra de nuestras líneas de trabajo asume la responsabilidad de generar y participar en redes que vinculan al Centro con organismos e instituciones educativos ocupados en la educación técnico-profesional, y con organismos, instituciones y empresas dedicados a la tecnología en general. Entre estas redes, se encuentra la Red Huitral, que conecta a CeNET con los Centros Regionales de Educación Tecnológica -CeRET- y con las Unidades de Cultura Tecnológica –UCT– instalados en todo el país.

También nos ocupa la tarea de producir materiales de capacitación docente. Desde CeNET hemos desarrollado distintas series de publicaciones –todas ellas disponibles en el espacio web www.inet.edu.ar–:

- *Educación Tecnológica*, que abarca materiales que posibilitan una definición curricular del área de la Tecnología en el ámbito escolar y que incluye marcos teóricos generales, de referencia, acerca del área en su conjunto y de sus contenidos, enfoques, procedimientos y estrategias didácticas más generales.

- *Desarrollo de contenidos,* nuestra segunda serie de publicaciones, que nuclea fascículos de capacitación en los que se profundiza en los campos de problemas y de contenidos de las distintas áreas del conocimiento tecnológico, y que recopila, también, experiencias de capacitación docente desarrolladas en cada una de estas áreas.

- *Educación con tecnologías*, que propicia el uso de tecnologías de la información y de la comunicación como recursos didácticos, en las clases de todas las áreas y espacios curriculares.

- *Educadores en Tecnología,* serie de publicaciones que focaliza el análisis y las propuestas en uno de los constituyentes del proceso didáctico: el profesional que enseña Tecnología, ahondando en los rasgos de su formación, de sus prácticas, de sus procesos de capacitación, de su vinculación con los lineamientos curriculares y con las políticas educativas, de interactividad con sus alumnos, y con sus propios saberes y modos de hacer.

- *Documentos de la escuela técnica*, que difunde los marcos normativos y curriculares que desde el CONET –Consejo Nacional de Educación Técnica- delinearon la educación técnica de nuestro país, entre 1959 y 1995.

- *Ciencias para la Educación Tecnológica*, que presenta contenidos científicos asociados con los distintos campos de la tecnología, los que aportan marcos conceptuales que permiten explicar y fundamentar los problemas de nuestra área.

- *Recursos didácticos*, que presenta contenidos tecnológicos y científicos, estrategias –curriculares, didácticas y referidas a procedimientos de construcción– que permiten al profesor de la educación técnico-profesional desarrollar, con sus alumnos, un equipamiento específico para integrar en sus clases.

Desde esta última serie de materiales de capacitación, nos proponemos brindar herramientas que permitan a los docentes no sólo integrar y transferir sus saberes y capacidades, sino también, y fundamentalmente, acompañarlos en su búsqueda de soluciones creativas e innovadoras a las problemáticas con las que puedan enfrentarse en el proceso de enseñanza en el área técnica.

En todos los casos, se trata de propuestas de enseñanza basadas en la resolución de problemas, que integran ciencias básicas y tecnología, y que incluyen recursos didácticos apropiados para la educación técnico–profesional.

Los espacios de problemas tecnológicos, las consignas de trabajo, las estrategias de enseñanza, los contenidos involucrados y, finalmente, los recursos didácticos están planteados en la serie de publicaciones que aquí presentamos, como un testimonio de realidad que da cuenta de la potencialidad educativa del modelo de problematización en el campo de la enseñanza y del aprendizaje de la tecnología, que esperamos que resulte de utilidad para los profesores de la educación técnico-profesional de nuestro país.

Juan Manuel Kirschenbaum
Director Nacional del Centro Nacional de Educación Tecnológica.
Instituto Nacional de Educación Tecnológica

La serie "Recursos didácticos"

Desde esta serie de publicaciones del Centro Nacional de Educación Tecnológica, nos proponemos:

- Poner a consideración de los educadores un equipamiento didáctico a integrar en los procesos de enseñanza y de aprendizaje del área técnica que coordinan.
- Contribuir a la actualización de los docentes de la educación técnico-profesional, en lo que hace a conocimientos tecnológicos y científicos.

Inicialmente, hemos previsto el desarrollo de veinte publicaciones con las que intentamos abarcar diferentes contenidos de este campo curricular vastísimo que es el de la educación técnico-profesional.

En cada una de estas publicaciones es posible reconocer una estructura didáctica común:

1 **Problemas tecnológicos en el aula.** En esta primera parte del material se describen situaciones de enseñanza y de aprendizaje del campo de la educación técnico-profesional centradas en la resolución de problemas tecnológicos, y se presenta una propuesta de equipamiento didáctico, pertinente como recurso para resolver esas situaciones tecnológicas y didácticas planteadas.

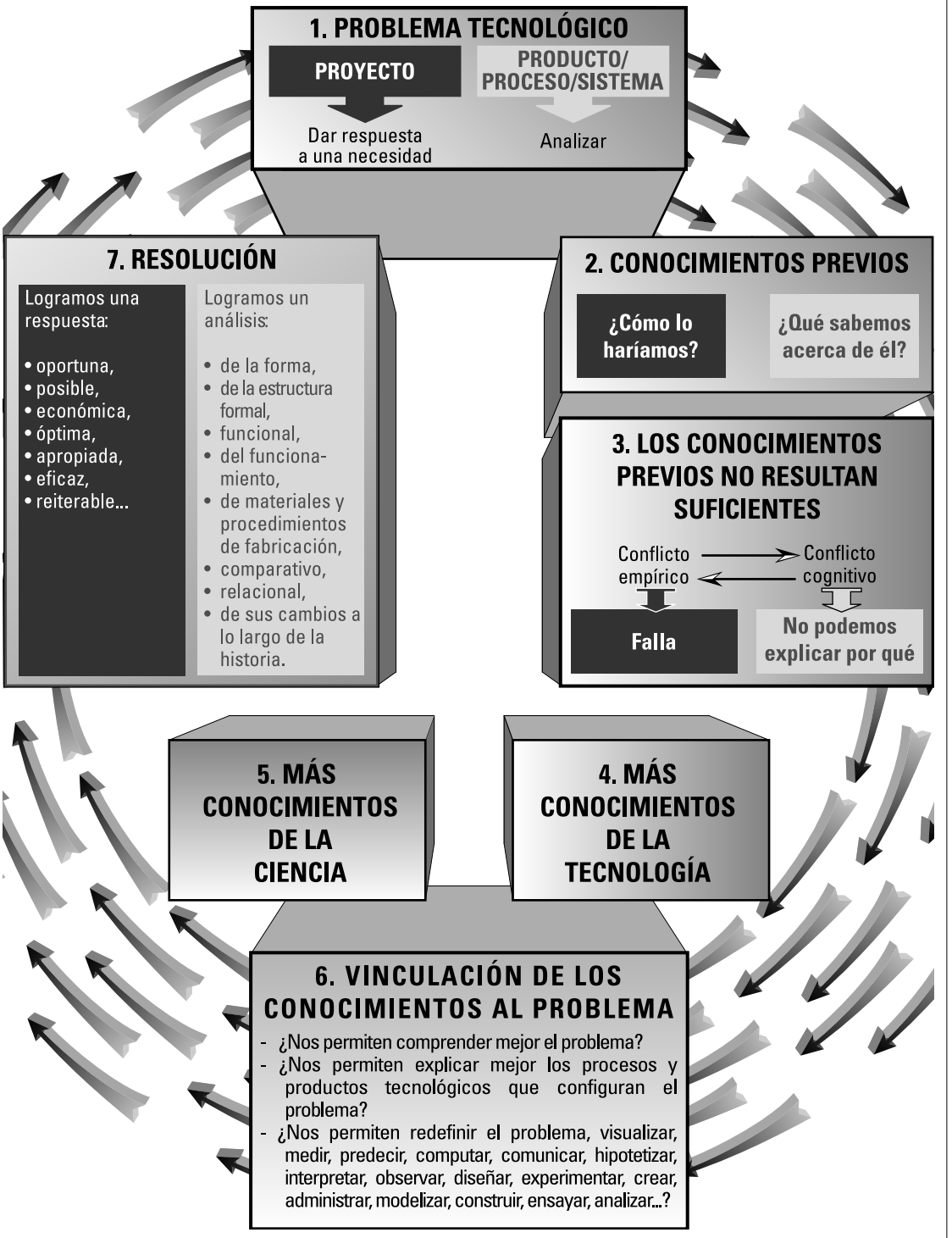
2 **Encuadre teórico para los problemas.** En vinculación con los problemas didácticos y tecnológicos que constituyen el punto de partida, se presentan conceptos tecnológicos y conceptos científicos asociados.

3 **Hacia una resolución técnica. Manual de procedimientos para la construcción y el funcionamiento del equipo.** Aquí se describe el equipo terminado y se muestra su esquema de funcionamiento; se presentan todas sus partes, y los materiales, herramientas e instrumentos necesarios para su desarrollo; asimismo, se pauta el "paso a paso" de su construcción, armado, ensayo y control.

4 **El equipo en el aula.** En esta parte del material escrito, se retoman las situaciones problemáticas iniciales, aportando sugerencias para la inclusión del recurso didáctico construido en las tareas que docente y alumnos concretan en el aula.

5 **La puesta en práctica.** Este tramo de la publicación plantea la evaluación del material didáctico y de la experiencia de puesta en práctica de las estrategias didácticas sugeridas. Implica una retroalimentación –de resolución voluntaria– de los profesores destinatarios hacia el Centro Nacional de Educación Tecnológica, así como el punto de partida para el diseño de nuevos equipos.

Esta secuencia de cuestiones y de momentos didácticos no es azarosa. Intenta replicar –en una producción escrita– las mismas instancias de trabajo que los profesores de Tecnología ponemos en práctica en nuestras clases:


1. PROBLEMA TECNOLÓGICO
PROYECTO
PRODUCTO/ PROCESO/SISTEMA
Dar respuesta a una necesidad
Analizar
2. CONOCIMIENTOS PREVIOS
¿Cómo lo haríamos?
¿Qué sabemos acerca de él?
3. LOS CONOCIMIENTOS PREVIOS NO RESULTAN SUFICIENTES
Conflicto empírico
Conflicto cognitivo
Falla
No podemos explicar por qué
4. MÁS CONOCIMIENTOS DE LA TECNOLOGÍA
5. MÁS CONOCIMIENTOS DE LA CIENCIA
6. VINCULACIÓN DE LOS CONOCIMIENTOS AL PROBLEMA
- ¿Nos permiten comprender mejor el problema?
- ¿Nos permiten explicar mejor los procesos y productos tecnológicos que configuran el problema?
- ¿Nos permiten redefinir el problema, visualizar, medir, predecir, computar, comunicar, hipotetizar, interpretar, observar, diseñar, experimentar, crear, administrar, modelizar, construir, ensayar, analizar...?
7. RESOLUCIÓN
Logramos una respuesta:
• oportuna,
• posible,
• económica,
• óptima,
• apropiada,
• eficaz,
• reiterable...
Logramos un análisis:
• de la forma,
• de la estructura formal,
• funcional,
• del funciona-miento,
• de materiales y procedimientos de fabricación,
• comparativo,
• relacional,
• de sus cambios a lo largo de la historia.

Es a través de este circuito de trabajo (problema-respuestas iniciales-inclusión teórica-respuestas más eficaces) como enseñamos y como aprenden nuestros alumnos en el área:

- La tarea comienza cuando el profesor presenta a sus alumnos una **situación codificada en la que es posible reconocer un problema tecnológico**; para configurar y resolver este problema, es necesario que el grupo ponga en marcha un proyecto tecnológico, y que encare análisis de productos o de procesos desarrollados por distintos grupos sociales para resolver algún problema análogo. Indudablemente, no se trata de cualquier problema sino de uno que ocasiona obstáculos cognitivos a los alumnos respecto de un aspecto del mundo artificial que el profesor –en su marco curricular de decisiones– ha definido como relevante.

- El proceso de enseñanza y de aprendizaje comienza con el planteamiento de esa situación tecnológica seleccionada por el profesor y con la construcción del espacio-problema por parte de los alumnos, y continúa con la búsqueda de **respuestas**.

- Esta detección y construcción de respuestas no se sustenta sólo en los conocimientos que el grupo dispone sino en la **integración de nuevos contenidos.**

- El enriquecimiento de los modos de "ver" y de encarar la resolución de un problema tecnológico –por la adquisición de nuevos conceptos y de nuevas formas técnicas de intervención en la situación desencadenante– suele estar **distribuida materialmente** –en equipamiento, en materiales, en herramientas–.

No es lo mismo contar con este equipamiento que prescindir de él.

Por esto, lo que intentamos desde nuestra serie de publicaciones es acercar al profesor distintos recursos didácticos que ayuden a sus alumnos en esta tarea de problematización y de intervención –sustentada teórica y técnicamente– en el mundo tecnológico.

Caracterizamos como **recurso didáctico** a todo material o componente informático seleccionado por un educador, quien ha evaluado en aquél posibilidades ciertas para actuar como mediador entre un problema de la realidad, un contenido a enseñar y un grupo de alumnos, facilitando procesos de comprensión, análisis, profundización, integración, síntesis, transferencia, producción o evaluación.

Al seleccionar los recursos didácticos que forman parte de nuestra serie de publicaciones, hemos considerado, en primer término, su potencialidad para posibilitar, a los alumnos de la educación técnico-profesional, configurar y resolver distintos problemas tecnológicos.

Y, en segundo término, nos preocupó que cumplieran con determinados rasgos que les permitieran constituirse en medios eficaces del conocimiento y en buenos estructurantes cognitivos, al ser incluidos en un aula por un profesor que los ha evaluado como perti-

nentes. Las cualidades que consideramos fundamentales en cada equipo que promovemos desde nuestra serie de publicaciones "Recursos didácticos", son:

- Modularidad (puede adaptarse a diversos usos).
- Resistencia (puede ser utilizado por los alumnos, sin peligro de romperse con facilidad).
- Seguridad y durabilidad (integrado por materiales no tóxicos ni peligrosos, y durables).
- Adaptabilidad (puede ser utilizado en el taller, aula o laboratorio).
- Acoplabilidad (puede ser unido o combinado con otros recursos didácticos).
- Compatibilidad (todos los componentes, bloques y sistemas permiten ser integrados entre sí).
- Facilidad de armado y desarmado (posibilita pruebas, correcciones e incorporación de nuevas funciones).
- Pertinencia (los componentes, bloques funcionales y sistemas son adecuados para el trabajo con los contenidos curriculares de la educación técnico-profesional).
- Fiabilidad (se pueden realizar las tareas preestablecidas, de la manera esperada).
- Coherencia (en todos los componentes, bloques funcionales o sistemas se siguen las mismas normas y criterios para el armado y utilización).
- Escalabilidad (es posible utilizarlo en proyectos de diferente nivel de complejidad).
- Reutilización (los diversos componentes, bloques o sistemas pueden ser desmontados para volver al estado original).
- Incrementabilidad (posibilidad de ir agregando piezas o completando el equipo en forma progresiva).

Haydeé Noceti
Coordinadora de la acción "Conocimientos científico-tecnológicos para el desarrollo de equipos e instrumentos".
Centro Nacional de Educación Tecnológica

Educación técnico-profesional

12. Entrenador en lógica programada

λ Θ α Ψ δ Σ σ Ω φ Serie: Recursos didácticos

Este material de capacitación fue desarrollado por:

Sergio Noriega.
Es Ingeniero en Telecomunicaciones. Se desempeña como Profesional de Apoyo Principal en la Comisión de Investigaciones Científicas de la provincia de Buenos Aires (CIC), con lugar de trabajo en el Laboratorio Metrológico para las Comunicaciones Ópticas (LAMECO) del Centro de Investigaciones Ópticas (CIOp). Es profesor titular en la cátedra "Introducción a los sistemas lógicos y digitales" (Facultad de Ingeniería. Universidad Nacional de La Plata) y profesor asociado en la cátedra "Telecomunicaciones I" (Facultad de Ingeniería y Ciencias Exactas. Universidad Argentina de la Empresa).

Coordinación general:
Haydeé Noceti

Diseño didáctico:
Ana Rúa

Administración:
Adriana Perrone

Monitoreo y evaluación:
Laura Irurzun

Diseño gráfico:
Tomás Ahumada
Karina Lacava
Alejandro Carlos Mertel

Diseño de tapa:
Laura Lopresti
Juan Manuel Kirschenbaum

Con la colaboración
del equipo de profesionales
del Centro Nacional
de Educación Tecnológica

Índice

λ Θ α Ψ δ Σ σ Ω φ Δ π Φ χ ∅ β Ξ ∞ ∇

1. Problemas tecnológicos en el aula

En comparación con la electrónica analógica, la electrónica digital ha ganado mucho terreno en las últimas décadas.

Hoy en día, el diseño digital no sólo tiene relevancia en la computación y en las telecomunicaciones sino que, además, brinda soluciones muy eficientes en áreas como la industria (con controladores digitales de procesos, por ejemplo) o la electromedicina (equipos de monitoreo y adquisición de datos, tomografía computada), por nombrar sólo algunas. Esta eficiencia es posible gracias al desarrollo de distintos tipos de dispositivos digitales basados en la utilización de microprocesadores y de circuitos de lógica programada.

Un **dispositivo lógico programable** –a diferencia de un microprocesador– es un conjunto de compuertas que se encuentran contenidas dentro de un chip que se programa desde una computadora. De acuerdo con la necesidad, es posible interconectar sus componentes internos a fin que de que se implemente el circuito deseado: algo sencillo como un contador o algo sofisticado como un microprocesador, ya que la complejidad del diseño depende del tipo y de las características del chip que se utilice.

Un **microprocesador** es un circuito digital complejo que funciona con el control de un programa escrito en una memoria. Sus aplicaciones abarcan desde el control de una máquina expendedora de café hasta el de un trasbordador espacial.

Resulta imprescindible, entonces, estudiar esta tecnología con los alumnos, ya que se encuentra en un sinnúmero de equipos digitales modernos.

Consideremos algunas situaciones didácticas que involucran contenidos de lógica programada.

Para sus pruebas en el "Laboratorio de autotrónica", los alumnos necesitan medir el tiempo que trascurre al accionar un actuador neumático.

El dispositivo que requieren es:

- Un reloj digital que cuente segundos desde 00 hasta 59 y que, allí, comience una nueva secuencia. La visualización podría realizarse a través de un display de dos dígitos de 7 segmentos cada uno, de tipo *led*.

A partir de esta descripción inicial, el grupo encara el desarrollo del reloj desde el "Laboratorio de electrónica". El proyecto va a terminar en un circuito impreso y funcionando.

Al considerar las posibles formas de

implementar las operaciones iniciales que permitan desarrollar el reloj, surgen distintas alternativas:

- Emplear circuitos integrados de tecnología TTL –lógica transistor-transistor– para resolver cada una de las partes que constituyen el reloj; por ejemplo, utilizar la serie 74LS de TTL.
- Emplear tecnología CMOS –transistor de efecto de campo de simetría complementaria–; por ejemplo, la serie 4000 de *National Semiconductor*®[1] o la 14000 de *Texas Instruments*®[2].
- Emplear circuitos integrados dedicados CMOS, que ya vienen preparados para cumplir la función de reloj.

Con su profesor de electrónica, los alumnos analizan que:

- La primera opción –si bien aún es viable– implicaría utilizar una tecnología que ya está en desuso; fundamentalmente, debido a que consume mucha corriente y que, al ocupar mayor espacio en el chip, permite una menor densidad de integración –implementar menos lógica en un área dada de silicio– con respecto a la tecnología CMOS, por lo que su costo resulta más alto.
- Para el uso de la tecnología CMOS, podrían emplear el MC14017B como contador; éste es un simple contador BCD –decimal codificado en binario; circuito contador de décadas– con dos chips. O, mejor aún, podrían usar el MC14518B que es un contador BCD doble, implementado en un solo chip. Como decodificador BCD a 7 segmentos del reloj, es posible emplear el MC14511B que es muy utilizado o, también, los circuitos MC14513, MC14547 u otros. Y, como fuente de reloj, acudir a, por ejemplo, un oscilador a cristal de cuarzo, empleando un cristal y compuertas inversoras.

Los alumnos optan, entonces, por la base de un circuito contador de décadas formado por dos contadores –uno para las unidades y otro para las decenas de segundos–. Además, prevén la inclusión de una fuente de reloj de un segundo de período, para aplicar a la entrada de reloj de los contadores. Para implementar la parte de visualización –como van a usar dos display de 7 segmentos a *led*–, interconectarán un decodificador BCD a 7 segmentos entre las salidas de cada contador y dichos display.

A continuación, basándose en las hojas de datos de los manuales de CMOS, el profesor propone a los estudiantes que seleccionen los componentes, considerando los criterios de:

- Simplicidad; van a decidirse por aquellos circuitos integrados que hagan el diseño lo más sintético posible –es decir, con la menor cantidad de componentes– y, preferentemente, constituido por aquellos que puedan

[1] *National Semiconductor*: http://www.national.com
[2] *Texas Instruments*: http://www.ti.com

cumplir con más de una de las funciones deseadas.

- Disponibilidad en el mercado local; para esto, uno de los rasgos requeridos es que los componentes no sean obsoletos.
- Precio accesible.

El paso siguiente consiste en considerar cómo pueden saber si el diseño que van a implementar funciona o no.

Las maneras que los estudiantes conocen son dos:

- Mediante la información suministrada por la hoja de datos de los componentes seleccionados, pueden realizar un diagrama de tiempos a través del cual sea posible constatar que los diferentes componentes trabajan en forma correcta.
- Comprobar el funcionamiento, armando los componentes electrónicos en una plaqueta de ensayos tipo *Experimentor®* o *Proto-Board*.

Seleccionados ya los circuitos a emplear, comprobada la viabilidad del diseño y habiendo realizado el diagrama circuital, la siguiente etapa que los alumnos encaran es la de la construcción. Para esto, desarrollan el circuito impreso que va a permitirles interconectar los circuitos integrados y los otros componentes asociados (resistencias, conectores, capacitores, etc.).

Un equipo de alumnos opta por realizar el circuito con una plaqueta del tipo *Plaquetodo®*[3] a la que sueldan los pines de los chips por medio de alambres de cobre; otro equipo fabrica el circuito impreso[4].

Una vez finalizado el impreso –tarea que les lleva seis horas de clase–, proceden a soldar los componentes del reloj digital y lo prueban.

Pero, su profesor ha previsto para ellos otro desafío:

– *Y, ¿si, las prácticas en el laboratorio de autotrónica hicieran necesario modificar alguna de las características del reloj? Por ejemplo, agregar los dígitos correspondientes a los minutos, de modo tal que el reloj cuente, en total, una hora –es decir, desde 00:00 hasta 59:59–?*

La respuesta general es de desaliento: Este cambio implicaría rediseñar el proyecto

[3] *Editorial Técnica Plaquetodo*: http://www.plaquetodo.com

[4] Lo hacen, empleando una placa de pertinax con dos caras recubiertas de una delgada lámina de cobre; graban el diagrama del circuito impreso con los caminos de cobre en dichas caras y eliminan el resto. Para el diseño de circuitos impresos, utilizan los programas *OrCAD®* o *Protel® –Cimmetry Systems Inc.* http://www.cimmetry.com/formats.html– que permiten imprimir y, posteriormente, con un método fotográfico, imprimir un "negativo" o un "positivo" para usar como máscara, en el paso de fabricación del impreso. Entonces, los alumnos depositan el papel traslucido positivo sobre una de las caras y la iluminan con una lámpara de rayos ultravioleta durante varios minutos; un film positivo queda traslúcido donde no debe haber cobre y negro donde sí debe haberlo; en el caso positivo, el producto químico fotosensible que contiene la placa deja la película endurecida donde la luz no pasa –por el negro del film– y débil donde la luz pasa. Como la placa es doble faz, se repite el procedimiento en la otra cara. Una vez concluida esta etapa, los alumnos sumergen la placa en ácido (generalmente, percloruro férrico) durante el tiempo suficiente como para que se elimine el cobre de la zona que recibió la luz ultravioleta y quede sólo el cobre de la parte del film que era negra.

para duplicar la cantidad de contadores, decodificadores y displays; inclusive, sería necesario volver a desarrollar del circuito impreso, ya que el que hicieron no serviría más; y, para utilizar los componentes anteriores, habría que desoldarlos –con la posibilidad de dañarlos– y volverlos a soldar.

Entonces, el profesor interviene:

– *Quiero enseñarles una metodología distinta, que evita todas estas dificultades...*

El profesor y los alumnos de "Bases de lógica digital" se encuentran analizando los diferentes tipos de circuitos que podrían utilizarse para concretar un proyecto de electrónica digital:

- Circuitos elementales, tales como compuertas *and*, *nand*, *or*, *nor*, *inversor* y aquellas derivadas de las anteriores.
- Circuitos de lógica combinatoria, tales como: sumadores, restadores, decodificadores, codificadores, multiplexores, demultiplexores, etc.
- Circuitos de lógica secuencial, tales como: *flip-flops*, *latches*, contadores, registros de desplazamiento, etc.

En todos los casos de circuitos combinatorios, los alumnos van a necesitar conocer su tabla de verdad, a fin de saber qué realiza cada uno de ellos; y, además, para la mayoría de los circuitos secuenciales, contar con los diagramas de tiempo, a fin de entender su funcionamiento.

Una forma en que los alumnos realizan estas operaciones consiste en analizar cada circuito de tecnología estándar tipo TTL o CMOS, ubicándolos de a uno por vez en una plaqueta experimental tipo *Experimentor®*, y efectuar las interconexiones necesarias con cables de cobre entre aquellos y los componentes necesarios. Empleando llaves, pulsadores y diodos emisores de luz tipo *led*, pueden visualizar las entradas y salidas, para comprobar su funcionamiento. Por otra parte, empleando un tester digital (multímetro) o un osciloscopio, les es posible realizar la función de visualizar los resultados y de analizar las señales dinámicas de circuitos secuenciales –un contador, un registro de desplazamiento–.

Pero, este procedimiento de trabajo obliga a los alumnos a modificar el experimento cada vez que cambian el circuito integrado a analizar, ya que la mayoría de los chips de lógica tradicional TTL y CMOS contiene sólo una función específica a realizar (sumador de 4 bits, contador de un solo dígito tipo BCD, etc.).

Les resultaría útil contar con un procedimiento de análisis más eficaz.

En el taller, el profesor se propone comenzar a trabajar problemáticas vinculadas con la lógica programada, con sus alumnos.

Para esto, les plantea:

– *Les acerco el diagrama circuital (esquemático) de un producto tecnológico que uste-*

des conocen –pero que no voy a anticiparles–. Por favor, examínenlo y compartan con el grupo cómo consideran que sería posible determinar su función.

Los alumnos plantean que una manera de definir la funcionalidad del circuito es a través del desarrollo de:

- ecuaciones que describen su funcionamiento,
- tablas de verdad,
- diagramas temporales.

El profesor reconoce la corrección de la respuesta; pero, plantea que, en este caso, el procedimiento puede resultar bastante tedioso e insumir mucho tiempo de análisis, con la posibilidad constante de cometer errores que conduzcan a conclusiones erróneas.

Otra forma de establecer cuál es la función del circuito es implementarlo; pero, esto llevaría mucho más tiempo y dificultades.

¿Qué hacer, en estos casos?

Los alumnos están desarrollando un contador de eventos de módulo 20 que integrarán a un equipo de llenado de frascos con comprimidos, para una industria farmacéutica.

El profesor presenta a sus alumnos el circuito digital del contador y plantea:

– Este circuito contiene errores. Mi propuesta es que ustedes piensen cómo es posible detectarlos.

En cada una de estas situaciones de enseñanza, parece óptimo integrar contenidos de lógica programada, ya que alumnos y profesores están trabajando en torno a problemas tecnológicos que, según la elección adoptada de los componentes utilizados, podrían implicar permanentes cambios en el hardware, no sólo por la cantidad de chips y modelos empleados sino por la modificación del circuito impreso.

La utilización de uno o más chips de **lógica programada** hace posible realizar todas las funciones; y, si las condiciones del proyecto así lo exigen permite, además, generar lógica, ampliar o cambiar los circuitos internos con facilidad, y replantear el esquema original.

El recurso didáctico que proponemos

El equipo didáctico **Entrenador en lógica programada** que estamos proponiéndole incluir en su clase resulta un modelo de mucha utilidad para comprender cómo es posible implementar diferentes circuitos digitales, basándose en la utilización de dispositivos configurables por hardware.

Este recurso didáctico permite avanzar en la comprensión de contenidos tales como:

- circuitos lógicos programables, nueva tecnología en el diseño digital,
- principio de funcionamiento y arquitecturas actuales para la implementación de los circuitos lógicos programables,
- diseño, simulación y programación de dispositivos lógicos programables por hardware.

El entrenador intenta, así, introducir al alumno en el mundo de la tecnología de diseño lógico por hardware, dando respuestas a los interrogantes de:

– ¿Cuáles son las ventajas de emplear circuitos programables por hardware, en lugar de usar lógica digital convencional?
– ¿Cuál es el proceso tecnológico que permite implementar físicamente hardware digital desde un chip?
– ¿Cuáles son las diferentes formas para realizar un diseño lógico, empleando este tipo de tecnología?
– ¿Qué herramientas de software y hardware se necesita para lograrlo?

El equipo está compuesto por dos placas de circuito impreso (la principal que posee el chip y una auxiliar con ejemplos de diseño) y un conector especial para programación.

La placa principal tiene alojado un circuito lógico programable del tipo CPLD –dispositivo lógico programable complejo– y está compuesta por subcircuitos:

- **Fuente de alimentación, y conectores de entradas y salidas digitales**. Permite alimentar a los circuitos internos y externos que se conecten con una tensión de 5 V regulada, proveniente de un regulador de tensión estabilizada; para esto, emplea una fuente externa de alimentación de 220 V a 12 V de corriente continua o, eventualmente, una fuente auxiliar mediante la conexión de una batería de 9 V de corriente continua para usar en caso de no contar con un suministro de energía eléctrica.
- **Circuito lógico programable complejo –CPLD– tipo EPM7128**. Este circuito integrado es el componente principal del *kit*; mediante su programación desde una computadora personal, puede ser configurado para que cumpla un gran número de funciones lógicas.
- **Oscilador a cristal de cuarzo**. Posibilita la generación de una señal de onda cuadrada, con muy alta estabilidad temporal, para aquellas experiencias en las que se requiera de una señal de reloj de precisión.
- **Conector para programación de la CPLD**. Es implementado en base a un conector tipo DB25 con un circuito integrado y componentes pasivos, a fin de brindar una interfaz entre la computadora personal (desde un puerto paralelo) y el dispositivo lógico programable (en este caso, una CPLD tipo EPM7128).

La placa auxiliar presenta ejemplos de aplicaciones que pueden implementarse en un aula[5].

A través del recurso didáctico **Entrenador en lógica programada**, los alumnos pueden realizar experiencias tales como:

- Comprender la tecnología basada en dispositivos lógicos programables por hardware.
- Realizar diseños lógicos simples y complejos con estos dispositivos y con el software asociado a ellos; para concretarlos, basta con programar al chip y conectar aquellos componentes que se requieran, desde uno o varios circuitos impresos vía los conectores DB25.

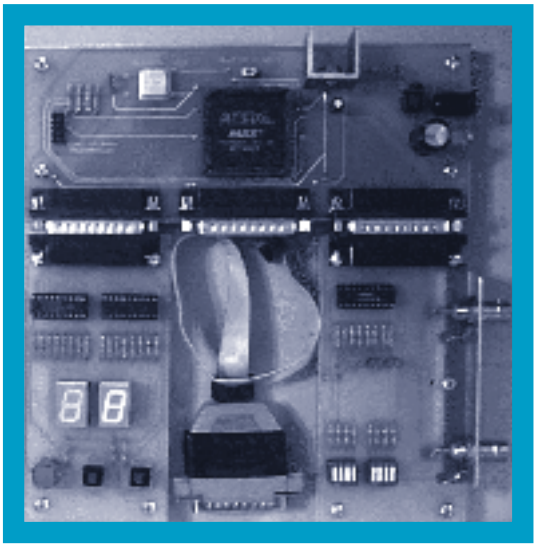
[5]Usted encontrará estas aplicaciones en el CD que acompaña esta publicación.

- Utilizar el equipo como entrenador básico de lógica digital, ya que puede emular cualquier tipo de compuerta o circuito más complejo. Por ejemplo, se podría enseñar cómo funciona un contador BCD y cómo programarlo dentro, conectando a los conectores DB25, aquellos componentes necesarios para su demostración (por ejemplo llaves, pulsadores y diodos emisores de luz).
- Simular el comportamiento de un circuito digital genérico en forma temporal, empleando el software asociado al circuito integrado. Se constituye, así, en una herramienta adicional para el análisis y síntesis de circuitos digitales.

Contando con este equipo, los alumnos del primer testimonio podrían diseñar sintéticamente el reloj y, a partir de allí, agregar o quitar funciones, a través de una simple reprogramación. También:

- realizar circuitos mucho más complejos y que trabajen a mayor velocidad que aquellos a implementar con circuitos integrados digitales en tecnología tradicional,
- ingresar los diseños de manera gráfica o mediante un lenguaje de programación especial a fin de que, luego, el programa compilador sintetice el circuito pedido,
- simular cualquier diseño digital que se realice en tecnología TTL o CMOS y, de esta manera, comprobar que no contenga errores funcionales en su diseño.

Porque, con la ayuda de los circuitos lógicos programables, es posible realizar el diseño del reloj –como tantos otros mucho más complejos–, empleando la mínima cantidad de circuitos integrados. Con un solo chip, en principio, se pueden sintetizar las funciones de la fuente de generación de reloj de un segundo de período, de los contadores y de los decodificadores BCD a 7 segmentos; y como, en general, las prestaciones de estos chips son grandes, queda todavía más lógica para implementar otras funciones.



Las ventajas de emplear este tipo de tecnología en la que el chip se puede configurar para que haga –dentro de ciertos límites– lo que el usuario requiere:

- No se necesita tener diferentes tipos de circuitos integrados en stock.
- Al implementar toda o casi toda la lógica necesaria en un chip, las velocidades de trabajo (máxima frecuencia de reloj a uti-

lizar) son muy superiores respecto de emplear lógica discreta.

En una computadora personal, por ejemplo, existen varios de estos circuitos que la hacen más veloz y que permiten disminuir su tamaño –al requerir menos cantidad de chips–.

- Seleccionando adecuadamente el circuito integrado, podemos destinar lógica adicional para ampliar el circuito, si esto es necesario.
- Es posible reconfigurar ciertos circuitos de lógica programada para hacer un nuevo diseño o para mejorar el existente. Esto constituye una ventaja extra, ya que puede suceder que –si se toman ciertas previsiones inicialmente– no sea necesario modificar el circuito impreso o, si debe hacérselo, los cambios sean mínimos, ganando en el tiempo de rediseño.
- En la etapa de diseño, la mayoría de los fabricantes de circuitos de lógica programada, ofrece un software para edición, simulación y programación de sus chips. Esto es muy importante ya que, en general, hace mucho más fácil el diseño y, además, mediante el simulador es posible saber si el proyecto funciona correctamente, sin necesidad de tener que armarlo previamente.

Si los alumnos del segundo testimonio contaran con el entrenador, podrían comprender el funcionamiento de diversos circuitos integrados simples y complejos que se enseñan en escuelas secundarias y universidades, y que forman parte de la base de circuitos mucho más complejos –memorias, conversores analógico-digitales, microprocesadores, etc.– y:

- simular estos circuitos desde el punto de vista funcional, a fin de comprobar sus tablas de verdad;
- desde el punto de vista temporal, simular cualquiera de estos circuitos realizados en tecnología TTL o CMOS, para estudiar los efectos que introducen los tiempos de retardo de un circuito físico real;
- implementar el hardware de un circuito digital simple o complejo –sumadores, decodificadores, contadores...–, con la ayuda de llaves, pulsadores, osciladores, diodos tipo LED, etc. que se implementen en diversas placas que se interconecten al entrenador. Porque, como el dispositivo empleado en el entrenador es programable, es posible seguir utilizando las mismas conexiones de pulsadores, llaves, LED, etc. para los diferentes tipos de circuitos a ensayar –éstos son creados dentro del chip, las veces que sean necesarias–.

En el taller, integrar el entrenador como recurso didáctico para que los alumnos detecten qué hace el circuito dado por su profesor, posibilita:

- realizar su simulación temporal, para estudiar el circuito por etapas si éste es muy complejo,
- implementarlo en hardware dentro del chip del entrenador y excitarlo convenientemente con señales digitales, para analizar la evolución de las salidas con el instrumental adecuado.

Y, en el último testimonio, una vez detectado el problema, el grupo de alumnos va a poder:

- Rediseñar el circuito y volver a simularlo, para comprobar que responde apropiadamente.

2. Encuadre teórico para los problemas

Hacia circuitos integrados

La electrónica se conoce desde la década de 1930.

La invención de la válvula gaseosa da origen a un nuevo componente eléctrico que, entre otras cosas, posibilita amplificar señales eléctricas. La invención del transistor en 1947, por su parte, comienza a generar un cambio respecto de cómo deberían construirse los componentes electrónicos; entonces, la tecnología sufre un redireccionamiento notable, al comenzar a trabajar sobre materiales sólidos y al diseñar un transistor -más pequeño y confiable, comparado con los tubos al vacío-.

> La ENAC - *Numerical Integrator And Computer-*, primera computadora electrónica, estaba conformada por 18.000 válvulas que se quemaban constantemente y que la hacían muy poco confiable, con un consumo de 200.000 watts de potencia y ocupando el espacio de toda una habitación; con el transistor se logra disminuir la cantidad de componentes y el área de las computadoras.

A partir de 1950, el tamaño de los dispositivos electrónicos comienza a reducirse dramáticamente. Y, desde 1958, surge la palabra **microelectrónica**. Un bloque *-chip-* de silicio de un área de 0.5 cm² pasa a contener de 10 a 20 transistores con varios diodos, resistencias y condensadores.

Así se configura la idea de **circuito integrado**, un circuito eléctrico muy avanzado formado, generalmente, por transistores, diodos, resistencias y capacitores conectados convenientemente a fin de realizar una tarea específica.

Hacia mediados de los '50 se construyen circuitos electrónicos en laboratorios industriales de dos compañías estadounidenses: *Texas Instruments*® y *Fairchild Semiconductor*®. Jack Kilby, de la primera empresa, es quien inventa el circuito integrado y, posteriormente, Robert Noyce hace mejoras que permiten resolver problemas de encapsulamiento de los chips.

En la década de 1970 ya se han desarrollado diversas familias de circuitos lógicos digi-

tales; las preponderantes son:

- TTL -lógica transistor-transistor-,
- ECL -lógica acoplada por emisor- y
- CMOS -lógica MOS (*Metal Oxide Semiconductor*; semiconductor de metal-óxido) de simetría complementaria-.

A medida que la tecnología electrónica digital sigue avanzando, se hace cada vez más compacta (introduce mayor cantidad de componentes en una misma área de silicio) y comienza a ser aplicada al diseño de dispositivos complejos tales como microprocesadores y otros dispositivos de alta densidad de integración como son las memorias de estado sólido.

Los primeros dispositivos comerciales que emplean circuitos de alta densidad de integración son las calculadoras, que dan origen a las computadoras comerciales, a comienzos del '80.

Hoy en día, se han alcanzado densidades de integración tan altas, que los circuitos integrados digitales pueden contener varias decenas de millones de transistores en un área de silicio de pocos milímetros cuadrados. Éste es el caso de los microprocesadores que se emplean en las computadoras personales como, por ejemplo, los conocidos *Pentium*® de *Intel*.

Tecnología de familias lógicas

Una **familia lógica** es una tecnología que, empleando un conjunto particular de componentes dispuestos circuitalmente de una forma dada, permite implementar físicamente funciones lógicas, según lo establecido por el álgebra de Boole.

Ya en 1970, comienza a aparecer una familia lógica que puede competir con la TTL[6]; se trata de la CMOS, basada en el uso de transistores de efecto de campo tipo MOSFET tanto de canal N como de canal P, para poder implementar cualquier función lógica primaria (and, or, negación) y, con ellas, cualquier otra función por compleja que ésta sea.

La TTL está desarrollada sobre la base de transistores bipolares del tipo NPN, con el agregado de diodos y resistencias. La CMOS, en cambio, sólo contiene transistores MOSFET en sus circuitos, con las ventajas de:

- bajo consumo sin señal,
- mayor inmunidad al ruido eléctrico,
- mayor capacidad de carga a la salida, para alimentar a otras compuertas,
- posibilidad de operar con tensiones de alimentación desde 3 V hasta 18 V.

Su principal desventaja es la de ser mucho más lenta que la TTL. Pero, con la mejora en la tecnología de fabricación de circuitos integrados y nuevas ideas para desarrollar esquemas de conexionado interno más eficientes, ambas familias van haciéndose cada vez más

[6]En este momento existe (y, aún está presente, con ciertas modificaciones) otra familia lógica denominada ECL -lógica acoplada por emisor-, basada en el uso de transistores bipolares, diodos y resistencias, mucho más veloz que TTL y CMOS, pero que emplea lógica binaria negativa y que trabaja con fuentes de alimentación negativas de -5,2 V. En la actualidad, se cuenta con otra familia lógica denominada BICMOS, que integra tanto transistores bipolares (de ahí el prefijo "bi") como de efecto de campo (CMOS) para implementar compuertas; se emplea en ciertas aplicaciones que requieren, principalmente, velocidad pero con gran capacidad de carga a la salida de dichas compuertas.

veloces.

De la primitiva TTL, se constituyen nuevas subfamilias (variaciones de la TTL con otros circuitos internos y transistores bipolares mejorados). De la inicialmente conocida serie 74 se pasa a la 74L, a la 74S y, por último, con la inclusión de transistores del tipo Schottky, se inicia la serie 74LS, 74ALS y 74F.

Por el lado de CMOS, de la serie 4000 inicial se pasa a la 74HC/HCT y, por último, a la 74AC/ACT.

En la carrera por conseguir una familia más rápida y de menor consumo, gana la CMOS frente a la TTL, ya que con la disminución del tamaño con que pueden fabricarse los transistores MOS, se consiguen los beneficios de:

- mayor velocidad de respuesta,
- menor consumo,
- mayor densidad de integración (para realizar una misma función lógica, CMOS sólo usa transistores y en menor cantidad que TTL).

Este último rasgo es decisivo, ya que permite la implementación de circuitos mucho más complejos -que con TTL- en una misma área de silicio y, además, con una velocidad un poco mayor que con la versión más rápida de la subfamilia TTL, que es la 74F.

TTL y CMOS trabajan con lógica binaria positiva y con tensiones de alimentación positivas; TTL emplea fuentes de +5 V, CMOS usa fuentes de entre +3 V y +18 V.

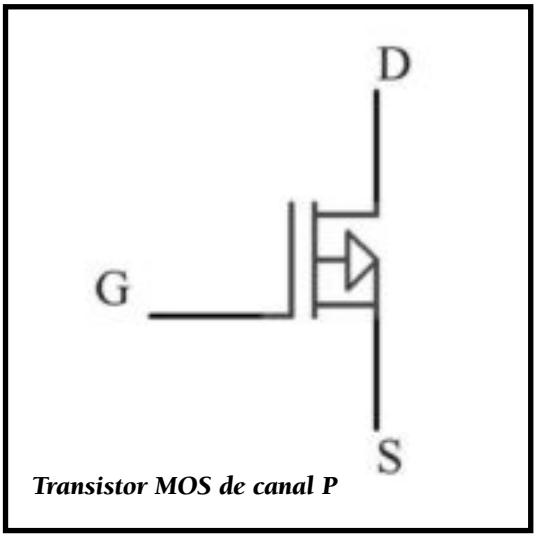
El transistor MOS, la base de toda la tecnología CMOS

Desde que se inventa la lógica CMOS hasta nuestros días, se sigue manteniendo la misma estructura para implementar, por ejemplo, un inversor.

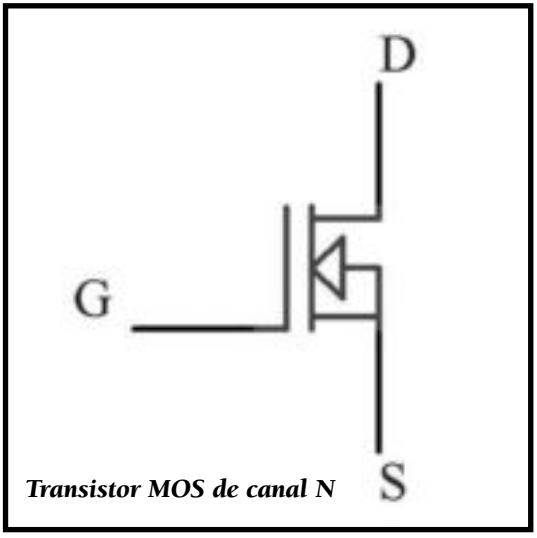
Un **transistor MOS** es un componente activo que, controlando la tensión de su compuerta -*gate*-, permite modificar la corriente que circula entre los terminales fuente -*source*- y drenaje -*drain*-. Dicho de otra manera, a diferencia de un transistor bipolar (en el que la corriente entre los terminales de emisor y colector se controla por la corriente de base), un transistor MOS permite controlar por tensión la corriente entre los terminales *drain* y *source*; de esta manera, se puede hacer que trabaje en diferentes zonas de trabajo como son las de corte, zona activa y de saturación.

Los cambios tecnológicos que han llevado a CMOS a reemplazar por completo a TTL, se centran en modificaciones realizadas en la fabricación de los transistores MOS de canal N y de canal P.

La diferencia entre un transistor de canal N y uno de canal P es el tipo de material con que se crea la zona de conducción entre el *source* y el *drain*.



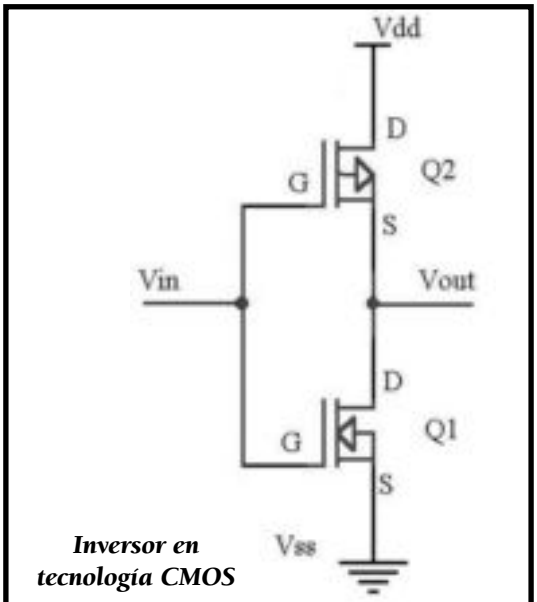
Transistor MOS de canal P



Transistor MOS de canal N

Esta diferencia, entre otras cosas, implica que es necesario cambiar la polaridad de la tensión entre *gate* y los otros terminales, para obtener los mismos resultados de trabajo con un tipo de transistor o con otro.

De esta manera, un inversor CMOS puede ser implementado de una manera muy simple, empleando dos transistores MOS.



Inversor en tecnología CMOS

- Los terminales de compuerta se unen entre sí y forman el terminal de entrada del inversor denominado, generalmente, **Vin**.
- La salida del inversor está constituida por la unión entre el *source* del transistor de canal P con el *drain* del transistor MOS de canal N, denominado **Vout**.
- El terminal de alimentación positiva está constituido por el *drain* del transistor de canal P, denominado **Vdd**.
- El terminal negativo o de masa, es el *source* del transistor MOS de canal N, denominado **Vss**.

- Alimentando el *drain* del transistor de canal p con una tensión positiva respecto del terminal *source* del transistor de canal n, se puede gobernar la salida de dicho inversor desde el terminal de entrada.
- Aplicando siempre una tensión positiva

entre Vin y Vss, del mismo valor que la de alimentación, se logra que la salida Vout vaya a un valor de tensión cercano a Vss (que equivale al estado lógico "0").

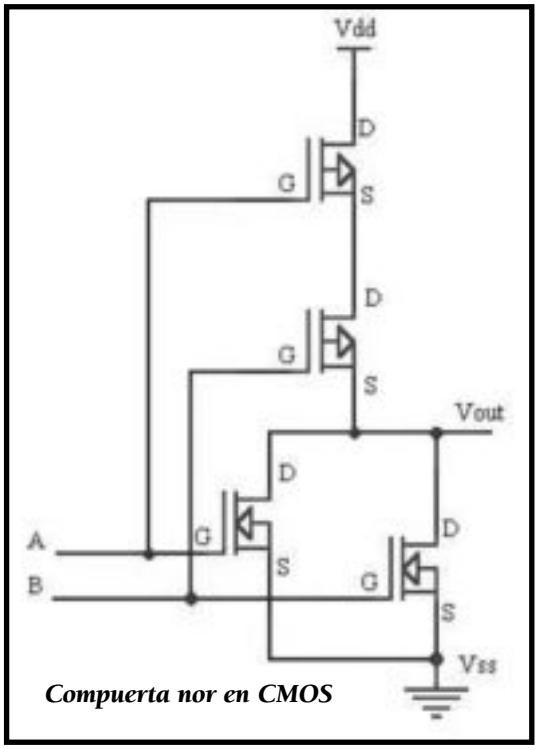
- En estas condiciones, se está aplicando al transistor MOS de canal P una tensión en el *gate* tal que lo tiende a cortar; por el contrario, dicha tensión Vin, hace conducir al transistor de canal N, saturándolo y, de esta manera, consiguiendo una tensión cercana a 0 V o Vss.
- Si se aplica, en cambio, una tensión nula a Vin (entrada cortocircuitada a Vss), la salida toma un valor cercano a Vdd (que equivale a un "1" lógico).
- Aquí, entonces, sucede lo contrario del caso anterior. La compuerta del transistor de canal N es Vss e impide que se active. El potencial aplicado a la compuerta del transistor MOS de canal P, en cambio, es más que suficiente para que conduzca y logre que la tensión Vout sea cercana a Vdd.

Para implementar otras compuertas diferentes al inversor, se aplica un esquema similar, usando transistores de canal P entre Vdd y la salida, y transistores de canal N entre la salida y Vss.

Por ejemplo, veamos cómo se implementa una **compuerta *nor*** de 2 entradas.

Este circuito consta de 2 transistores de canal P y 2 de canal N.

Los primeros están conectados en cascada entre Vdd y la salida. Los de canal N están en paralelo, conectados entre la salida y Vss.



Compuerta nor en CMOS

Cada una de las entradas está unida a un par de transistores N-P.

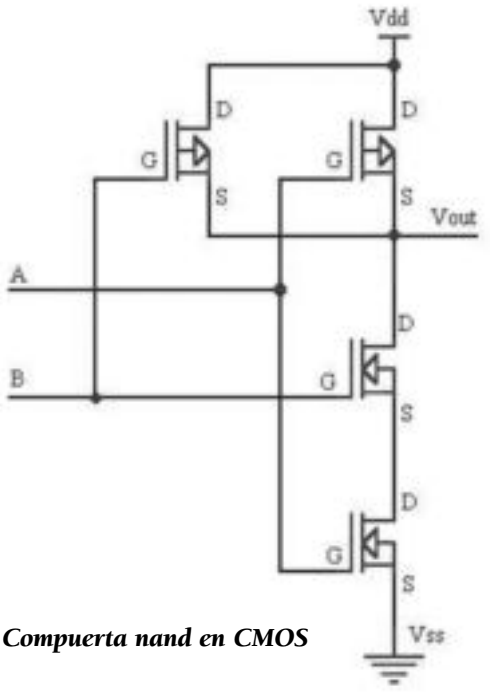
Del circuito se deduce que, para que la salida pase a Vdd ("1" lógico), se tiene que dar la condición de que ambos transistores de canal P estén activos, lo que, en principio, se logra poniendo ambas entradas a Vss.

Por otro lado, los transistores de canal N deben estar cortados, para que los transistores de canal P puedan gobernar la salida. Como están en paralelo, la única forma de lograr esto es que ambas compuertas de tipo N estén a Vss. Cualquier otra combinación de entradas (01, 10 ó 11) hace que, al menos una de las entradas, esté a Vdd, lo que implica que al menos uno de los transistores de canal P esté cortado y uno de los transistores

de canal N esté activo, con lo que la salida pasa a ser Vss ("0" lógico).

En definitiva, la tabla de verdad de este circuito es tal que sólo con las entradas a Vss se puede lograr que la salida vaya a Vdd; y esto sólo lo puede efectuar una compuerta nor.

En forma similar, una **compuerta *nand*** se puede fabricar según el siguiente circuito:



Compuerta nand en CMOS

El análisis es similar al anterior. Se puede observar que los transistores de canal P están en paralelo, mientras que los de canal N están en serie o cascada.

Aquí, la única manera de que la salida pueda estar en Vss ("0" lógico) es cuando ambos transistores de canal N están activos; y esto se logra con ambas entradas a Vdd ("1" lógico).

Cualquier otra combinación de entradas (00, 01 ó 10) hace que al menos un transistor de canal P se active y que uno de canal N esté cortado, por lo que la salida es siempre Vdd. La conclusión es que: La tabla de verdad obtenida es la de una compuerta *nand*.

¿Cómo se obtienen las compuertas *or* y *nor*? Sobre la base de las anteriores, conectando un inversor a la salida de ellas.

- Una *or* se fabrica con una *nor* seguida de un inversor.
- Una *and*, de igual manera, con una *nand* seguida de un inversor.

Basándonos en estas compuertas y en otros circuitos especiales (por ejemplo, las denominadas *pass-gate* -compuertas de transmisión-) es posible construir cualquier tipo de circuito lógico, desde un circuito combinacional tan simple como un multiplexor hasta dispositivos más complejos que nuclean lógica combinatoria y secuencial, tales como un microprocesador.

Recuerde que un microprocesador es un conjunto de circuitos basados, principalmente, en registros formados por flip-flops, y una lógica combinatoria que realiza operaciones aritméticas y lógicas denominada ALU -unidad aritmético-lógica-. Cada uno de estos circuitos está implementado por combinaciones de las compuertas básicas como las que describimos; cada una de estas compuertas está hecha, a su vez, por circuitos formados por transistores de efecto de campo de canal N y P.

Como conclusión:

Todo diseño lógico, no importa su complejidad, se basa en la interconexión apropiada de transistores de efecto de campo de canal N y P que son los que constituyen cada una de las compuertas necesarias para implementar la o las funciones lógicas requeridas.

Procesos de fabricación de circuitos integrados

Le recomendamos leer "Historia de los circuitos integrados" en **http://nobelprize.org/physics/educational/integrated_circuit/history/**

Los circuitos integrados, tanto analógicos como digitales, son fabricados con técnicas muy complejas y variadas que requieren instalaciones muy costosas, mantenidas en ambientes denominados "de sala limpia", con extrema pureza en el aire (sin ninguna partícula de polvo u otro componente que pueda depositarse en las obleas de silicio antes o después de su procesamiento). Empresas tales como *Intel*®, *AMD*®, *Texas Instrument*®, *Motorola*® y *National Semiconductors*®, por nombrar sólo algunas, disponen de tal infraestructura.

El material de referencia para comenzar a fabricar los circuitos integrados es la oblea -*wafer*-, un trozo de silicio de alta pureza, generalmente con forma circular, de más de 10 centímetros de diámetro y unos pocos milímetros de espesor.

Dado que los procesos de fabricación son muy costos, una forma de poder abaratarlos es la de fabricar varios chips en una misma oblea. Por diversos métodos de fabricación, el circuito electrónico así grabado en la oblea, se repite varias veces.

El paso siguiente consiste en cortar la oblea a fin de separar los chips y de depositar cada uno de ellos en un encapsulado que, según el caso, puede ser metálico, plástico o cerámico. Este encapsulado ya dispone de pines para, luego, poder conectar el circuito integrado a otros componentes.

El último paso consiste en soldar los contactos del chip a los respectivos pines, a fin de completar el circuito eléctrico.

Este procedimiento de fabricación de un circuito integrado tiene variaciones. Hoy en día, con la necesidad de emplear circuitos cada vez más veloces, los materiales y las técnicas van cambiando; por ejemplo, en algunas ocasiones se reemplaza el silicio por otro componente y, en otras, el chip se deposita sobre un material que hace las veces de circuito impreso, soldándose a éste con delgados alambres (circuitos denominados híbridos), por lo que no existe el encapsulado (los chips de este tipo se denominan *die*).

Si bien en la actualidad hay varias técnicas de fabricación de circuitos integrados, una de las utilizadas es la denominada *stepping* -por pasos- que permite, mediante el empleo de máscaras, ir construyendo las diferentes partes de los semiconductores (principal-

mente, transistores MOS), resistencias y capacitores, sobre una superficie (de silicio, por lo general).

Luz UV aplicada a oblea a través de máscara

Los momentos de este proceso *stepping*, son:

1. Con un software especial se hace el diseño del circuito eléctrico que va a grabarse en la oblea de silicio (Si).
2. Se fabrican las máscaras y patrones necesarios para comenzar la fabricación del chip; estas máscaras son placas que contienen el diseño del circuito integrado, con partes opacas y partes traslúcidas. Luego, sobre la oblea de silicio, son iluminadas con luz.
3. Con condiciones estrictas de calidad, se fabrica el cristal de silicio de muy alta pureza.
4. La barra de silicio que se obtiene es cortada en rodajas con una sierra de diamante. Se logran, así, varios *wafer* de silicio que son la base para comenzar a fabricar los circuitos integrados.
5. Cada uno de los *wafer* de silicio es recubierto con una capa de óxido de silicio.
6. Sobre la capa de óxido se coloca otra de un material que es sensible a la luz ultravioleta (UV).
7. La máscara que se ha fabricado se coloca entre el *wafer* y una lámpara de luz ultravioleta. Las partes traslúcidas de la máscara dejan pasar la luz UV y las opacas, no. En las partes traslúcidas donde la luz UV ilumina, el material fotosensible se degrada y, luego, puede ser removido.
8. El proceso es repetido a lo largo de la superficie del *wafer*, a fin de que se pueda aprovechar el material para obtener decenas de chips del mismo tipo. Se remueve el material fotosensible.
9. El *wafer* es tratado químicamente con ciertos productos. Este proceso se denomina *etching* y permite eliminar el material aislante (la capa de óxido de silicio) que ha quedado expuesto en las zonas donde el material fotosensible se ha eliminado.
10. El *wafer* es sometido a un tratamiento que permite cambiar las propiedades eléctricas de las zonas que han quedado expuestas, al ser eliminada la capa de óxido de Si. Este proceso se denomina *doping* -dopaje-.

Los pasos 5 al 10 pueden repetirse varias veces para construir el circuito integrado capa por capa.

11. Finalmente, cuando el chip está terminado, se agrega una capa metálica para interconectar los componentes unos con otros. Este proceso se denomina *metalization* -metalización- y es realizado de una manera similar a la anterior.
12. Sobre la capa metálica se agrega una capa de material fotosensible *-photoresist-*.
13. Repitiendo el proceso, se coloca la máscara que contiene los caminos metálicos que el chip va a recorrer y se aplica luz ultravioleta. El material *photoresist* queda degradado en las zonas donde ha pasado la luz ultravioleta a través de la máscara.
14. Con productos químicos se remueve el material fotosensible que fue alcanzado por la luz UV.
15. Se aplica otro proceso de *etching* a fin de eliminar el metal de las zonas donde ha sido removido el material fotosensible.
16. En este momento del proceso, el chip ya está internamente completo.
17. La mayoría de los chips actuales permite la aplicación de varias capas metálicas para realizar las interconexiones necesarias; es decir, el agregado de una segunda capa donde previamente se ha puesto una capa aislante.

La fabricación de chips no-programables y programables, es en general similar, salvo en las etapas de implementación de las diferentes capas metálicas que se emplean para formar la matriz de interconexión programable.

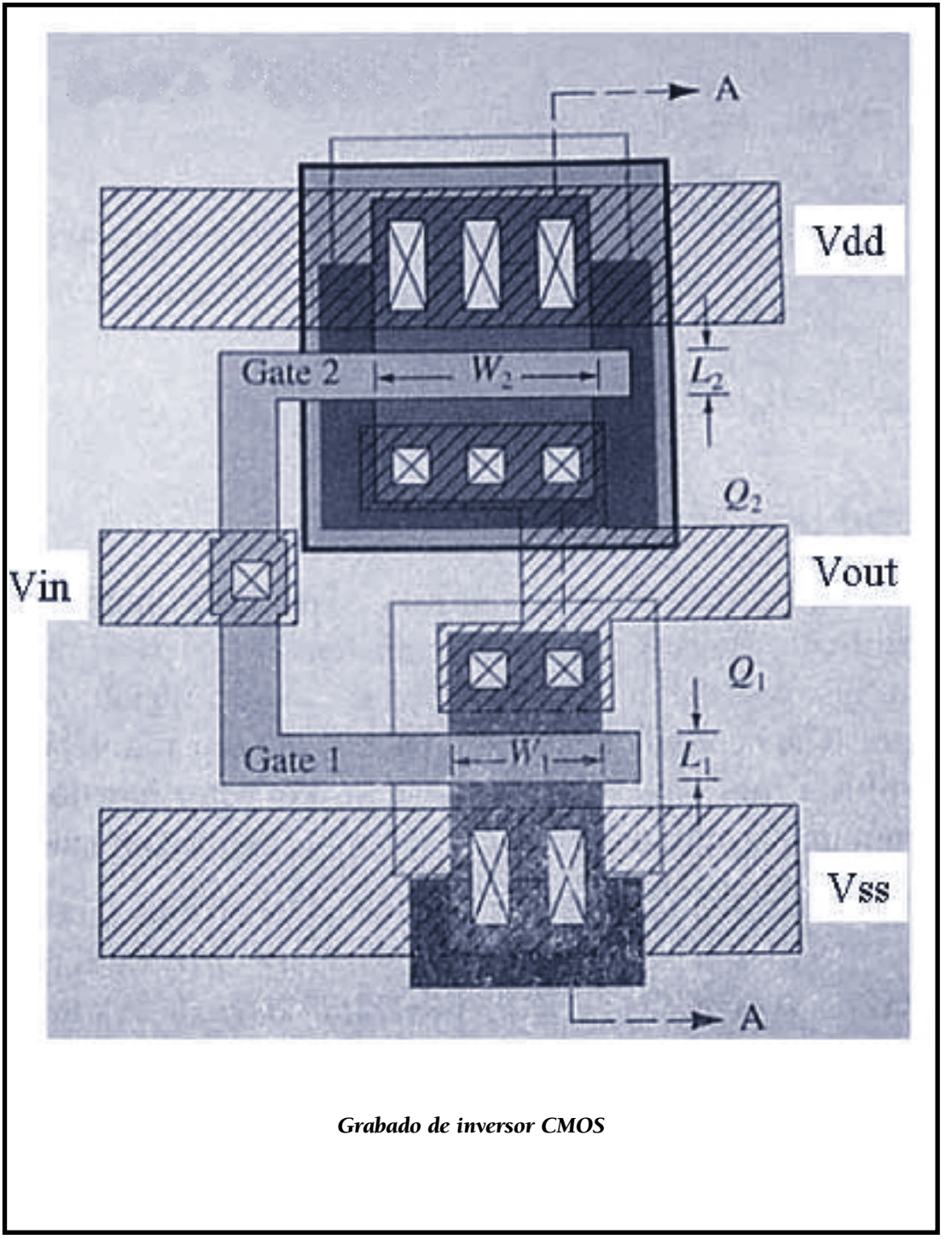
18. Aquellos chips que no son programables requieren una prueba antes de ser conectados al encapsulado. Esta prueba pretende comprobar si cada uno de los chips fabricados en la oblea de Si funciona adecuadamente, antes de pasar a la próxima etapa de construcción.
19. Los chips de la oblea son separados con una sierra de diamante.
20. Finalmente, cada chip es empaquetado (encapsulado) en una carcasa que le brinda protección e interconexión con el mundo exterior. Previamente, se suelda cada contacto del chip con el pin exterior correspondiente.

A modo de ejemplo, veamos cómo se puede fabricar un inversor CMOS:

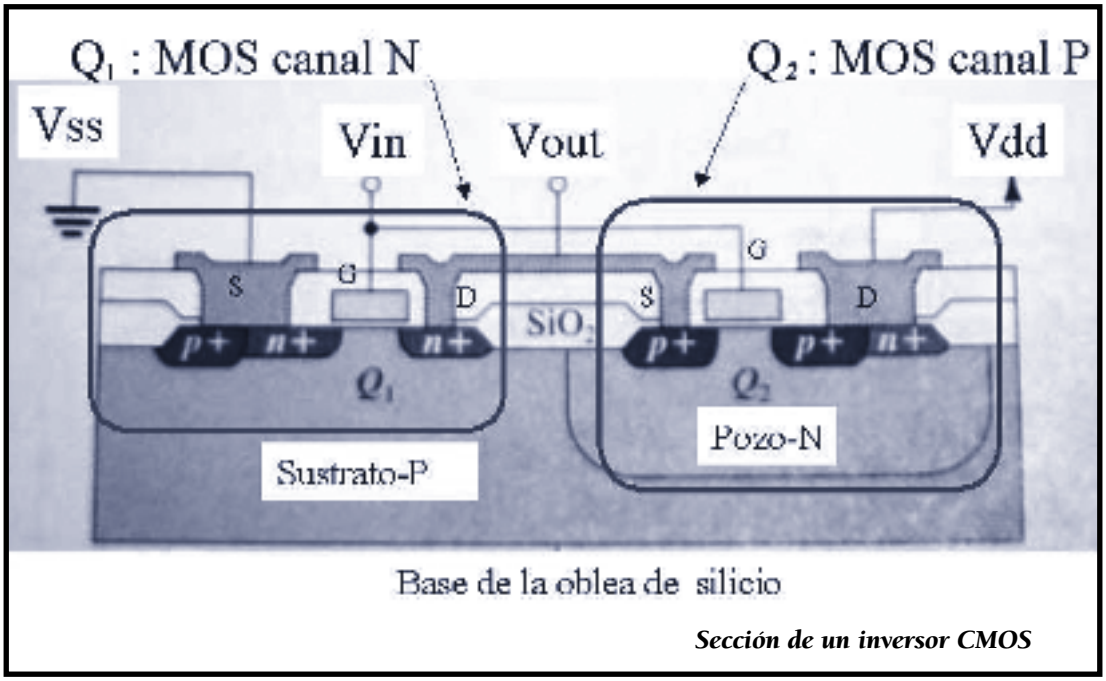
La figura de la próxima página es una vista superior de cómo se vería el inversor implementado; es el esquema final que se diseña en una computadora y da origen a las diferentes máscaras que se aplicarán oportunamente durante el proceso de fabricación del integrado.

Allí:

- Q_1 es el transistor de canal N.
- Q_2 el del canal P.
- G, D y S corresponden a *gate, drain* y *source*, respectivamente.
- A indica dónde se haría la vista del corte de la oblea.
- Sustrato-P es el material base del *wafer* de silicio.



Grabado de inversor CMOS



Sección de un inversor CMOS

Para fabricar el transistor de canal P, es necesario que el material que se emplee sea un sustrato tipo P que, en principio, una los terminales de *drain* y *source*. Para fabricar el transistor de canal P se debe emplear un sustrato-N.

Para ello se debe dopar la zona de interés para formar el denominado Pozo-N (esto se puede hacer con una máscara tal que sólo el área que se quiere cambiar de Sustrato-P a Pozo-N, pueda ser afectada al contaminarla con químicos que cambien las propiedades eléctricas del material).

Una vez hecho esto, se crean los *drain* y *source* de ambos transistores, para lo cual también hay que dopar con impurezas para formar las zonas P+ y N+, lo que también se hace con una máscara adicional.

Para formar los gate es necesario cubrir las zonas apropiadas con una capa de óxido de Si, lo que se efectúa en una etapa previa a las anteriores.

Por último, metalizan los contactos, a fin de poder interconectar el inversor con los pines del encapsulado.

Un tema muy importante es el relacionado con las **dimensiones de los transistores**. En particular con la dimensión marcada como L, que denota la longitud del canal de ambos transistores N

Este rasgo es el que permite que los microprocesadores modernos puedan alcanzar velocidades de procesamiento del orden de los GigaHertz (1 GHz equivale a 1.000 MHz o 1.000.000.000 Hz).

y P. Este parámetro es fundamental ya que, cuando menor sea, más rápido es el transistor fabricado -es decir, mayor es su frecuencia de trabajo-.

Los avances de la tecnología CMOS en la fabricación de circuitos integrados permiten lograr un tamaño cada vez menor en la construcción de los transistores.

Esto implica el logro de dos objetivos importantes:

- aumentar la velocidad de conmutación y
- permitir la ubicación de mayor número de transistores en un área de silicio, lo que redunda en mayor lógica a implementar.

Para tener una idea del tamaño logrado, digamos que los microprocesadores *Pentium*® actuales están fabricados sobre la base de transistores cuya longitud de canal L es de 130 nanometros (1 nanometro -nm- equivale a 0,000000001 metros o 0,13 micrómetros -µm-). *Intel*® está trabajando en el desarrollo de nuevos chips con longitudes de canal de los transistores de 90 nm (0,09 µm).

En 1965, un ingeniero en electrónica, Gordon Moore, observa que la tecnología avanza de tal forma que los procesos de fabricación de circuitos integrados llevan a que el número de transistores por área de silicio se duplique cada año, dato que se confirma con el correr del tiempo.

La siguiente tabla ilustra lo dicho para el caso de los modelos de microprocesadores que la empresa *Intel*® ha ido desarrollando, desde el microprocesador 386 usado en las computadoras XT hasta los modernos Pentium 4.

Año de fabricación	Nombre del microprocesador	Cantidad de transistores
1985	386	275.000
1989	486DX	1.180.000
1993	Pentium	3.100.000
1997	Pentium II	7.500.000
1999	Pentium III	24.000.000
2000	Pentium 4	42.000.000

Diseño digital. Alternativas de implementación

Un poco de historia

En los comienzos de la era de los circuitos integrados, los chips tenían una estructura rígida, de tal manera que, una vez que estaban listos para ser usados, no existía la posibilidad de modificar su funcionamiento. Un chip -ya fuera una compuerta *nand* o un microprocesador- cumplían con lo establecido en sus hojas de datos... ¡ y nada más!

Así, si un usuario deseaba algún circuito especial, no podía hacer cambios por su cuenta; debía pedírselos a las grandes empresas y éstas -por los costos que involucra diseñar un chip totalmente nuevo- cobraban una fortuna para realizar ese diseño. Incluso, para que resultara más económico contar con un chip "a medida", el usuario tenía que comprar quizás cientos de miles de ellos, solución que es aún válida para aquellas empresas que necesitan de un chip especial y en grandes cantidades (por ejemplo, empresas que arman teléfonos celulares y que requieren ciertos tipos de chips que, al ser comprados en cantidad, pueden amortizar el costo inicial).

Pero, en los '70, con la creciente mejora en la tecnología de fabricación de los circuitos electrónicos -básicamente, con los cambios relacionados con la densidad de integración: cada vez más transistores en una misma área del chip-, comienza a pensarse de una manera diferente.

En 1978 entra al mercado electrónico un tipo diferente de circuito integrado digital fabricado por la empresa *Monolitic Memories*®. Dicho circuito, denominado PAL -*Programmable Array Logic*; arreglo lógico programable-, es el precursor de los dispositivos programables que se emplean en la actualidad y permite que el usuario pueda programarlo, contando con un aparato especial que debe adquirir.

La idea de ese entonces es la de fabricar, dentro del chip, una serie de compuertas lógicas interconectadas de una forma especial a través de fusibles. Tales fusibles se queman con ese aparato especial logrando, así, implementar la función lógica deseada. Pero, una vez que un fusible es quemado, no hay posibilidad alguna de volver a la condición inicial, por lo que el chip se puede programar una sola vez.

A partir de este producto tecnológico comienza una carrera vertiginosa de diferentes fabricantes de circuitos integrados, a fin de poder afianzarse en el mercado electrónico con dispositivos cada vez más poderosos.

Los SPLD -*Simple Programmable Logical Device*; dispositivos lógicos programables simples- son los que comienzan a dominar el mercado en pequeños desarrollos de electrónica digital reemplazando, de a poco, a los circuitos integrados tradicionales TTL y CMOS.

Nuevos cambios tecnológicos, tanto en la fabricación de circuitos integrados como en la concepción de ideas para implementar circuitos programables, dan paso a los CPLD -

Complex Programmable Logic Device; dispositivos lógicos programables complejos-.

En la década de 1980 se introducen al mercado los FPGA -*Field Programmable Gate Array*; arreglo de compuertas programables por el usuario- que, a diferencia de los CPLD, tienen la capacidad de poder implementar memoria tanto RAM como ROM.

Varias ventajas interesantes son introducidas con los desarrollos de los CPLD y FPGA:

- Posibilidad de reprogramar los circuitos tantas veces como se deseara permitiendo, de esta manera, usar un mismo chip como prototipo para realizar pruebas y, luego, emplearlo como el chip definitivo de diseño[7].
- Posibilidad de incorporar lógica adicional dentro del chip, lo que permite al fabricante realizar un test de estado general. Éste constituye un factor importante en lo económico, ya que gracias a esta cualidad, los circuitos integrados programables pueden adquirirse a un costo mucho menor que si el fabricante tuviera que hacer las pruebas de sus integrados de la manera tradicional[8].
- Poner a disposición del usuario un sofisticado software para realizar el diseño, la simulación y la programación de los diferentes tipos de dispositivos programables. Dado que las estructuras internas de estos chips son tan complejas, no es posible realizar ningún diseño sino a través de estos programas compiladores que los fabricantes suministran. Existen versiones básicas que cubren gran parte de las expectativas de los usuarios a las que es posible acceder en forma gratuita; otras versiones más sofisticadas para el diseño de circuitos deben ser adquiridas.

Hoy en día las CPLD y FPGA[9] permiten diseñar no sólo circuitos digitales -como controladores de bus PCI en computadoras personales- sino dispositivos tan complejos como microprocesadores.

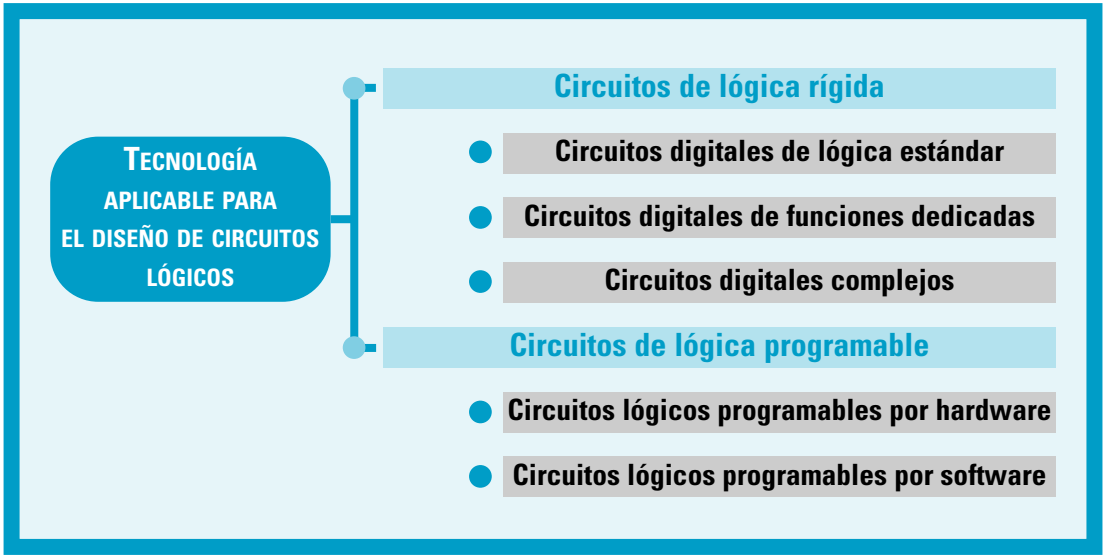
Tecnología para el diseño lógico

Podemos realizar la siguiente clasificación:

[7] En determinadas aplicaciones, como las aeroespaciales y las satelitarias, existen chips que son programados sólo una vez, ya que la tecnología empleada en esos casos requiere que los circuitos integrados sean muy resistentes a la radiación cósmica. Así, las técnicas usuales de reprogramación no resultan confiables para estos casos particulares.

[8] El test tradicional comprende el empleo de puntas de prueba que se conectan sobre los diferentes pines del circuito integrado antes de encapsularlo; luego, se inyectan y miden señales en determinados lugares, para comprobar su funcionamiento. Pero, a medida que los circuitos electrónicos implementados en los chips se hacen más grandes y con mayor número de pines, el proceso de test en fábrica se complica y, finalmente, se traduce en un aumento del costo de los circuitos. Actualmente, sacrificando algo del área del chip, en la mayoría de los circuitos integrados programables, el fabricante puede hacer su test empleando algunos pocos pines para enviar y recibir datos, e inyectar señales de reloj. Este proceso se realiza en forma serie, entrando información por un pin y recibiendo datos desde otro pin, y permite que los mismos pines de la prueba puedan ser utilizados por el usuario final.

[9] En el mercado también existen otros dispositivos, los ASIC -Application Specific Integrated Circuit; circuitos integrados para aplicaciones específicas-, un paso intermedio entre los circuitos integrados que no pueden programarse y los mencionados SPLD, CPLD y FPGA. Se trata de circuitos en los que el usuario puede diseñar lógica sobre una estructura preestablecida y enviar, luego, el modelo terminado para que el fabricante construya el chip.



Son **circuitos de lógica rígida** todos aquellos circuitos lógicos digitales fabricados sin que la función o funciones establecidas en el momento de su creación puedan ser modificadas.

- **Circuitos de lógica estándar**. Son los enmarcados dentro de las series de circuitos de lógica TTL, ECL y CMOS que permiten realizar funciones lógicas básicas tales como *and, or, nand, nor*, etc. y otras más complejas como multiplexores, contadores, etc. Generalmente, cada función lógica se encuentra implementada en un chip, por lo que deben emplearse varios de ellos para conseguir armar el diseño requerido. Estos circuitos tienen una baja capacidad de generación de lógica.
- **Circuitos de funciones dedicadas**. Son circuitos de mediana complejidad y densidad de integración que realizan ciertas funciones específicas que permiten implementar en forma parcial un diseño completo; por ejemplo, generadores de reloj a cristal, frecuencímetros, relojes, conversores analógico-digitales, memorias de estado sólido, etc.
- **Circuitos digitales complejos**. Son aquellos en los que se sintetizan varios bloques lógicos funcionales de relativa gran complejidad, como es el caso de los microprocesadores.

Los **circuitos de lógica programable** son aquellos circuitos que el usuario -a través de un software y, en algunos casos, de un programador suministrados por el fabricante- puede configurar para implementar el diseño lógico requerido.

Dentro de su programabilidad, es posible hacer una distinción de acuerdo con la naturaleza de la programación efectuada:

- **Circuitos lógicos programables por hardware**. La estructura de estos circuitos está formada por un conjunto de bloques lógicos que pueden interconectarse entre sí a fin de construir el circuito lógico

deseado. Por lo general, estas interconexiones se logran empleando transistores como llaves que pueden, por ejemplo, interconectar dos líneas o no, dependiendo de si el transistor conduce o está cortado. Estos dispositivos conforman los SPLD, CPLD y FPGA.

- **Circuitos lógicos programables por software**. La programación se realiza sobre una memoria de estado sólido. Así, este tipo de circuito -con el control del programa almacenado- puede realizar operaciones lógicas, aritméticas y transferencias de datos, según se haya establecido.

Circuitos lógicos programables por hardware

El concepto de lógica programada surge de la necesidad de desarrollar diseños digitales cada vez más complejos y más veloces que empleen la menor cantidad de circuitos integrados. Porque, el empleo de lógica estándar -por ejemplo, de tecnología CMOS- implica que, para poder diseñar un circuito digital de relativa complejidad se requiera un número apreciable de chips de diferentes funciones.

Hoy en día, los diseñadores se enfrentan a especificaciones que van variando de proyecto a proyecto e, inclusive, a algunas que no se definen en el momento del desarrollo. Por otra parte, los tiempos de desarrollo son cada vez más cortos y requieren una infraestructura que permita realizar el proceso lo más rápido posible y, además, efectuar pruebas confiables a fin de detectar y resolver errores.

Las ventajas de la lógica programada por hardware frente a la lógica estándar y a los circuitos de lógica dedicada son:

- **Diseño más veloz**. Parte o todo el proyecto está dentro del chip, con lo que se evitan los retardos que se generan en las pistas de circuitos impresos.
- **Diseño más pequeño y económico**. Son pocos los componentes empleados, es menor el área de circuito impreso, y se reduce el costo hora-hombre para diseño y armado.
- **Diseño más confiable**. El chip es probado en fábrica y su diseño es simulado por el usuario empleando software. En el diseño del impreso hay mucha menos cantidad de soldaduras a realizar y de componentes que pueden fallar.
- **Menor costo de stock**. Sólo se necesita disponer de algunos chips diferentes según el grado de complejidad del diseño.
- **Flexibilidad para nuevos diseños**. Estos diseños pueden guardarse en una computadora y ser reutilizados en otros proyectos. Empleando versiones universales de software es posible hacer un diseño que, luego, se implemente en cualquier dispositivo programable de

diferente fabricante; esto habilita a diseñar aplicaciones específicas y a ofrecerlas a usuarios.

- **Migración a nuevas tecnologías**. El conocimiento adquirido para diseñar con dispositivos lógicos programables permite que el usuario pueda adaptarse en forma relativamente simple a los constantes cambios tecnológicos que se van produciendo. El software que brinda el fabricante es versátil, lo que permite que el diseñador pueda, por ejemplo, pasar de una familia de circuitos integrados a otra sin necesidad de rehacer sus diseños.
- **Protección intelectual**. Por lo general, los circuitos de lógica programada vienen con la opción de protección contra lectura -si el diseñador no lo permite, no es posible copiar dicho chip-.

Las desventajas de emplear la tecnología de la lógica programada:

- Al disponer de lógica adicional para realizar la programación de tales dispositivos -lo que implica interconectar bloques lógicos entre sí para que cumplan con la función lógica deseada-, se está agregando más tiempo de retardo, lo que limita la máxima frecuencia de operación. En este sentido, los dispositivos ASIC tienen ventaja sobre los CPLD y FPGA, ya que no hay elementos de interconexión programables; pero, resultan caros para la mayoría de los usuarios.
- Si se pretende lograr un diseño optimizado, se requiere de cierto entrenamiento para comprender cómo usar el software asociado.

En forma generalizada, se puede decir que existen tres componentes básicos para el desarrollo de un diseño lógico:

- Microprocesador. Puede ser un microprocesador propiamente dicho o variaciones de él -por ejemplo, un microcontrolador o DSP -procesador digital de señales-.
- Circuito lógico programable. CPLD, FPGA[10] o ASIC.
- Memoria de estado sólido. Ésta, en algunos casos, puede estar implementada en un microprocesador o en un dispositivo lógico programable.

Dependiendo del proyecto, alguno de los dos primeros debe estar presente en el diseño.

¿Qué es un PLD? Clasificación

Un PLD -*Programmable Logic Device*; dispositivo lógico programable por hardware- es un circuito integrado que puede implementar variados tipos de funciones lógicas mediante una programación que, generalmente, se realiza desde una computadora personal.

Internamente, cada chip dispone de una serie de bloques lógicos que pueden interconectarse entre sí gracias a una serie de llaves y de alambres que sirven para llevar señales desde diversos puntos del chip.

[10] Con los avances de la tecnología de dispositivos programables por hardware, ya se están ofreciendo dispositivos FPGA con capacidad de emular microprocesadores; es decir, que se pueden programar internamente para que copien todos los circuitos que forman parte de dichos microprocesadores. De esta manera y considerando que las FPGA también pueden emular memoria, en un futuro cercano se contará con chips de suficiente alta densidad de integración como para poder realizar casi cualquier proyecto en un solo chip programable.

Es así como, conectando convenientemente dichas llaves, se pueden implementar combinaciones de diversos tipos de circuitos lógicos.

Cada uno de sus bloques básicos está formado por un conjunto de compuertas y un flip-flop.

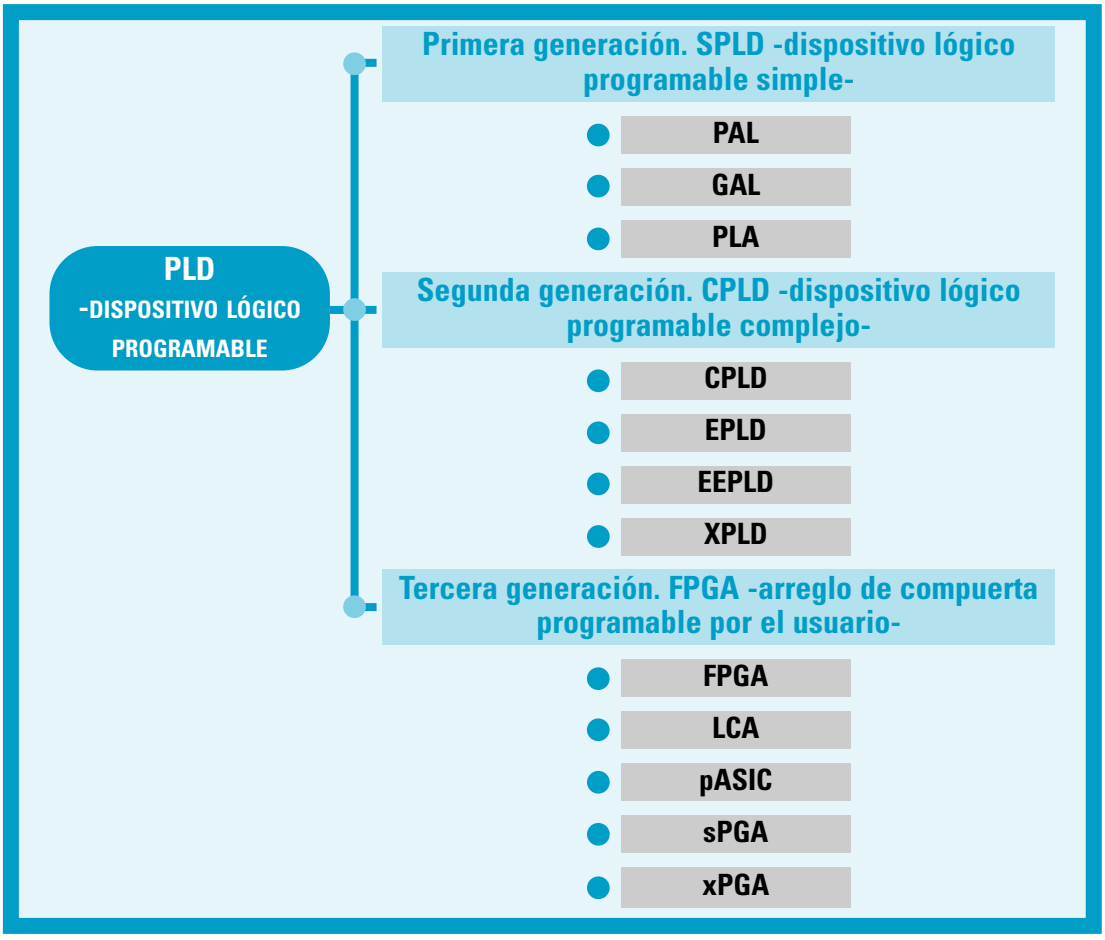
El conjunto de compuertas tiene la capacidad de realizar una o más funciones lógicas, según el tipo de dispositivo.

El flip-flop se puede emplear (también según el dispositivo de que se trate) para conectarse a la salida de dicha función o para ser usado en forma independiente.

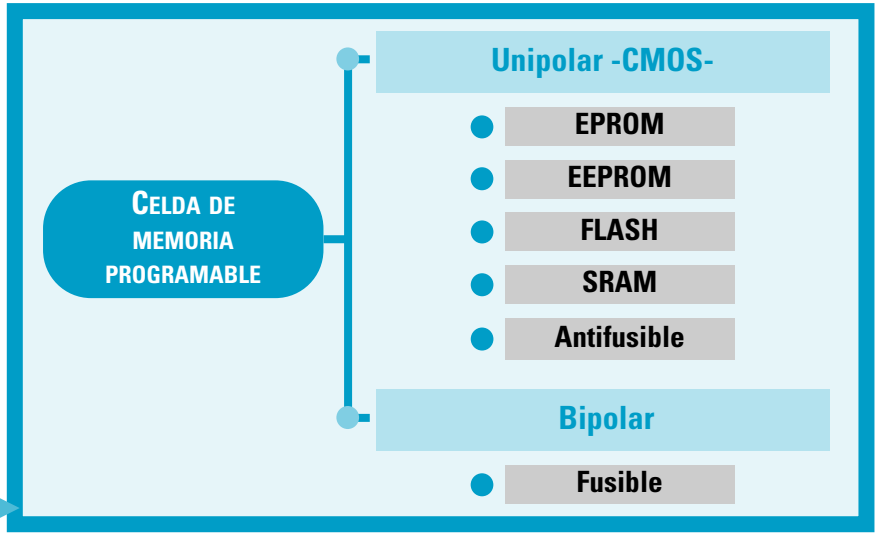
Clasificación

Desde el primer circuito lógico programable en la década de 1980, han ido apareciendo dispositivos cada vez más complejos y de mayor número de bloques lógicos.

Desde el punto de vista de su estructura interna, es posible realizar una clasificación cronológica de dispositivos PLD:



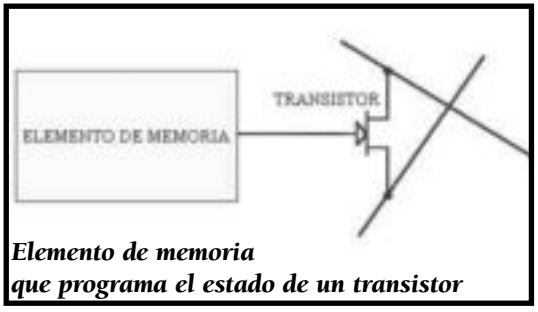
Otra posible clasificación se refiere a la tecnología empleada para programar dichas PLD: Qué tipo de elemento de memoria se emplea para poder realizar las interconexiones internas dentro de cada dispositivo, a fin de armar la lógica requerida:



Generalmente, se emplean transistores como llaves para realizar las interconexiones internas; pero, en otros casos, se emplean fusibles o antifusibles, dependiendo de si la interconexión inicialmente está cerrada o abierta, respectivamente.

A diferencia de los fusibles o antifusibles, si se usa un transistor MOS como llave, es necesario programarlo desde su compuerta a fin de definir su estado de conducción.

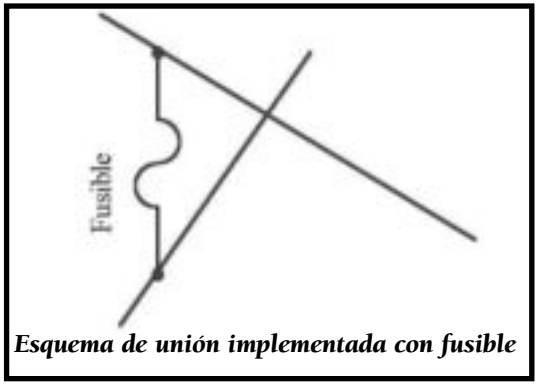
Para ello se emplean elementos de memoria de una sola salida que definen este nivel lógico. Estos elementos pueden ser celdas RAM o transistores EPROM, EEPROM o FLASH, como veremos más adelante.



Elemento de memoria que programa el estado de un transistor

El concepto de **fusible** es simple:

- Si se lo deja en el circuito, deja intacta la unión entre líneas eléctricas.
- Si se lo quema, dichas líneas quedan aisladas permanentemente.

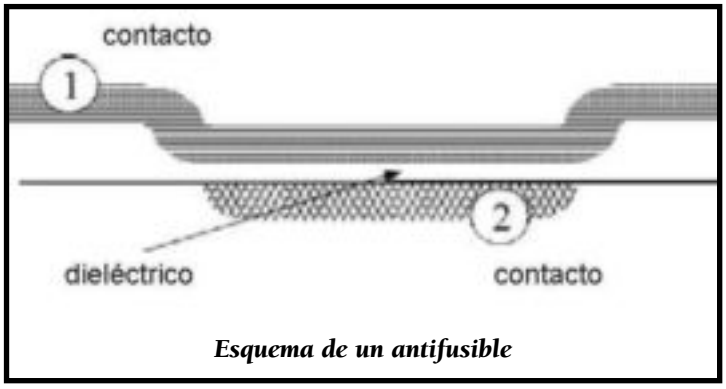


Esquema de unión implementada con fusible

El concepto de **antifusible** es el inverso:

- Sin programarlo, deja la unión abierta.
- Si se lo programa, se crea un cortocircuito interno y se unen eléctricamente sus extremos.

El antifusible se basa en el uso de dos placas de material conductor y, entre medio de ambas -a modo de sándwich-, un elemento semiconductor funciona como aislante. Al aplicarle tensión al semiconductor para programarlo, el aislante se elimina y quedan las placas conductoras en contacto, logrando que se unan eléctricamente. Exactamente lo contrario a un fusible; de allí su nombre.



Esquema de un antifusible

En la actualidad, los PLD están centrados en el uso de la tecnología CMOS en reemplazo de la bipolar (esta última se sigue fabricando sólo para mantener en stock aquellos dispositivos antiguos que requieren de repuestos).

La idea es que, en cada transistor que se use como llave, exista un elemento de memoria que lo configure para que esté siempre cerrado o siempre abierto.

Dentro de la tecnología CMOS, tenemos cinco tipos diferentes de elementos de memoria -ROM, PROM, EPROM, EEPROM, FLASH- para poder programar a cada uno de los transistores que funcionan como llaves. Un elemento de memoria basado en tecnología EPROM y sus derivados (EEPROM y FLASH) permite programar a la salida de éste, un estado lógico "0" o "1" y mantenerlo indefinidamente, aún luego de haber quitado la tensión de alimentación al dispositivo PLD. Esto sucede también con las tecnologías denominadas "fusible" y "antifusible". Estos tipos de elementos de memoria son no-volátiles: La información para la programación interna del PLD no se pierde al desconectar su fuente de alimentación. La única tecnología que es de tipo volátil -pierde su información al quitar la tensión de alimentación- es la SRAM -*Static Random Acces Memory*; memoria estática de acceso aleatorio-.

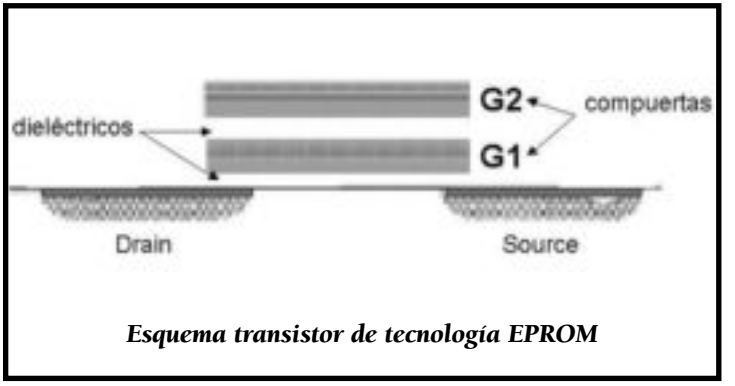
> Cuando hablamos de programación nos referimos a definir el estado de cada una de esas llaves: abiertas o cerradas.

Considerando si es posible reprogramar varias veces los dispositivos, podemos reclasificar las tecnologías en las que se basan los elementos de memoria en:

- Elementos de memoria de tecnología programable OTP -*One Time Programmable*- programables una sola vez: PROM, fusibles, antifusibles.
- Elementos de memoria de tecnología reprogramable: EPROM, EEPROM, FLASH y SRAM.

Los primeros dispositivos SPLD, tienen incorporada la tecnología PROM para poder ser programados, lo que permite configurarlos sólo una vez. Si algo falla o se requiere modificar su configuración, se debe emplear otro chip. Con EPROM se gana en cuanto a

poder lograr la reprogramación; para ello se incorpora al chip una ventana de cuarzo, por la que se puede borrar el contenido de la memoria interna exponiéndola a la luz ultravioleta; pero, los tiempos prolongados de espera hasta que este proceso se complete -alrededor de 30 minutos- constituyen una desventaja.



Esquema transistor de tecnología EPROM

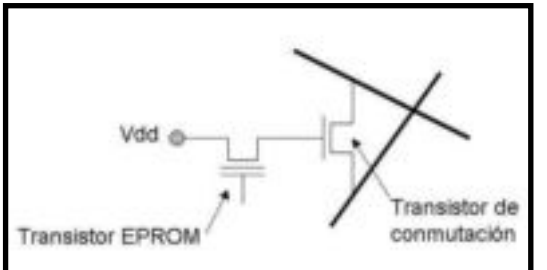
Una mejora lograda sobre la base de esta tecnología, son los elementos de memoria EEPROM y FLASH. La mayoría de los SPLD y CPLD actuales tienen elementos de memoria incorporados para poder configurar internamente a dichos dispositivos.

La tecnología EPROM y sus derivados (EEPROM y FLASH) se basa en el empleo de un transistor MOS que posee una doble compuerta, indicadas como G1 y G2: La externa es G2 mientras que G1 es la interna -compuerta flotante-. Esta constitución permite que el transistor pueda retener la información una vez que se lo ha programado.

- Si se quiere que el transistor se comporte como un circuito abierto, no se le aplica carga alguna a la compuerta G2.
- Si se desea que quede programado permanentemente en conducción cada vez que se lo alimenta, se aplica una tensión de programación a G2.

Al realizar esto, las cargas aplicadas quedan almacenadas en la compuerta flotante sin poder escapar, aún si se quita la tensión de alimentación.

A modo de ejemplo, en la figura siguiente, presentamos el caso de una unión programable entre dos líneas que transportan señales. Si el transistor EPROM está programado, actúa como un circuito cerrado y activa al transistor de conmutación uniendo las líneas de interconexión; en cambio, si no ha sido programado, el transistor de conmutación queda permanentemente cortado y las líneas no se unen.



Programación de transistor con tecnología EPROM

La desventaja de un transistor de este tipo es que no es posible borrar la programación, es decir, quitar las cargas acumuladas en la compuerta flotante, salvo que se aplique luz ultravioleta sobre dicha zona. Para ello, el dispositivo debe tener una ventana de cuarzo

en la parte superior del chip, a fin de borrar todos los elementos de memoria y reutilizar el integrado para otro proyecto.

Los transistores hechos con estas técnicas son similares a los implementados con EPROM. La diferencia radica en la construcción de las dos compuertas, que permiten el borrado eléctrico.

Dispositivos lógicos programables simples -SPLD-

El primer dispositivo de este tipo que se inventa es PAL *-Programmable Logic Array*; arreglo lógico programable-.

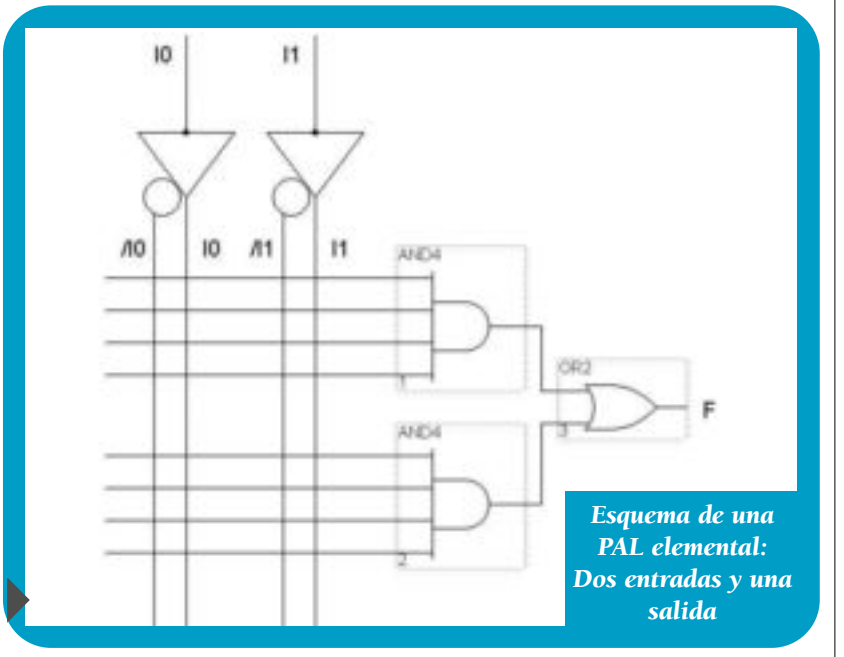
En un ejemplo sencillo tenemos un circuito de dos entradas y una salida. Está formado por una compuerta *or* de 2 entradas y 2 compuertas *and* de 4 entradas cada una. A cada *and* se conectan las 2 entradas externas y sus correspondientes negaciones, dando un total de 4 entradas. Las líneas horizontales y verticales forman una matriz de interconexión entre las entradas, sus negaciones y las entradas de las compuertas *and*; ésta es de 8 x 4 (8 líneas horizontales y 4 verticales). Si se unen intersecciones en forma coherente, es posible generar funciones lógicas.

Ejemplo 1. Si se une la intersección de la línea I0 con cualquiera de las entradas de la compuerta *and* superior y lo mismo con I1 con cualquiera de las entradas de la compuerta *and* inferior, dejando las demás intersecciones libres, se está generando la función F = I0 + I1.

Así es posible obtener varias funciones de 2 variables.

Pero, con este circuito no se pueden implementar todas las funciones posibles con 2 variables ya que se necesitan 16 términos producto y aquí sólo disponemos de 2 términos producto (2 *and*).

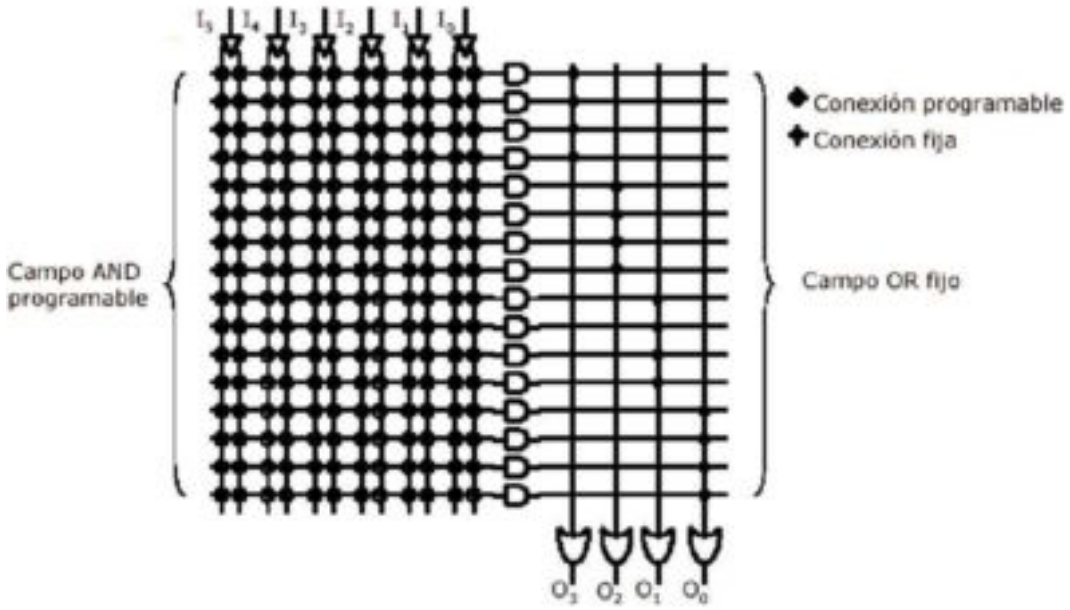
Ejemplo 2. Si a la compuerta *and* superior se conectan I0 y la negación de I1, y a la compuerta *and* inferior se conectan I1 y la negación de I0, se ha implementado una función *or*-exclusiva.



Esquema de una PAL elemental: Dos entradas y una salida

La función responde a la ecuación[11] F = /I0.I1 + I0./I1

En la siguiente figura vemos un circuito un poco más complejo:



PAL de 6 entradas y 4 salidas

Se trata de una PAL elemental que tiene 6 entradas y 4 salidas, y que está formada por una matriz de interconexión de 16 x 12 líneas, 16 compuertas *and* y 4 compuertas *or*.

Cada salida tiene asociada una compuerta *or* cuyas entradas se conectan a 4 salidas de *and* diferentes. Cada compuerta *and* se puede conectar a las 6 entradas y a sus negaciones. Tenemos, por lo tanto, 4 secciones *and-or* idénticas.

De esta manera, respecto del ejemplo anterior, es posible ahora resolver simultáneamente 4 funciones lógicas de 6 variables cada una, conectando convenientemente los cruces que correspondan.

[11] La barra indica negación.

En analogía con el ejemplo anterior, como en cada *or* no se tienen todos los términos productos posibles con 6 variables (64 en total), el número de funciones que se puede implementar en cada una de las salidas es limitado. Para lograrlo se hubiera requerido contar con 64 compuertas *and* que entraran a cada *or*. (Hemos planteado ex profeso esta cantidad de compuertas y de conexiones tan grande).

Término producto: Función lógica que se obtiene de hacer la función *and* de cualquier combinación entre las variables de entradas y sus negaciones.

Estudios realizados han demostrado que, en la mayoría de los casos, no es necesario contar con todos los términos productos para realizar una función; sólo son necesarios unos pocos de ellos. En este caso, como cada or tiene asociadas 4 compuertas and, los términos productos posibles son 4.

Si una función dada no puede resolverse con una sola sección *and-or*, puede emplearse una o más de las restantes e interconectar sus salidas a algunas de las entradas de la sección inicial[12].

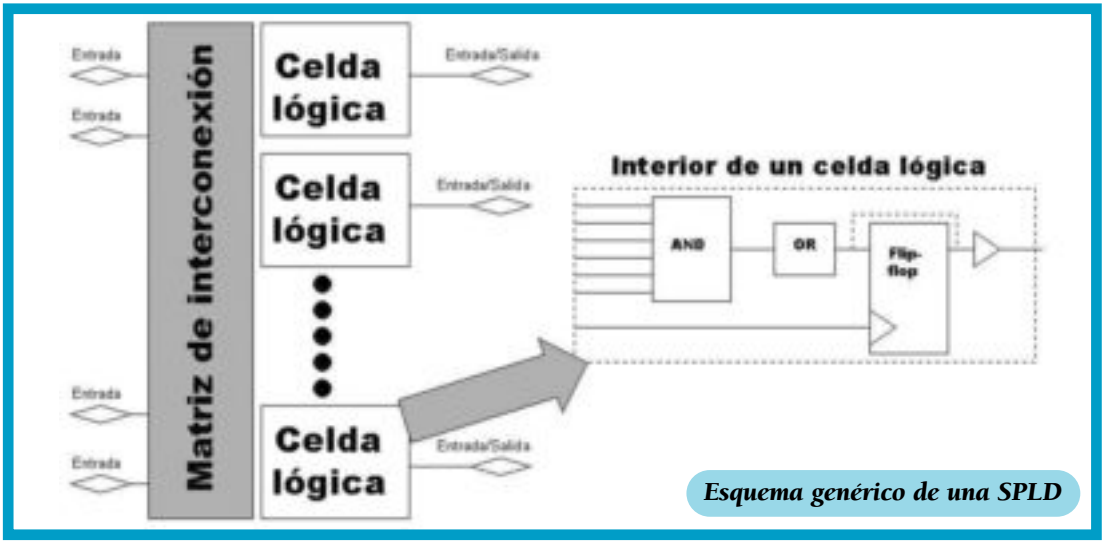
El esquema anterior es la base para fabricar la PAL.

Como se puede comprobar en el circuito, hay dos matrices, una la de las *and* y otra de las salidas de éstas con las *or*:

- la matriz de las *and* es programable,
- la matriz de las *or* es fija (cada *or* ya tiene asignada las salidas de 4 *and* y estas conexiones no se pueden remover).

Las primeras versiones de PAL disponían de fusibles en cada una de las intersecciones de la matriz *and*. Mediante un programador, se quemaban aquellos fusibles no deseados, y se dejaban sólo aquellos cuya variable o su negación se necesitaba que apareciera en la entrada de las *and*.

Como generalización, mostramos un diagrama en bloques de un dispositivo SPLD:



Esquema genérico de una SPLD

[12] Por simplicidad en el dibujo, las 12 entradas a cada *and* y a las 4 entradas a cada *or* se muestran con una sola línea de trazo.

Las características de este tipo de estructuras son:

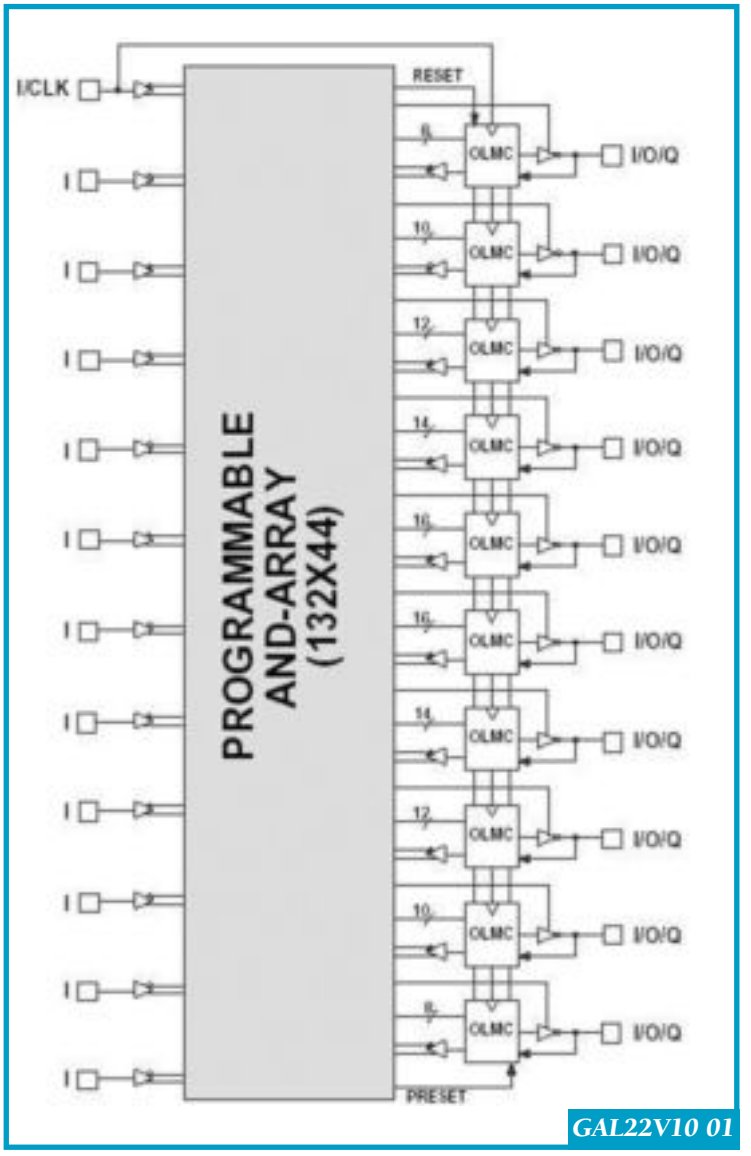
- Una macrocelda por salida.
- Un mínimo de dos macroceldas por dispositivo.
- Típicamente, todas las macroceldas iguales.
- Los términos producto o mintérminos o minitérminos, generados por una matriz *and-or*.
- Un flip-flop dedicado por cada macrocelda.

A través de esta estructura es posible programar:

- La matriz de interconexión.
- Internamente, cada macrocelda para generar lógica combinatoria o secuencial.
- La función de los pines conectados a cada una de las salidas de las macroceldas.

A continuación analizamos estos componentes en una versión moderna de una PAL la denominada GAL22V10 de la empresa Lattice®[13].

Se trata de una PAL -denominada GAL por el fabricante-, implementada en CMOS, que utiliza transistores de efecto de campo con tecnología EEPROM, para programar la matriz de interconexión entre 44 líneas verticales y 132 horizontales. Observe este diagrama en bloques donde esta matriz se representa como un bloque rectangular.



GAL22V10 01

[13] Lattice Semiconductor Corporation: http://www.latticesemi.com/

El dispositivo está formado por 10 bloques lógicos OLMC *-Output Logic Macrocell*; macrocelda de lógica de salida-. Cada uno de ellos termina en un pin del chip que puede ser usado como entrada o como salida; tenemos, por lo tanto, 10 pines de Entrada/Salida.

Hay 11 pines que funcionan como entradas de señales solamente -denominadas I- y otra -ICLK- que sirve como entrada de reloj a cada uno de los flip-flops que se encuentran implementados en cada módulo OLMC o como una entrada adicional a las 11 restantes.

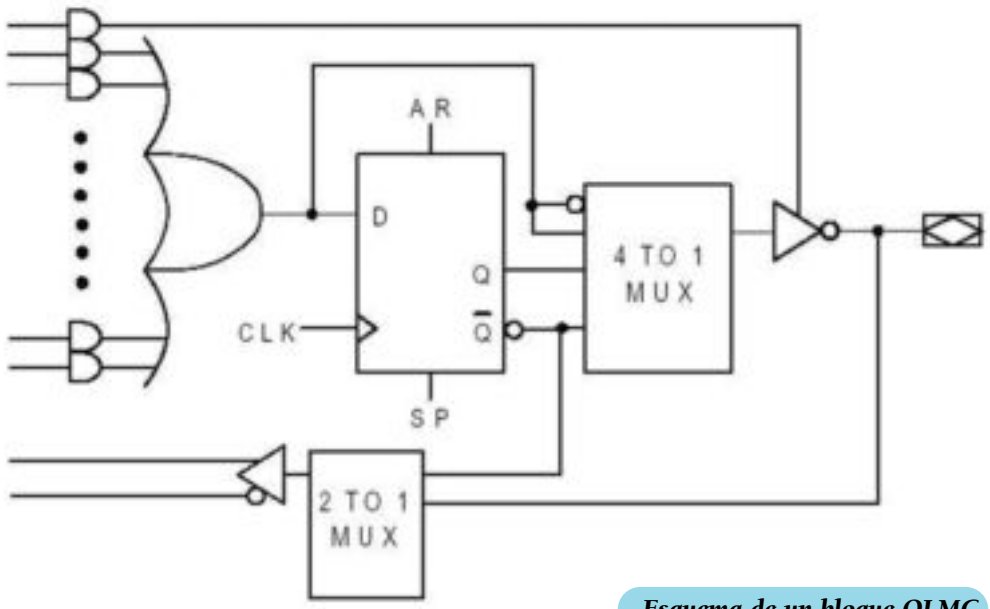
De cada uno de los pines de entrada, se envía a la matriz tanto la entrada sin negar, como su negación.

El bloque OLMC está formado por varias compuertas *and* de múltiples entradas cada una, que provienen de la matriz de interconexión. Todas ellas se conectan a una compuerta *or*.

La salida de dicha compuerta se conecta a un flip-flop tipo D, por un lado, y a dos entradas de un multiplexor *-mux-* de 4:1 (4 entradas de datos: 1 salida), por el otro.

Las salidas del flip-flop Q y /Q también se conectan a las entradas de dicho multiplexor. Su salida, vía un *buffer* inversor *tri-state*, envía la señal a un pin del integrado.

Por último, otro MUX 2:1 (2 entradas de datos: 1 salida), se emplea para seleccionar una de dos fuentes posibles de señal para in-



Esquema de un bloque OLMC

yectar a la matriz de interconexión. Las opciones son: desde el pin mismo o desde la salida negada del flip-flop.

> Recuerde que *tri-state* -tercer estado- es una condición de algunos *buffers* que pueden poner su salida en estado de alta impedancia.

Se puede notar, además, que hay otras tres señales denominadas AR, SP y CLK. Las dos primeras son las entradas de "reset asincrónico" y "preset asincrónico" del flip-flop. La tercera, la entrada de reloj.

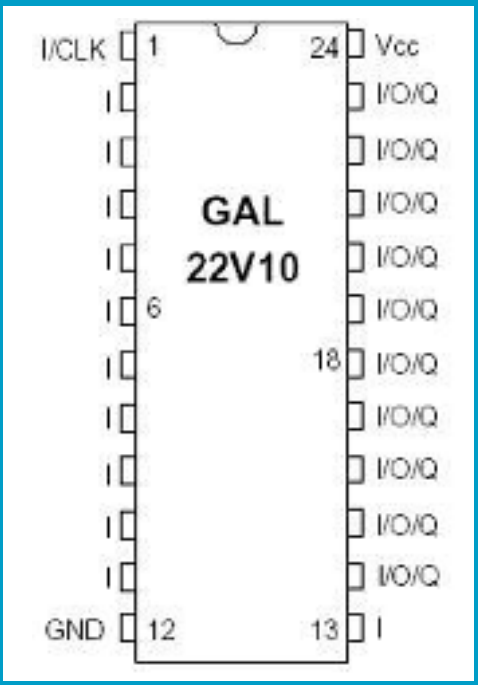
Este esquema es el mismo para cada uno de los 10 OLMC que tiene la GAL22V10.

Cada uno de los OLMC puede manejar un número diferente de términos producto (combinaciones entre las entradas y sus negaciones). A mayor número de términos producto, mayor es el número de funciones que se pueden implementar.

De este análisis, se puede deducir que:

- Es posible elegir que la salida sea registrada o no registrada, es decir, con el flip-flop entre la función generada por las compuertas *and-or* o, directamente, con la salida de la *or* hacia el pin de salida.
- Se puede elegir si la salida es con *tri-state* o no. Esta función *tri-state* es útil para el caso en que, por ejemplo, se quisiera usar el pin de conexión de una OLMC solamente como entrada. Esto se puede lograr inhibiendo el buffer (poniendo su salida en estado de alta impedancia), a través de la entrada de control que proviene de la matriz de interconexión.
- Se puede optar si la salida de la lógica de compuertas es negada o sin negar. Esto es posible ya que al MUX 4:1 ingresan, la salida de la *or* y su negación.
- Con el MUX 2:1 es posible seleccionar una señal que retorne a la matriz de interconexión: Puede ser el pin externo de la OLMC o la salida negada del flip-flop. Este rasgo es importante, ya que es el que hace posible implementar, por ejemplo, un contador o un registro de desplazamiento que requiera que la salida de un flip-flop se reutilice para tales implementaciones.

Tanto en SPLD como en CPLD y FPGA, el *mux* juega un papel muy importante a la hora de servir como circuito de selección de señales. Un *mux* 4:1 con 2 líneas de selección, puede elegir cuál de sus 4 entradas de datos va a estar presente a su salida. Es -ni más ni menos- una llave selectora digital.



Esquema de disposición de pines para encapsulado tipo DIP -Dual In-line Plastic; plástico doble en línea-

Ventajas de las SPLD[14]:

- Dado que las interconexiones que se pueden realizar son pocas, los tiempos de retardo que se obtienen son bastante predecibles; es decir, son fáciles de calcular.
- Son fáciles de usar en un diseño.

Desventajas de las SPLD:

- La utilización de los recursos en ineficiente. Por ejemplo, si en una OLMC no se usa el flip-flop, éste se pierde.
- Se pueden realizar sólo funciones lógicas muy simples, y con poca cantidad de variables de entrada y salida.

Según los vendedores se encuentran diferentes denominaciones de este tipo de chip, tales como:

- PAL *-Programmable Array Logic-*
- GAL *-Generic Array Logic-*
- PLA *-Programmable Logic Array-*

Dispositivos lógicos programables complejos -CPLD-

Estos dispositivos PLD complejos son el paso siguiente de las transformaciones de la lógica programada por hardware.

Las SPLD tienen ciertas limitaciones. Algunas de ellas son:

- Poca flexibilidad en el uso de elementos internos en cada macrocelda para resolver ciertas funciones lógicas.
- Del mismo modo, en la matriz empleada para interconectar entradas y señales de realimentación a las matrices *and-or*.
- Desperdicio de aquellos pines conectados a las macroceldas en las que las salidas se usen para reinyectar señales a la matriz de interconexión.

Es por ello que, para resolver todos estos problemas, con la creciente necesidad de poder implementar circuitos cada vez más grandes y complejos, aparecen los CPLD.

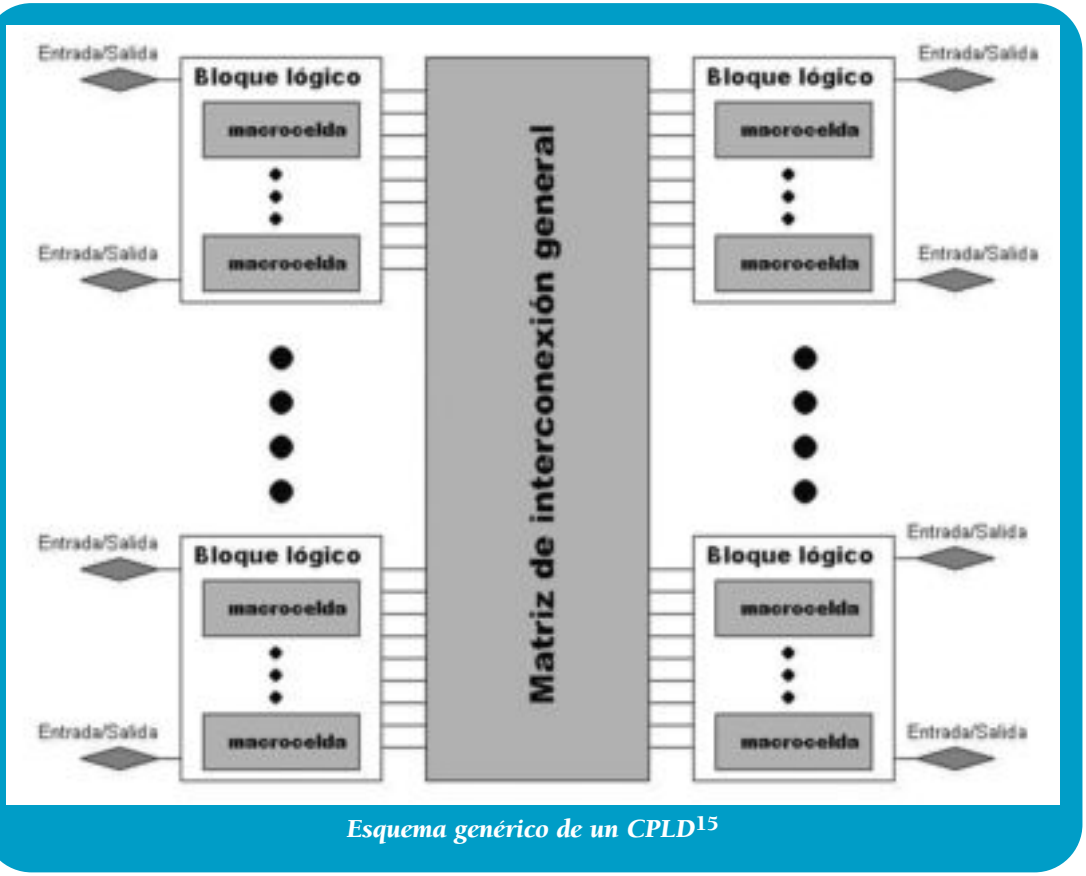
La concepción de esta estructura divide al chip en tres partes perfectamente separadas:

- Bloque de macroceldas.
- Bloque de matriz para interconexión entre macroceldas, y entre macroceldas y pines.
- Bloque de entrada/salida, formado por los pines físicos reales y la lógica asociada a ellos.

Las características más sobresalientes de estos dispositivos son:

- Los términos productos generados se programan dentro de cada macrocelda y no en una gran matriz -como es el caso de los SPLD-.

[14] La SPLD analizada, corresponde a un modelo nuevo de SPLD, que incluye mejoras respecto de las primeras versiones que salieron al mercado al fin de la década de 1970. Hacemos esta aclaración porque la estructura interna de los primitivos SPLD era mucho menos flexible que la de los OLMC vistos en el chip GAL22V10; en ellos, por ejemplo, no se utilizaban multiplexores como elementos de selección. Además, en esas primeras versiones, no se tenía la posibilidad de seleccionar si la salida podía ser registrada (con un flip-flop entre la salida real y la función *or* o *nor*) o no-registrada (sólo lógica combinatoria).



Esquema genérico de un CPLD[15]

- Hay un flip-flop dedicado en cada macrocelda, con todas sus entradas totalmente programables.
- Las macroceldas se agrupan en bloques.
- No existen pines dedicados a una macrocelda dada; éstos forman parte de un bloque separado.
- En general, los pines son todos de entrada/salida.

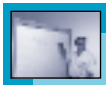
Los dispositivos CPLD adoptan la tecnología EEPROM para almacenar la información necesaria para la programación de los componentes internos y la matriz de interconexión.

A continuación analizamos la estructura de una CPLD, la familia MAX7000 de la empresa *Altera*®[16].

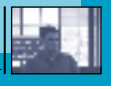
La familia MAX7000 es una de línea de dispositivos lógicos programables. Se trata de CPLD implementados en CMOS que emplean la tecnología EEPROM en los elementos de memoria, a fin de mantener la información de programación de los componentes internos configurables de estos chips.

[15] En la figura, el bloque entrada/salida está incluido en los denominados "bloques lógicos", aún cuando algunos fabricantes no incluyen este bloque como algo separado de los bloques lógicos.

[16] Altera: http://www.altera.com



El chip modelo EPM7128 es el que utilizamos como base para la generación del entrenador en lógica programada que le proponemos desarrollar con sus alumnos.



Trabajan con alimentación de 5 volt.

Están basados en el empleo de macroceldas que, según el modelo los chips, pueden contener desde 32 (la EPM7032) hasta 256 macroceldas (la EPM7256). Cada macrocelda contiene una matriz programable *and-or* para la síntesis de la función lógica deseada. Su salida puede ser no-registrada o registrada, gracias al flip-flop interno totalmente configurable.

Las macroceldas son agrupadas de a 16, formando los denominados LAB *-Logic Array Block*; bloque de arreglos lógicos-.

En esta estructura, las macroceldas están agrupadas de a 16 en cada LAB. Una EPM732 tiene 2 LAB con 16 macroceldas cada una; en cambio, una EPM256 tiene 16 LAB de 16 macroceldas en cada LAB.

Los pines físicos del chip pueden programarse como entrada, salida o entrada/salida. Están vinculados no a una macrocelda sino a un bloque de entrada/salida.

Cada bloque de entrada/salida -bloque I/O- nuclea de 6 a 16 pines físicos y puede conectarlos a la matriz de interconexión interna -PIA- o a los LAB.

Esto permite una gran flexibilidad ya que, salvo algunas excepciones, es posible que el usuario defina por dónde entra o sale una señal lógica del exterior.

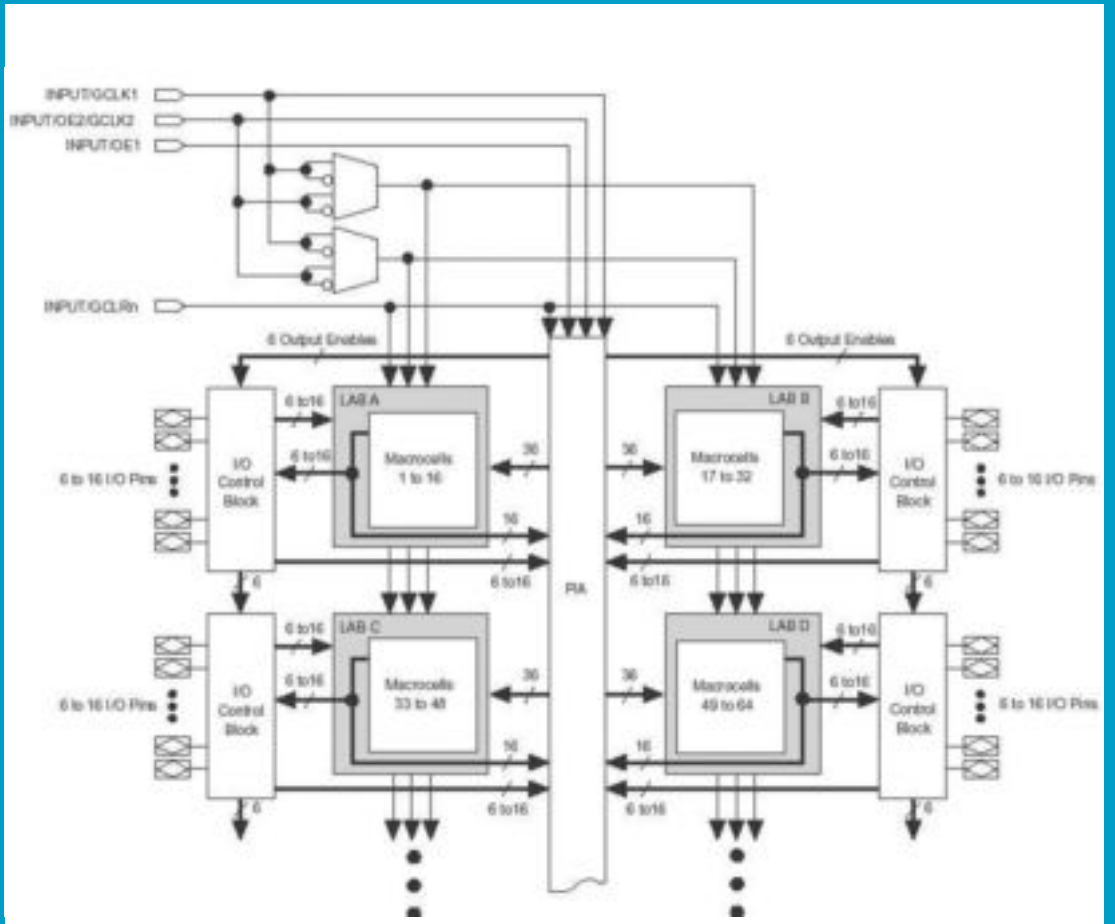
La PIA es análoga a un conjunto de autopistas verticales en cada una de las cuales existe una gran cantidad de carriles.

Tanto los LAB como los bloques I/O, se conectan a la PIA a través de conductores horizontales.

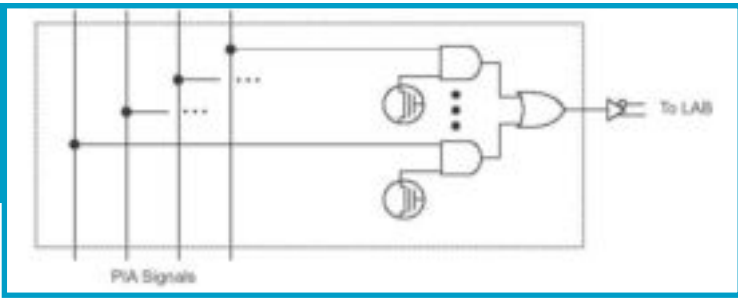
Diagrama en bloques de la MAX7000 tal como se presenta en su hoja de datos

Componentes:

- ***LAB -Logic Array Block; bloque de arreglos lógicos-.***
  Macrocelda -Macrocell-.
- ***I/O -Control Block; bloque de control de entradas/salidas-.***
- ***I/O -pins; pines de entrada/salida-.***
- ***PIA -Programmable Interconnect Array; arreglo de interconexión programable-.***
- ***Entradas de funciones específicas: GCLK1, GCLK2, GCLRn, OE1, OE2.***



Existe una gran cantidad de posibles conexiones entre la PIA y los conductores, posibilitando, así, generar la lógica deseada.



Ruteo de información entre la PIA y un LAB de la MAX7000[17]

[17] Seguimos mostrándole los planos originales, por lo que mantenemos el inglés de su nomenclatura.

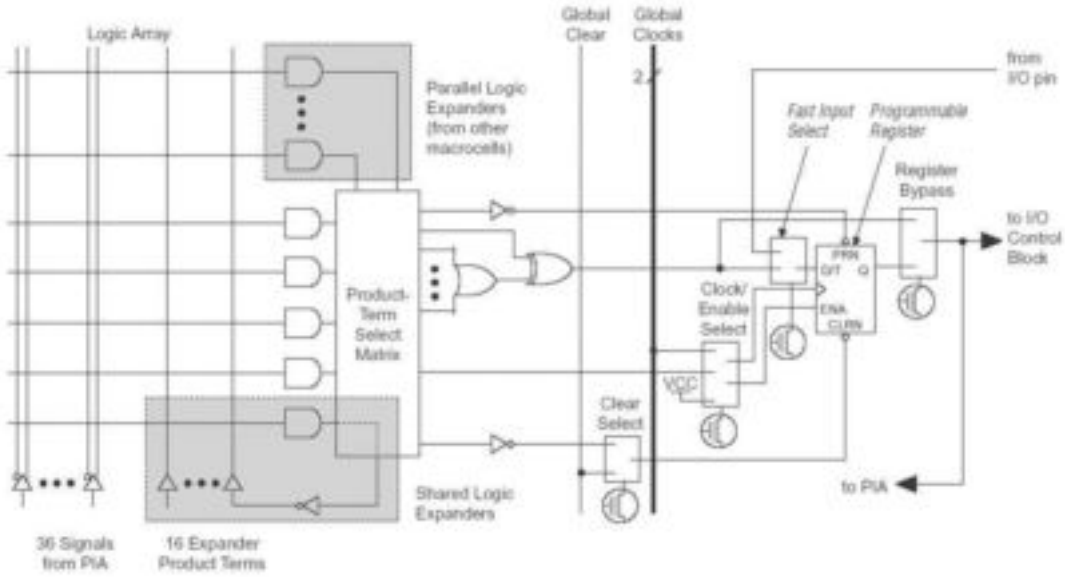
De los varios carriles que posee la PIA, es posible seleccionar uno para que vaya a un LAB en particular.

Cada señal de la PIA va a la entrada de una *and* de 2 entradas. La otra entrada tiene asociado un transistor FET que se programa para que dicha entrada se ponga a "0" (inhibiendo la salida de la and) o a "1" lógico (habilitando dicha salida).

Para elegir cuál señal de la PIA va a un LAB determinado, basta con poner un "1" en dicha compuerta y un "0" en las demás.

Note que la salida de la *or* va a un *buffer* que permite ingresar a un LAB con la señal PIA seleccionada, negada o sin negar.

Cada macrocelda contenida en un LAB tiene la siguiente estructura:



Macrocelda de la MAX7000

- ***Matriz de selección de término producto - Product-Term Select Matrix-. Interconecta varias compuertas and con una or donde se sintetiza la función lógica requerida.***

Siguiendo con el análisis de la estructura de la MAX7000, el esquema anterior nos permite observar que la salida de la *or* se conecta a la entrada de una or-exclusiva de 2 entradas.

De acuerdo a cómo se programe la otra entrada de la *or-exclusiva*, se puede tener a su salida la función negada o sin negar, de la matriz *and-or* resuelta.

- ***Un flip-flop, programable a través de varios multiplexores.***

- ***Expansores lógicos paralelo -Parallel Logic Expanders-. Permiten incluir otros términos producto, desde la salida de otras macroceldas. Esto posibilita, por ejemplo, realizar funciones con mayor número de entradas o mayor número de términos productos de las que dispone una sola macrocelda.***

- ***Expansores lógicos compartidos -Shared Logic Expanders-. Cada macrocelda dispone de compuertas and que pueden ser empleadas por otras macroceldas a fin de generar lógica muy compleja. Las salidas de estas and se pueden reinyectar nuevamente a la PIA.***

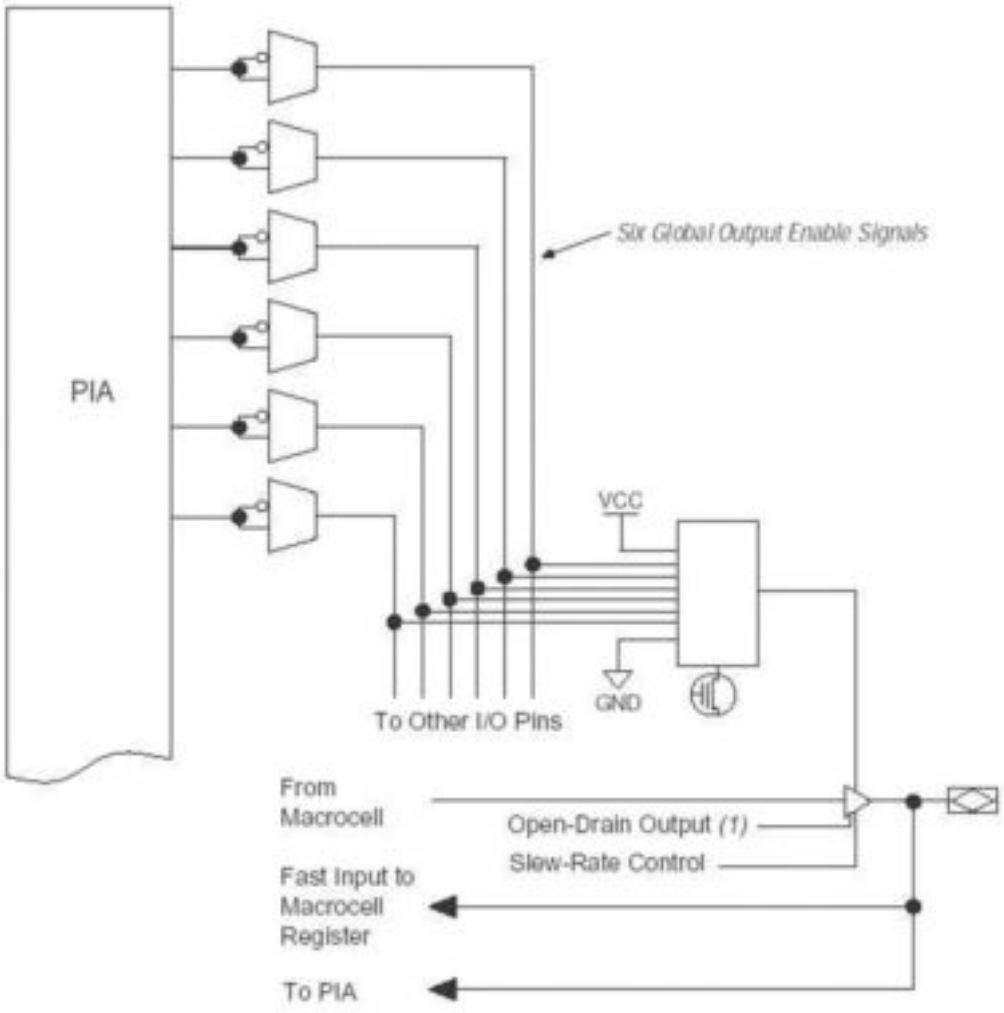
Luego, de esta compuerta -mediante el empleo de un mux denominado *Register Bypass*; registro de puenteo-, se puede seleccionar que dicha señal salga de la macrocelda hacia el bloque I/O correspondiente y, simultáneamente, introducirla a la PIA nuevamente.

El flip-flop, por otro lado, se puede programar para que su entrada de datos reciba señal de dicha or-exclusiva o desde un pin de I/O. Esto se selecciona con otro mux denominado *Fast Input Select* -selección rápida de entrada-.

El flip-flop puede ser programado para que funcione como tipo D, JK, SR o T. Sus entradas de CLK -reloj-, *Clear* -borrado-, *Preset* -ajuste-, *Enable* -habilitación de reloj-, pueden también ser programadas para recibir señales de diferentes fuentes; por ejemplo, la entrada de reloj, puede venir de la matriz de selección de término de producto o de uno de dos pines específicos de entrada para señales de reloj globales, denominadas GCLK1 y GCLK2.

Lo mismo sucede con la entrada de *Clear* del flip-flop. Puede venir de la matriz de selección de término de producto o del pin específico de entrada para señales de borrado global, denominada GCLRn.

Por último, analicemos uno de los elementos que conforman cada bloque de control de entrada/salida de la MAX7000.



Bloque de entrada-salida de la MAX7000

Se puede apreciar que cada uno de los pines físicos de la EPLD está asociado a un buffer no inversor del tipo *tri-state* (3 estados: "0", "1" y Z o de "alta impedancia"). La entrada de control de este estado proviene de un mux 8:1 (8 entradas de datos y 1 salida).

Empleando 2 líneas de selección para dicho *mux*, podemos seleccionar 8 diferentes fuentes de señal que comanden dicho *buffer*.

Como se puede observar en la figura, 6 de dichas señales provienen de la PIA y las otras 2 son Vcc (+5 V) y GND (masa).

Si se aplica Vcc, el buffer está siempre activo. Con GND, dicho buffer queda inactivo; por lo tanto, el pin se configura sólo para entrada de señales desde el exterior.

Empleando cualquiera de las opciones que vienen desde la PIA, se puede usar dicho pin como bidireccional -es decir, como entrada o salida-, según el nivel lógico de la línea de control proveniente de la PIA.

Se puede observar, además, que desde el pin hay dos caminos hacia dentro del dispositivo: Uno es directo hacia la PIA y el otro a un *mux* de una macrocelda que está directamente asociada a ese pin. El multiplexor puede programarse -como planteamos al analizar las macroceldas- para que la señal proveniente de dicho pin vaya directamente a la entrada del flip-flop de la macrocelda, convirtiendo a dicha entrada como registrada; es decir, en una entrada cuyo valor se captura dentro del dispositivo sólo cuando se activa el flip-flop correspondiente.

Las principales ventajas de las CPLD son:

- Tiempos de retardo predecibles, al igual que las SPLD.
- Retardos internos relativamente bajos.
- Eficiencia de los recursos internos, debido al diseño de la matriz de interconexión.
- Aptitud para diseños de baja a mediana complejidad.

Principales desventajas:

- Para implementar diseños de gran complejidad, la matriz se hace muy grande y, por lo tanto, muy cara.
- No permite emular memoria como las FPGA.

Según los vendedores, se pueden encontrar diferentes denominaciones de este tipo de chip tales como:

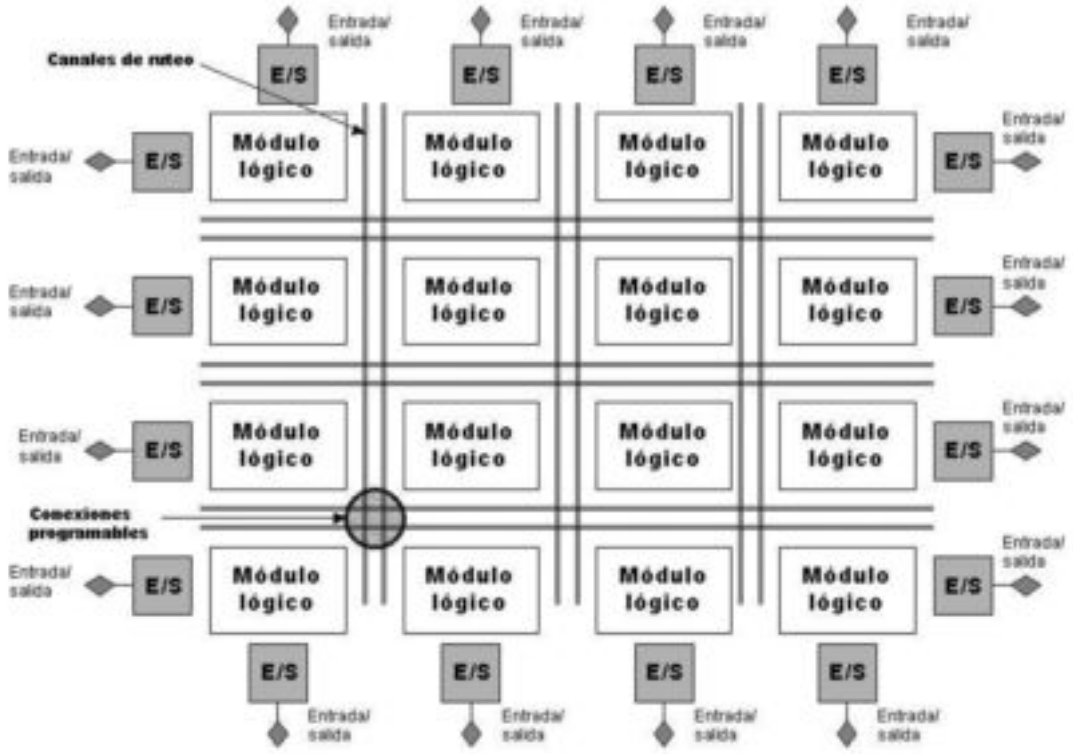
- CPLD -*Complex Programmable Logic Device*; dispositivo lógico programable complejo-.
- EPLD -*Erasable Programmable Logic Device*; dispositivo lógico programable borrable-.
- EEPLD -*Electrically-Erasable Programmable Logic Device*; dispositivo lógico programable borrable eléctricamente-.
- XPLD -*Expanded Programmable Logic Device*; dispositivo lógico programable expandido-.
- SPLD -*Segmented Programmable Logic Device*; dispositivo lógico programable simple[18]- .

[18] Esta última categoría no debe confundirse con la sigla SPLD en la que la letra S significa simple.

Arreglos lógicos programables -FPGA-

Los FPGA -*Field Programmable Gate Array*; arreglos de compuertas programables por el usuario- son desarrollados en 1984 por la empresa *Xilinx*® y constituyen el tercer paso en las transformaciones de los PLD.

La arquitectura de las FPGA consiste en muchos módulos lógicos ubicados en una estructura del tipo de arreglo matricial:



Esquema genérico FPGA

En general, una FPGA está formada por un número muy grande de módulos lógicos cada uno de los cuales tiene un circuito de lógica combinatoria y un flip-flop, en forma similar a las CPLD. Existe, además, un gran número de canales entre los módulos lógicos que son usados como elementos de ruteo.

Este tipo de disposición de canales permite una mayor flexibilidad para interconectar los módulos lógicos entre sí.

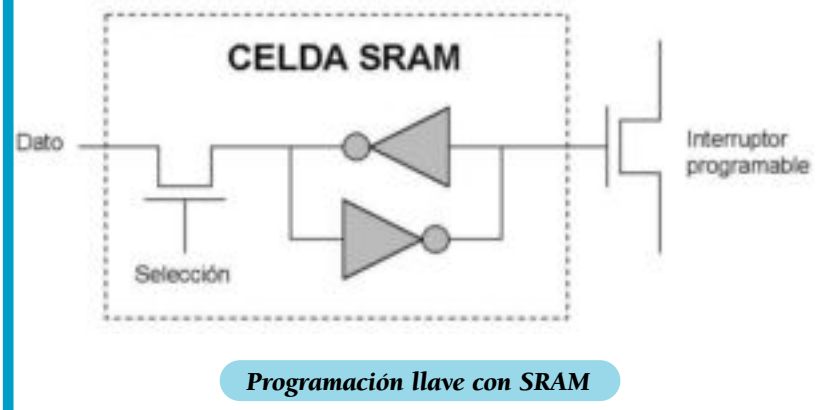
Rodeando dicho arreglo de módulos lógicos, se ubican los bloques de entrada/salida, los que tienen funciones similares a las de los CPLD.

Existen FPGA que son de grano fino y de grano grueso.

- Grano *fino*: Contiene muchos módulos en el chip. Cada módulo tiene, entonces, poca capacidad de implementar lógica combinatoria (por ejemplo, funciones de sólo 3 ó 4 variables de entrada).
- Grano *grueso*: Por el contrario, es una arquitectura en cuyos chip hay pocos módulos lógicos; pero, cada uno de ellos puede implementar lógica combinatoria de más variables de entrada (por ejemplo, de hasta 7 variables) e, inclusive, más de una función lógica por módulo (es posible encontrar módulos que pueden implementar hasta 4 funciones simultáneas de 2 variables de entrada cada uno).

Los elementos de memoria que suelen utilizarse en las FPGA para almacenamiento de la programación de interconexiones y componentes, son de dos tipos: el antifusible y SRAM. El primero constituye una tecnología no-volátil (mantiene lo programado aunque se quite la tensión de alimentación); el segundo tipo, no.

Para segundo caso, vemos un esquema de una celda de memoria SRAM que excita directamente a un transistor que puede ser empleado como llave:



Programación llave con SRAM

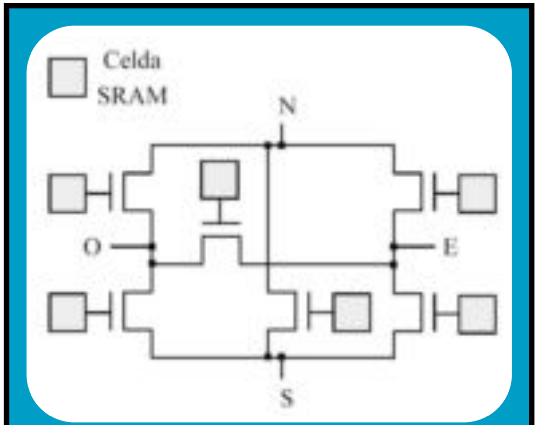
Este circuito consta de un transistor (lado izquierdo) que funciona como habilitador de señal y de un par de inversores conectados en antiparalelo que funcionan como un biestable (elemento básico de memoria).

Cuando la señal "Selección" lo permite, la información de "Dato" hace cambiar la salida del biestable. Al volver a inhibir al transistor de paso, queda almacenando el último valor que tenía "Dato".

En el ejemplo, esta celda SRAM se conecta al *Gate* de un transistor FET. De esta manera, se puede comandar su estado para hacerlo funcionar como una llave.

> Una celda SRAM está formada por 5 transistores MOS. Uno como habilitador y 4 como inversores. Recuerde que los inversores CMOS se fabrican con 2 transistores MOS.

Un caso más complejo es:

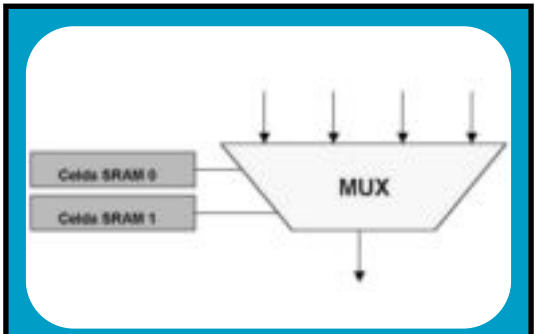


Llaves en XC4000

Aquí tenemos 4 cables denominados N, O, E y S que se pueden conectar empleando los 6 transistores dibujados. Este esquema permite tener flexibilidad en cuanto a las diferentes combinaciones entre ellos.

Cada transistor que funciona como llave está controlado por una celda de memoria.

Otro ejemplo de utilización de la celda SRAM es para configurar un multiplexor:



Programación mux

En este caso, programando las entradas de selección del mux 4:1 con 2 celdas de memoria, es posible seleccionar permanentemente cuál de las 4 entradas se conecta a la salida del *mux*.

Analicemos, ahora, la FPGA de la familia XC4000 de la empresa *Xilinx*®[19]. Esta familia comprende una serie de dispositivos, desde el más pequeño: XC4002XL hasta el más grande: XC4085XL.

Cada dispositivo está formado por un conjunto de bloques lógicos denominados CLB - *Configurable Logic Block*; bloque lógico configurable-, dispuestos en forma de un arreglo de N x N bloques: La XC4002X cuenta con un arreglo de 8 x 8 CLB que dan un total de 64; mientras que, la XC4085XL tiene un arreglo de 56 x 56 CLB, dando un total de 3136.

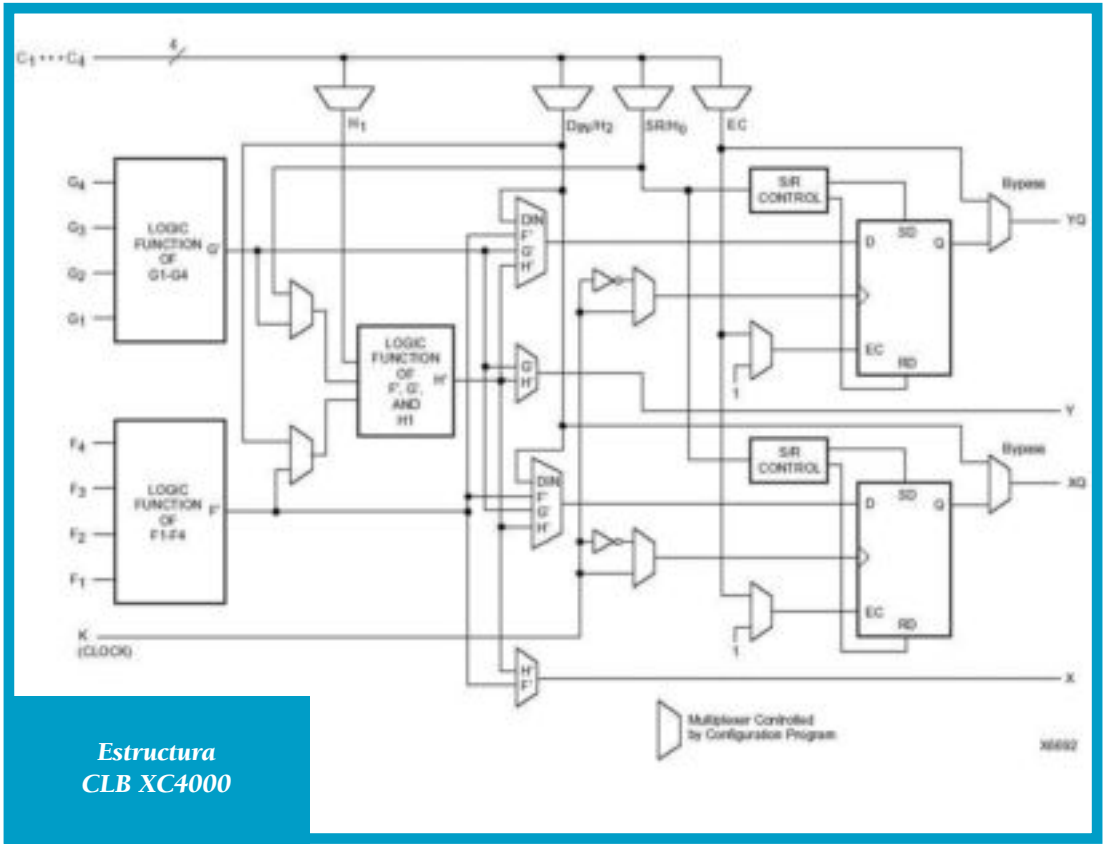
[19] Xilinx: *Programmable Logic Devices*, FPGA & CPLD: **http://www.xilinx.com/**

Estos CLB se pueden interconectar entre sí gracias a una sofisticada red de interconexión que permite una gran flexibilidad para implementar lógica compleja.

Rodeando a todo el arreglo de CLB se encuentran ubicados los IOB *-Input/Output Blocks*; bloques de entrada/salida-. Éstos disponen de circuitos internos que permiten una gran posibilidad de interconexión entre las entradas y/o salidas, y los recursos internos del chip.

Este tipo de FPGA emplea en el diseño de los CLB una estructura denominada LUT *-Look-Up Table*; tabla de Look-Up- que permite no sólo generar las funciones combinatorias necesarias sino, además, implementar varios tipos de memorias. Esto es muy importante ya que dentro del chip, por ejemplo, puede construirse un circuito que tenga la habilidad de obtener información desde el exterior, guardarlo en la memoria y permitir que otro dispositivo la lea. Esta característica es única y no está presente en los CPLD; es original en las FPGA basadas en tecnología SRAM.

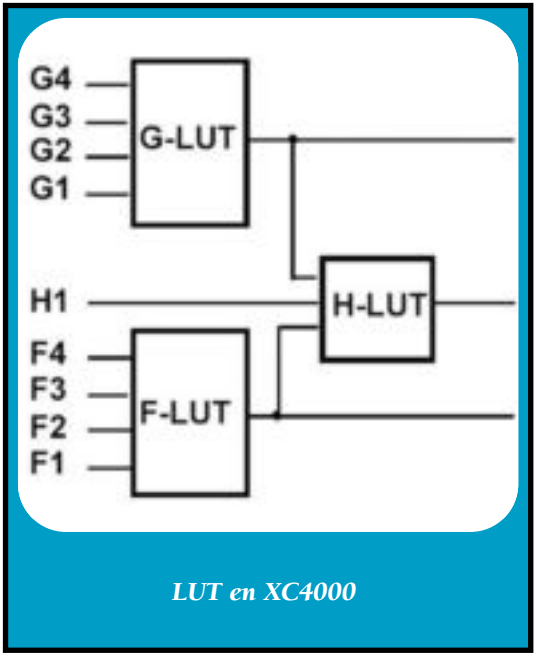
Trabaja con tensiones de 3.3 V y es tolerante a lógica de 5 V; es decir que, aunque se alimente con 3,3 V internamente, puede reconocer niveles lógicos de señales de entrada que provengan de dispositivos TTL.



Estructura CLB XC4000

Cada bloque consta de 3 generadores de funciones basados en una LUT, indicados como F1-F4, G1-G4y H1.

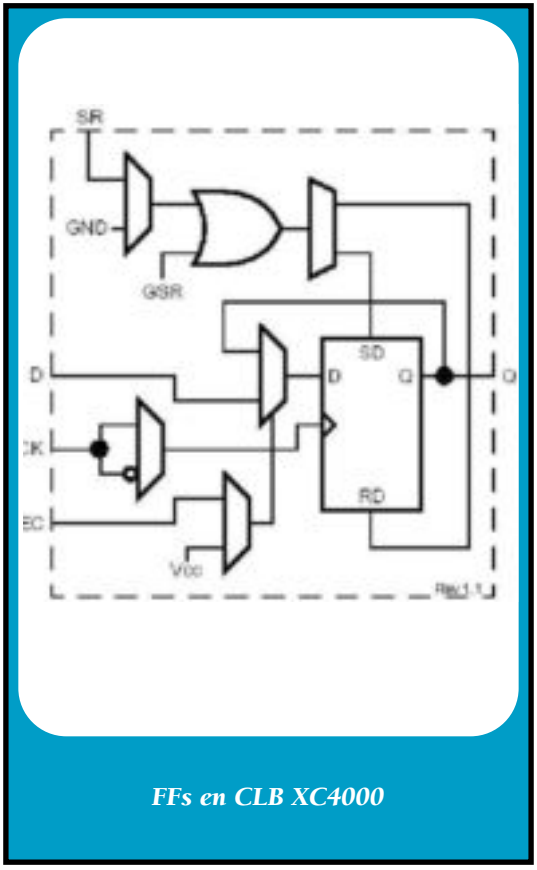
Los dos primeros pueden sintetizar funciones lógicas de 4 variables de entrada y 1 salida. El tercero realiza lo mismo pero de 3 variables de entrada, que son las salidas de los bloques anteriores más otra señal que puede provenir, por ejemplo, de otro CLB o del exterior.



LUT en XC4000

Incluye, además, 2 flip-flops tipo D cuyas entradas pueden provenir tanto de alguno de los generadores de funciones como de combinación de ellos o de la matriz de interconexión. Cada uno puede programarse en forma similar a lo que hemos planteado para CPLD.

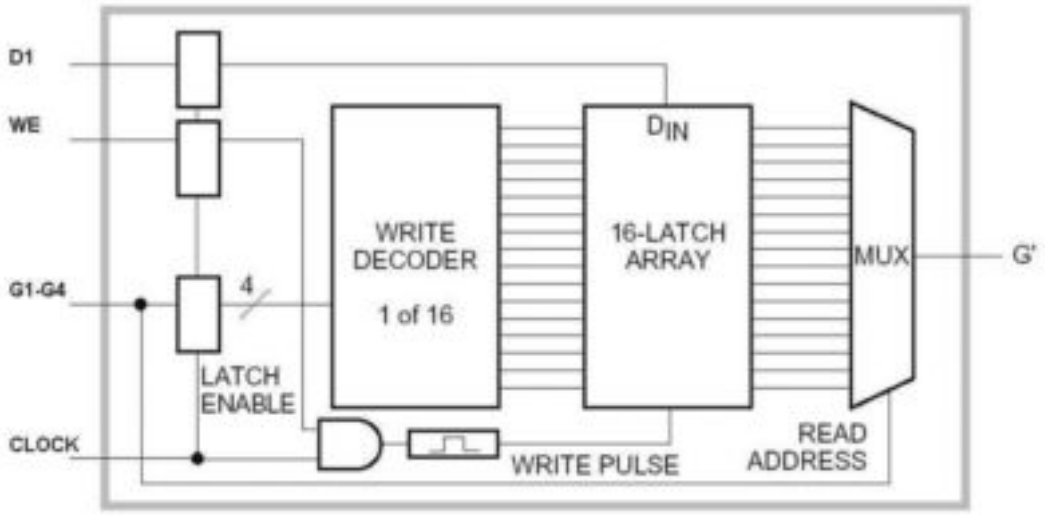
Mediante el empleo de *mux* es posible seleccionar fuente de entrada, señal de reloj, *reset*, *preset* y a dónde irá su salida Q.



FFs en CLB XC4000

Tiene 4 salidas diferentes: Y, X, YQ y XQ; todas ellas se interconectan a la matriz general del chip. Esto da una importante flexibilidad ya que, por ejemplo, en un CLB se puede, por un lado, usar la lógica combinatoria para generar más de una función lógica y, por el otro, emplear uno o ambos flip-flop en forma independiente para ser usados por otra parte del chip. Esta habilidad tampoco estaba presente en los CPLD.

En la siguiente figura vemos la estructura interna de una de las LUT, la que genera la función G´. El circuito para el generador de función F es idéntico.



LUT G XC4000

Dicha LUT está formada por un multiplexor de 16:1 (*mux*), un registro de desplazamiento serie-paralelo de 16 bits (16 *latch-array*) y un decodificador de escritura de 1:16 (*write decoder*).

La base para implementar una función lógica está cumplimentada por el uso del *mux* 16:1 (16 entradas y 1 salida). Con este componente podemos sintetizar cualquier función lógica de 4 variables (G1, G2, G3 y G4) que se entran por las líneas de selección del *mux*.

Recuerde que una función de 4 variables siempre se puede formar con una combinación dada entre los 16 mintérminos que existen para dicha cantidad de variables de entrada. Si el mintérmino existe en la función, entonces es necesario poner un "1" en la entrada de datos que corresponda y, si no aparece, se debe poner un "0". De esta manera, se completan las 16 entradas del mux con los "0" y "1" que se necesiten, implementándose así la función lógica requerida (Por ejemplo: Si una función tiene los mintérminos 0 -las 4 variables negadas- y 7 -las 4 variables sin negar-, entonces en la primera y última entrada de datos del *mux* se pone un "1" y, en las 14 restantes, un "0").

Como el *mux* sólo puede implementar funciones, se emplea el registro de desplazamiento para poder programarlo.

En esencia, una LUT es la combinación entre un *mux* de N:1 y un registro de desplazamiento serie-paralelo de N bits.

¿Por qué programar? En general, en las FPGA reprogramables hay que programar no sólo la matriz de interconexión -para que se unan los caminos necesarios para armar el circuito deseado- sino que, en los CLB usados, hay que programar cada generador de función.

Programar significa poner "0" y "1" en las entradas de datos del *mux* para que sintetice la función pedida. Esto se hace en forma "serie", con la ayuda del registro de desplazamiento.

Olvidemos, por un momento, que existe el decodificador de escritura. Por la entrada D1 -acceso a la entrada del primer flip-flop del mencionado registro- se va ingresando de a un dato lógico por vez, usando para ello una señal de reloj (*clk*) que sincroniza su aparición.

Así, cuando se mandan 16 ciclos de reloj, el registro de desplazamiento tiene sus 16 salidas con los 16 datos entrados secuencialmente.

Si se observa la figura, dichas salidas se conectan directamente a las entradas de datos del *mux*. Por lo tanto, luego de esas 16 cargas, el *mux* ha quedado ya configurado, y resta sólo entrar las señales por G1, G2, G3 y G4 para implementar la función lógica deseada.

Así, por ejemplo, desde una computadora se programa en forma "serie" a este componente.

En el modo de implementación de una memoria, el bloque LUT se configura de igual manera que antes.

Las direcciones de memoria corresponden a las líneas G1 a G4 en el LUT G y de F1 a F4 para la LUT F. De esta manera, en cada LUT se puede implementar una memoria de 1 x 16 bits -16 posiciones de memoria (o registros) de 1 bit cada una-.

Habiendo cargado los 16 valores en los registros de desplazamiento de ambas LUT, se puede acceder a cualquiera de los 16 registros de cada LUT, direccionando apropiadamente con los 4 bits de G1-G4 y F1-F4.

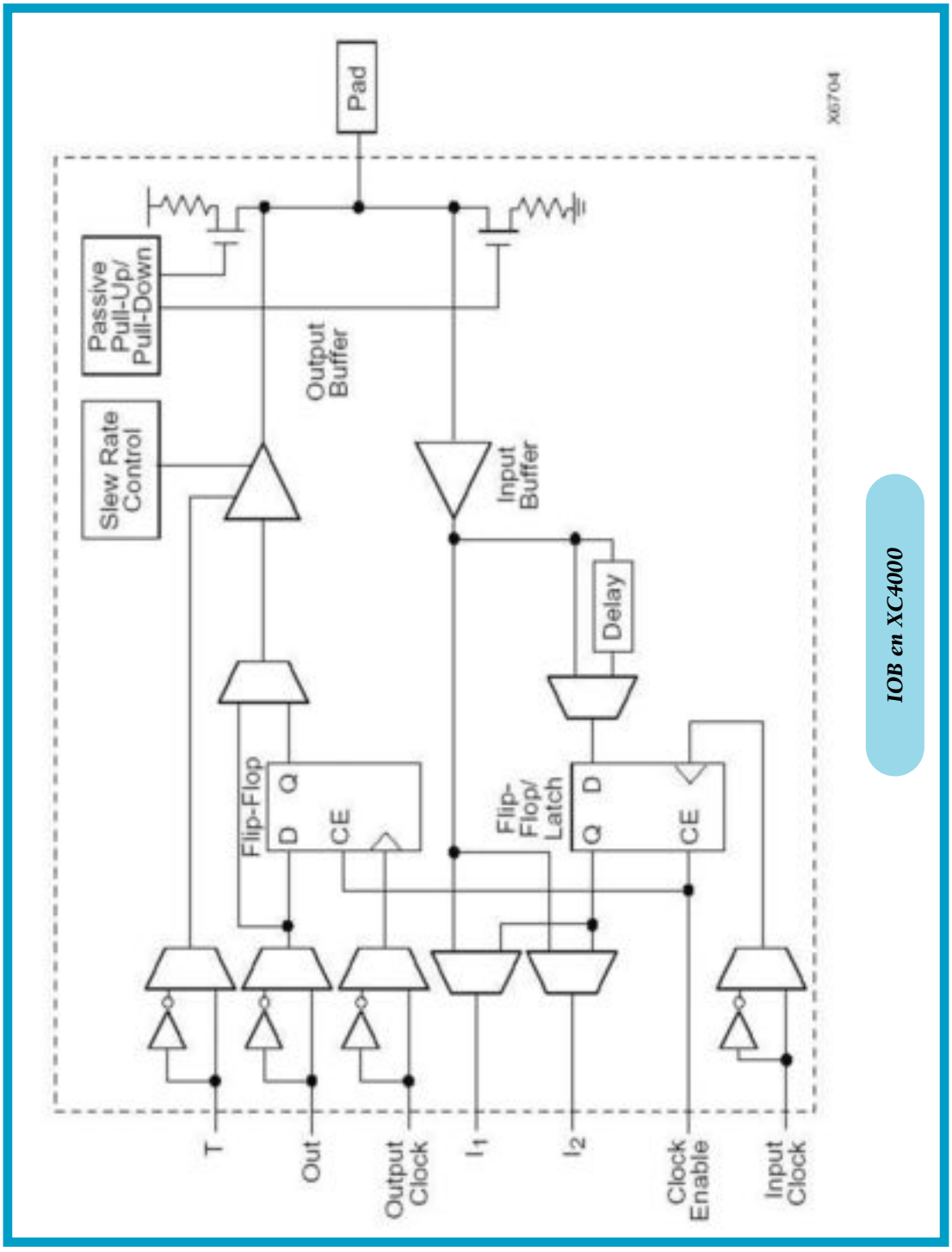
El hardware está previsto para que en cada CLB se pueda implementar una memoria de 2 x 16 bits (usando las dos LUT en paralelo) o una memoria de 1 x 32 bits (usando a las LUT en cascada).

Como en el caso de los CPLD, los bloques de entrada/salida son también configurables. Cada IOB -*Input/Output Block*; bloque de entrada/salida- está asociado a un pin físico o PAD.

El circuito es complejo y dispone de dos flip-flop, uno para programar, si es necesario, la entrada como registrada y el otro lo mismo, con la salida.

Existen otras funciones como la de programar si la salida será tipo *pull-up* (se conecta una resistencia entre la salida y +V) o *pull-down* (conectando una resistencia entre masa y la salida).

También -al igual que analizábamos antes con las CPLD- se puede programar la salida como *tri-state*, a fin de posibilitar el uso del pin en forma bidireccional (como de entrada/salida).



IOB en XC4000

Las ventajas de las FPGA son:

- Utilización de recursos de manera eficiente.
- Muy alta densidad de integración.
- Muy alto nivel de complejidad en diseños lógicos.
- Altas frecuencias de operación.
- Posibilidad de emular memoria.

Y, las desventajas de las FPGA:

- Tiempo de retardo no muy predecible, lo que lleva a diseños internos complejos.
- Requieren herramientas de software caras para optimizar diseños complejos.
- Si los dispositivos deben ser muy rápidos y complejos, y se necesitan en cantidad, la opción más viable es utilizar los ASIC.

Programación

Tanto los dispositivos más simples -como los SPLD- o los más complejos -como los FPGA- tienen una característica básica en común: su programabilidad.

Debido a la gran cantidad de celdas lógicas y matrices de interconexión que forman parte de un circuito lógico programable, es imposible que el usuario pueda realizar las interconexiones entre dichas celdas o configurar cada una de ellas en forma manual.

Es por ello que cada fabricante ofrece al usuario una serie de herramientas de software para que realice las diferentes etapas del diseño con una PLD:

- edición,
- simulación y
- programación.

La **edición** es la etapa en la cual se entra la información requerida para describir el circuito a implementar.

La **simulación** es la etapa utilizada para comprobar, mediante el análisis de un diagrama temporal, si las salidas del circuito responden adecuadamente a las excitaciones de entrada definidas por el usuario.

En dispositivos tales como CPLD y FPGA que disponen de elementos de memoria EEPROM o SRAM, la **programación** generalmente se realiza de una manera muy simple: No se requiere de un programador externo conectado a una computadora personal, como en el caso de muchos de los microprocesadores que se obtienen en el mercado; en esencia, el programador es la computadora personal y sólo se necesitan cables y mínimo hardware para conectar el PLD a la PC a través del puerto paralelo o en serie (RS-232).

Por ejemplo, la línea de SPLD y FPGA de la empresa *Altera*® utiliza sólo 5 pines del chip, para ser conectados a una PC; luego de programado, dichos pines pueden ser reutilizados por el circuito del usuario.

El hardware requerido por los CPLD y FPGA

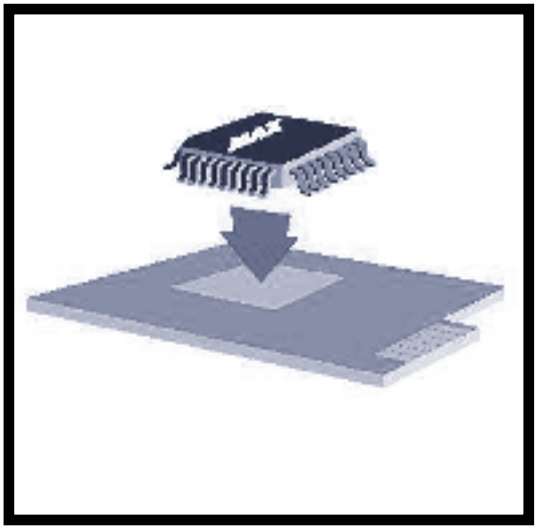
es diferente; lo explicaremos de modo diferencial a continuación

Programación de CPLD. Los CPLD suelen tener tecnología EEPROM para realizar su configuración; esto implica que, una vez programados, la información no se puede borrar ni siquiera quitando la alimentación del circuito.

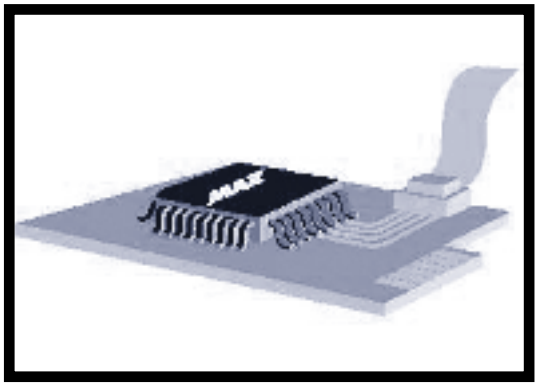
Por esto, para programarlos sólo basta emplear un cable de interconexión entre la PC y los chips.

Las siguientes figuras muestran los pasos a seguir para la programación de un chip:

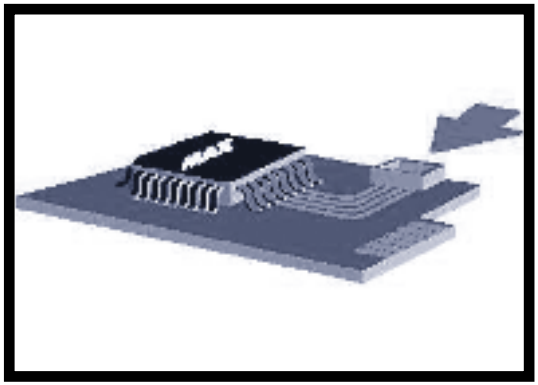
1. Se suelda el chip al circuito impreso definitivo. Generalmente, se deja un conector sólo para uso de programación.

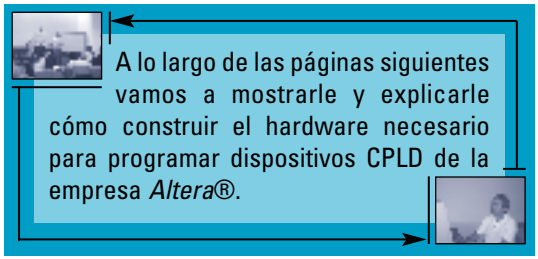


2. Se conecta el cable necesario entre la PC y el conector que está en el circuito impreso. Se procede a programar al chip.



3. Si es necesario modificar su configuración por algún error o por una actualización, se procede a repetir el paso anterior.





Programación de FPGA. Aquellas FPGA que están basadas en tecnología SRAM (memorias volátiles) tienen la desventaja de que, una vez que se quita la tensión de alimentación al chip, éste se desprograma total-

mente, queda completamente desconfigurado.

Generalmente, este tipo de FPGA se utiliza junto con otro chip que es una memoria EEPROM serie. Lo que se programa inicialmente desde la PC es esta memoria con la información necesaria para, luego, transferirse a la FPGA.

> Esta memoria, una vez programada, guarda la información en forma permanente.

En el circuito impreso, se sueldan la FPGA y la memoria, conectándolas según indica el fabricante. En general, los pines necesarios son muy pocos.

Cuando se aplica la tensión de alimentación, comienza un proceso de booteo -inicialización-, a través del que la FPGA controla la transferencia de datos desde la memoria hacia ella, hasta que esté completamente programada.

Una vez finalizado este proceso, la FPGA ya se puede usar normalmente. Esta operación suele durar algunas decenas de milisegundos[20].

---

[20] Seguramente le interesará leer:
- Wakerly, J. F. (2000; 3° ed.) *Digital Design:Principles and Practices.* Prentice Hall International.
- Tavernier, Christian (1994) *Circuitos lógicos programables*. Paraninfo. Madrid.
- Lloris Ruiz, Antonio; Espinoza, Alberto (1996) *Diseño lógico*. McGraw-Hill.
- Vyemura, John P. (2000) *Diseño de sistemas digitales*, Thomson.
- Sedra-Kenneth A.; Smith, C. (1999; 4° ed.) *Circuitos microelectrónicos*. Oxford University Press. México.
- Pearson-Prentice Hall (2003; 3° ed.) *Diseño digital*.

3. Hacia una resolución técnica

Manual de procedimientos para la construcción y el funcionamiento del equipo

El producto

El entrenador en lógica programada está diseñado sobre la base del circuito integrado EPM7128SLC84 de la firma *Altera*®.

Este chip es un CPLD -*Complex Programmable Logic Device*; dispositivo lógico programable complejo- de 128 macroceldas que trabaja con +5 V y tiene un encapsulado tipo PLCC de 84 pines.

Está formado por estos componentes:

- **Placa principal**: Aloja el CPLD con una fuente de alimentación regulada de +5 V y un oscilador a cristal de cuarzo para ser usado como fuente de reloj de referencia de alta precisión.
- **Interfaz de programación**: Circuito que sirve para configurar el dispositivo sin tener que sacarlo de la placa principal, además de protegerlo de la conexión directa con el puerto paralelo de la computadora personal.
- **Placa de entrenamiento número 1**: Ejemplo de aplicación de un contador de 2 dígitos
- **Placa de entrenamiento número 2**: Para ser implementada por los alumnos.

Para desarrollar el entrenador es necesario emplear algún dispositivo comercial.

Nuestra elección se hizo considerando que este dispositivo estuviera todavía en vigencia, se obtuviera a un precio accesible, tuviera la complejidad necesaria para poder desarrollar proyectos interesantes, y la tensión de alimentación de trabajo y el tipo de encapsulado permitieran su rápido y fácil reemplazo en caso de necesidad.

Esta última consideración es importante, ya que la mayoría de los chips actuales CPLD y FPGA incluyen encapsulados para montaje superficial, que pueden constituir una limitación importante en el armado del circuito impreso, corriendo peligro en caso de tener que reponer el chip asociado a dicha placa.

Nuestra opción no va en desmedro de dispositivos de otras marcas que puedan adquirirse en el mercado.

La empresa proveedora aconseja emplear el software *Quartus* para nuevos diseños, ya que éste soporta los dispositivos más modernos. Pero, la elección hecha aquí del software MAXPLUS-II (que todavía está en vigencia) se debe, principalmente, a que existe literatura acerca de él en Internet y, además, porque este soft funciona desde Windows 95 en adelante; en cambio, *Quartus* sólo es soportado por Windows XP y, desde el punto de vista didáctico, puede resultar más difícil de aprender.

- **Software MAXPLUS-II**: Suministrado por la empresa, permite realizar las tareas de diseño y simulación del proyecto lógico, y la programación del dispositivo.
- **Placa de entrenamiento número 3**: Ejemplo de aplicación como entrenador en lógica estándar (compuertas, multiplexores, sumadores, etc.).

El objetivo primario de este equipo es el de introducir a docentes y alumnos al mundo de la lógica programada por hardware.

El entrenador en lógica programada permite:

- Comprender la tecnología basada en dispositivos lógicos programables por hardware.
- Realizar diseños lógicos simples y complejos con el dispositivo CPLD y el software asociado a éste.
- Utilizarlo como entrenador básico de lógica digital, ya que puede emular cualquier tipo de compuerta o circuito más complejo.
- Simular el comportamiento de un circuito digital genérico en forma temporal, empleando el software asociado al circuito integrado; así se cuenta con una herramienta adicional para el análisis y síntesis de circuitos digitales.

El equipo y el marco conceptual que le hemos acercado permiten, por un lado, construir una visión local de esta tecnología, a fin de que los alumnos puedan desarrollar equipos basados en otros modelos y marcas de CPLD; y, por el otro, apropiarse de las nociones básicas de nivel general para abordar, en el futuro, implementaciones con otros dispositivos tales como los FPGA.

Los componentes

El ambiente de desarrollo de este equipo de entrenamiento en lógica programada está basado en dos partes:

1. **Hardware**: Contiene un CPLD a ser armado por los alumnos.
2. **Software**: Es suministrado por el fabricante del chip.

A partir de aquí vamos a describir tanto el hardware como el software.

1. Hardware

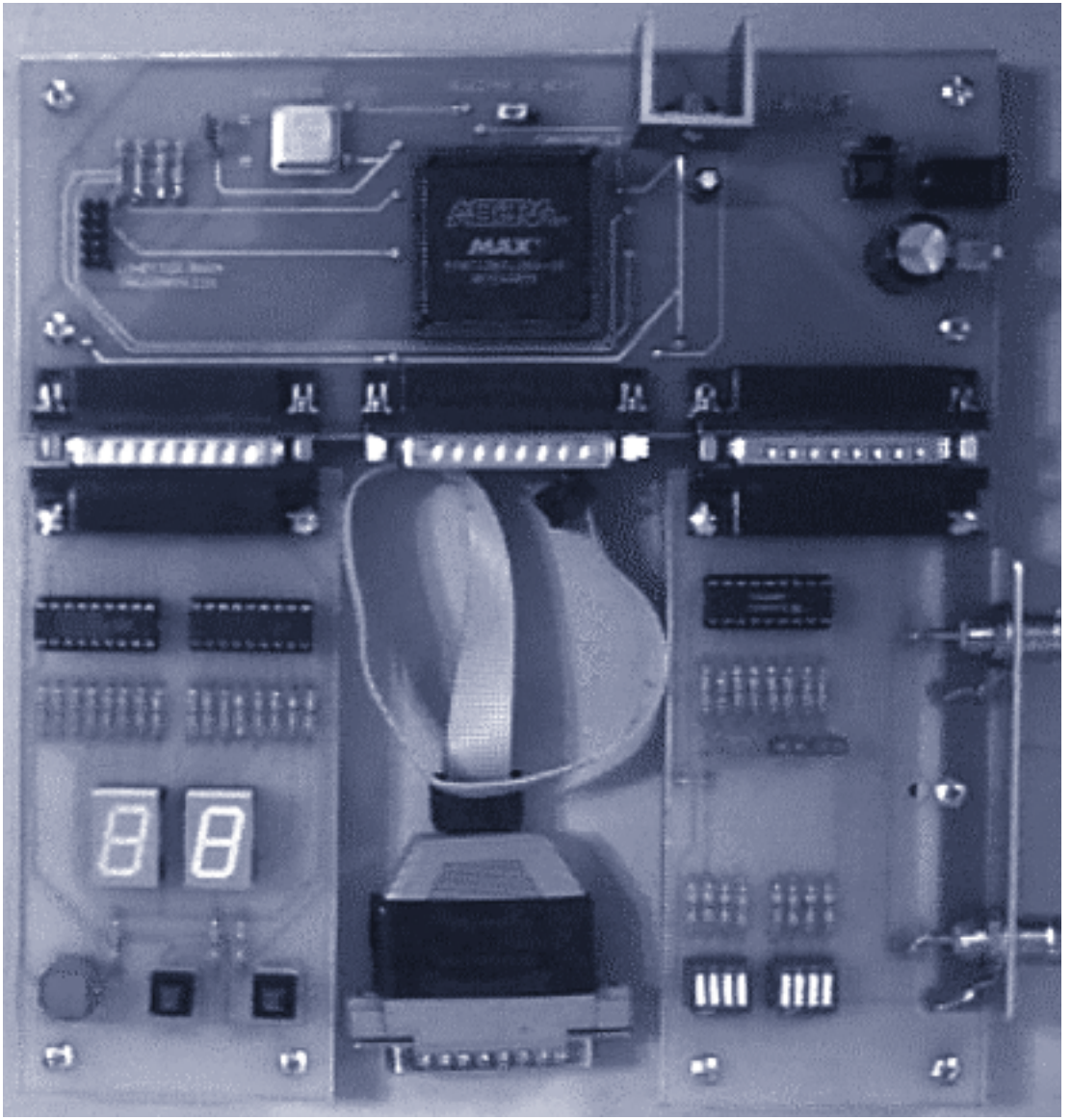
Los componentes descriptos aquí son:

- El chip EPM7128SCL84.
- La placa principal.
- La interfaz de programación.
- La placa de experimentación número 1.
- La placa de experimentación número 2.
- La placa de experimentación número 3.

El chip. El equipo está basado en el circuito integrado EPM7128 que corresponde a la familia de dispositivos lógicos programables tipo CPLD de la empresa *Altera*®.

El modelo seleccionado es el EPM7128SCL84-15:

- La sigla EPM significa: Familia MAX de dispositivos PLD basados en EEPROM.



- El número 7 indica: Familia MAX7000
- El número 128 indica: Modelo y número de macroceldas que posee[21].
- La letra S indica: Programación ISP (permite programar directamente al dispositivo desde una PC, sin uso de progra-

[21] Los modelos incluidos en la familia MAX7000 parten desde 32 macroceldas (la MAX7032) hasta 256 macroceldas (la MAX7256), pasando por 64, 96, 128, 160 y 192 macroceldas.

mador alguno)[22].

- La sigla CL significa: Encapsulado PLCC[23]
- El número 84: 84 pines.
- El número 15: 15 nanosegundos de retardo en una conexión directa entre un pin configurado como entrada y otro pin configurado como salida[24].

El EPM7128 tiene 128 macroceldas que están distribuidas en el chip en 8 bloques LAB de 16 macroceldas cada uno.

Cada bloque LAB -*Logic Array Block*; o bloque de arreglo lógico- tiene comunicación con una matriz de interconexión denominada PIA -*Programmable Interconnect Array*; arreglo de interconexión programable- que es la encargada de interconectar a los LAB y, por lo tanto, a las macroceldas entre sí.

Los 84 pines que posee el chip se distribuyen en:

- 60 pines de entrada/salida, seleccionables por el usuario.
- Pines dedicados de entrada para programación y uso posterior, para el usuario.
- Pines dedicados para control global interno o uso como entradas, para el usuario.
- 16 pines de alimentación.

De todos ellos, sólo 2 no han sido utilizados, dado que no se disponía de más pines en los conectores DB25 y que se decidió reforzar las señales de alimentación Vcc y GND en ellos.

De todas maneras, los 66 pines disponibles son más que suficientes para la mayoría de los proyectos didácticos que se puedan realizar.

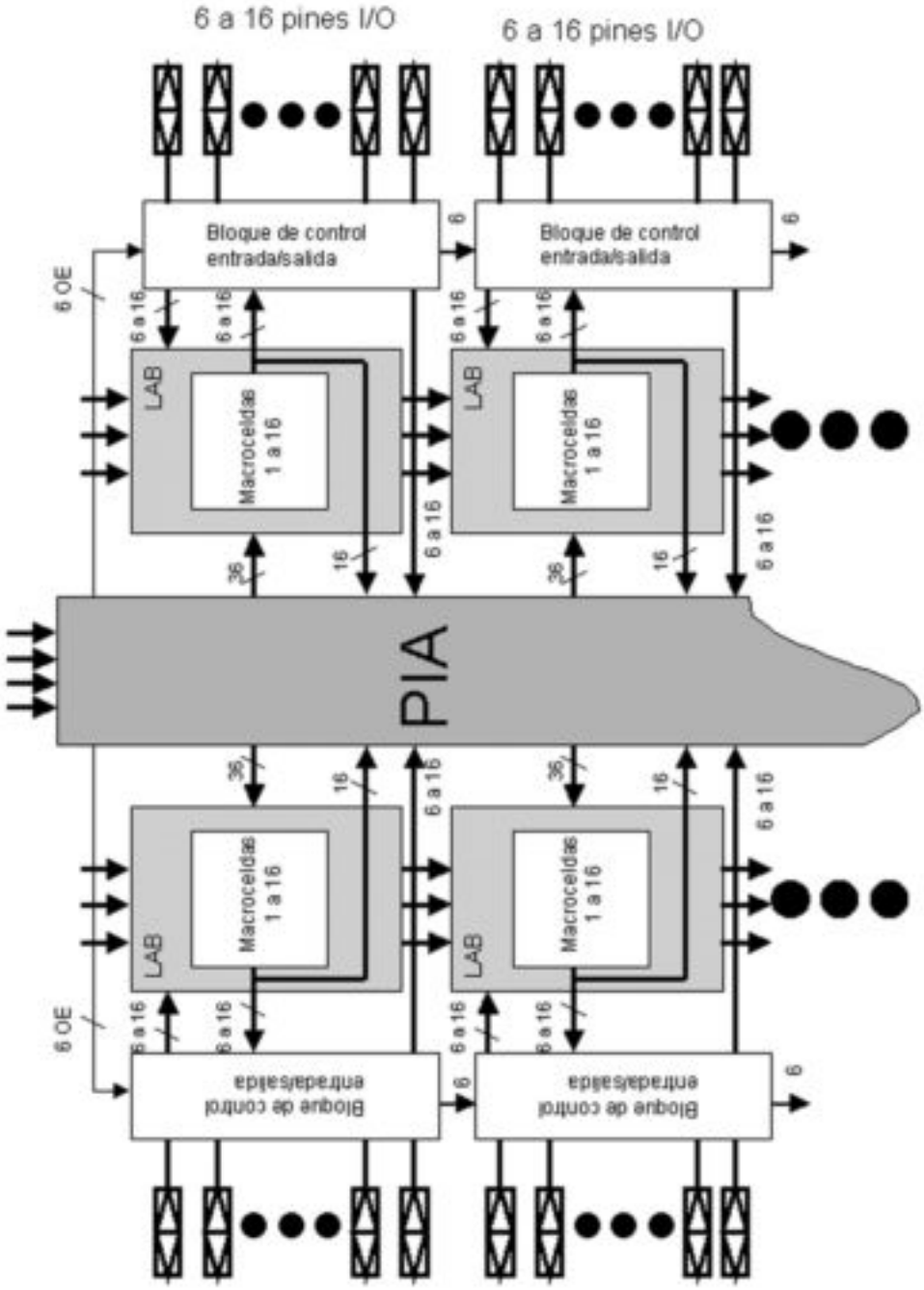
Pines relevantes en el EPM7128 son:

- Pin 1 (INPUT/GCLRn): Programable como entrada simple o entrada para clear global negada.
- Pin 2 (INPUT/OE2/GCLK2): Programable como entrada simple, entrada de control *tri-state* o entrada de reloj

[22] Este dispositivo tiene implementada la función ISP -*In-System Programmabiliy*; programable dentro del sistema-, lo que significa que posee internamente los circuitos necesarios para poder programar las celdas de memoria EEPROM que configuran los componentes internos del chip (multiplexores, matriz de interconexión, etc.) sin necesidad de hardware adicional (programador). La norma que se emplea para ello es la IEEE Standard 1149.1 (1990) redactada por el IEEE -*Institute of Electrical and Electronic Engineering*; Instituto de Ingeniería Eléctrica y Electrónica-; ésta, más conocida como JTAG -*Joint Test Action Group*- es una interfaz de 4 pines con la cual es posible programar desde un procesador host (por ejemplo, una PC) a cualquier dispositivo lógico, ya sea un PLD o un microprocesador que trabaje con esta norma. La familia MAX7000S, como la mayoría de los chips CPLD y FPGA de la empresa por la que hemos optado, dispone de esta interfaz. La EPM7128S tiene 4 pines denominados: TCK, TDI, TDO y TMS, que son los que se deben conectar a la PC a través del puerto paralelo.

[23] Los modelos hasta EPM7128 vienen con la opción de encapsulado PLCC: a partir de allí, todo es montaje superficial.

[24] El número después del guión da información sobre la velocidad del dispositivo; indica el tiempo de retardo que se puede obtener a la salida de un pin configurado como salida, excitando con una señal desde otro pin configurado como entrada, si se programa el chip de tal forma que se conecte internamente a ambos pines sin lógica en medio. Cuanto más chico es dicho número, más rápido es el chip internamente.



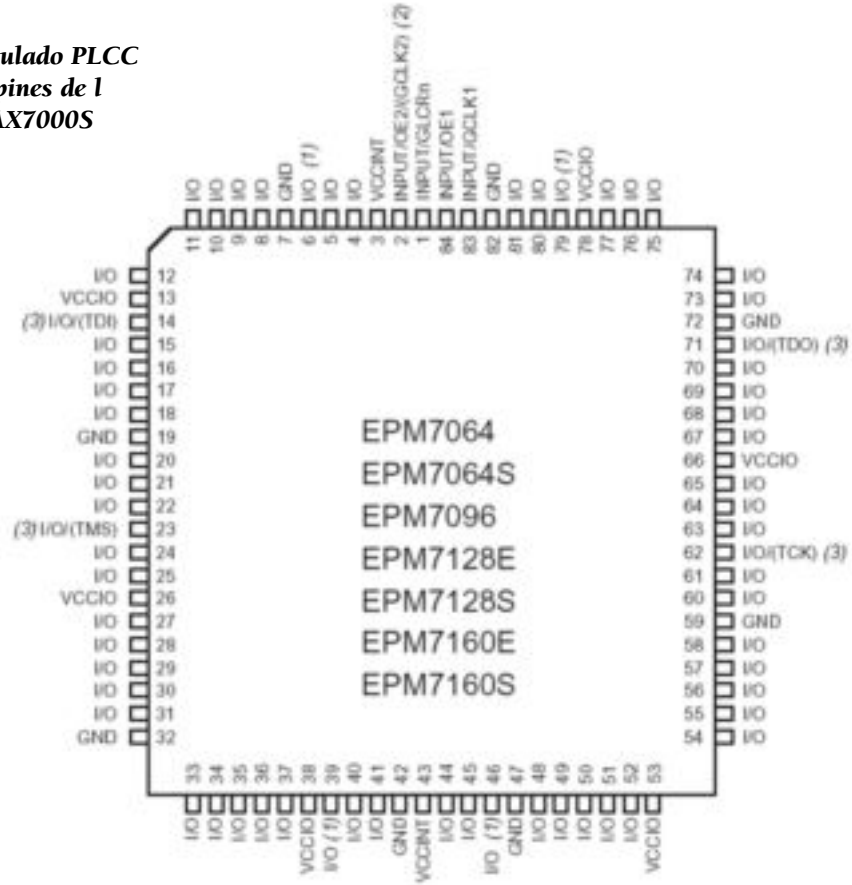
Esquema general de la serie MAX7000S

global 2.

- Pines 14 (I-O/TDI), 23(I-O/TMS), 62(I-O/TCK) y 71(I-O/TDO): Programables como entrada/salida o para programación del dispositivo.
- Pin 83 (INPUT/GCLK1): Programable como entrada simple o reloj global 1.
- Pin 84 (INPUT/OE1): Programable como entrada simple o entrada de control tri-state.
- Pines 3, 13, 26, 38, 43, 53, 66 y 78: Corresponden a tensión de alimentación positiva (Vcc = +5 V).
- Pines 7, 19, 32, 42, 47, 59, 72 y 82: Corresponden a tensión de alimentación 0 V (GND).
- Demás pines. Programables como entrada, salida o entrada/salida, por el usuario.

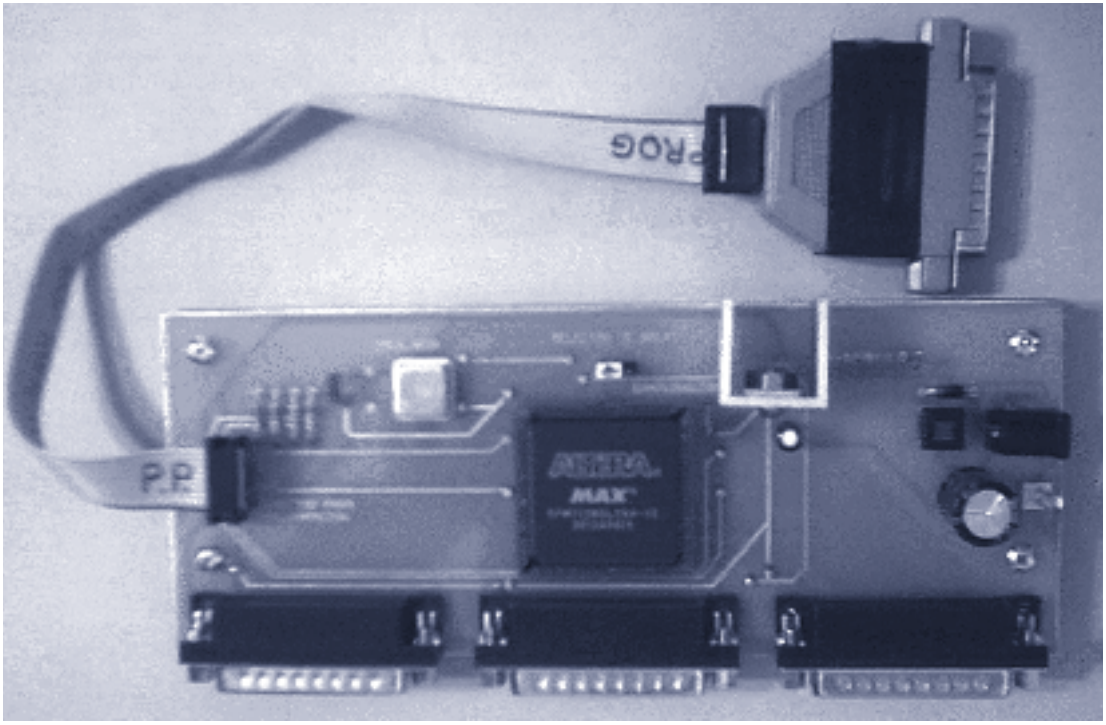
En caso de que usted y sus alumnos quieran diseñar otro impreso empleando este chip, deben conectar todos los pines de alimentación indicados. Éstos no son redundantes, ya que conectan distintas partes internas del chip que deben recibir las diferentes tensiones de alimentación.

Encapsulado PLCC 84 pines de l MAX7000S

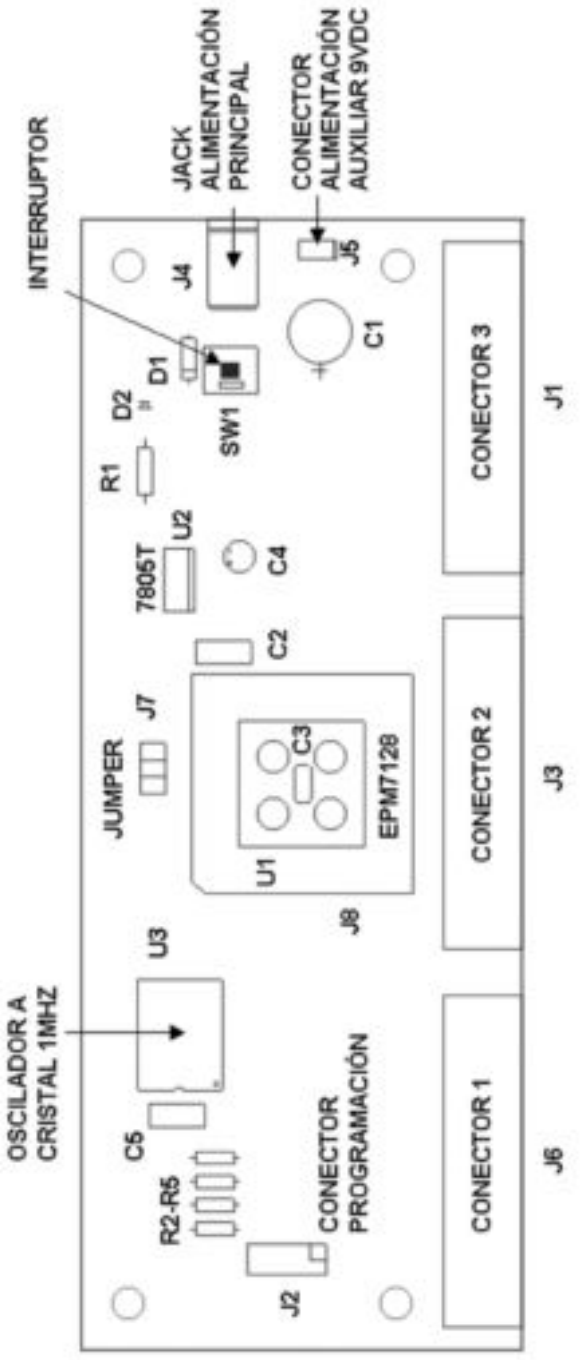


La placa principal. Esta placa de circuito impreso tiene por finalidad:

- Alojar al circuito integrado EPM7128 compatible con el encapsulado PLCC, a fin de permitir su posible reemplazo.
- Dar alimentación con tensión regulada a éste y a los circuitos alojados en las placas de experimentación.
- Interconectar el chip de lógica programada con la PC, para su programación; para esto, se dispone de un conector específico.
- Interconectar el chip con hasta 3 placas de experimentación, para la realización de múltiples proyectos de diseños lógicos.
- Alojar al oscilador de cristal de cuarzo, para generar señales de reloj de alta precisión y estabilidad temporal al chip de lógica programada.



Placa principal junto con la interfaz de programación



Esquema de la placa indicando cada componente

U1. CPLD modelo EPM7128SCL84.

U2. Regulador de tensión fija de +5 V LM7805T; suministra tensión de +5 V hasta 1 A de corriente.

U3. Oscilador a cristal de cuarzo de 4 MHz JITO, compatible con señales TTL.

J2. Conector de programación tipo IDC10 para cable plano de 2 x 5 contactos.

J4. Conector para entrada de alimentación principal de +12 VDC, tipo plug de 3,5 mm con corte.

J5. Conector para entrada auxiliar de alimentación, para conectar una batería de +9 VDC, para el caso de no disponer de una fuente externa de alimentación de 220 VAC a +12 VDC.

J6. Conector 1 para conexión con placa de experimentación número 1, tipo DB25 macho a 90° para impreso.

J3. Conector 2 para conexión con placa de experimentación número 2, tipo DB25 macho a 90° para impreso.

J1. Conector 3 para conexión con placa de experimentación número 3, tipo DB25 macho a 90° para impreso.

D1. Diodo de protección contra inversión de polaridad.

D2. Lámpara para indicación de alimentación, tipo led, 3 mm color rojo.

SW1. Interruptor tipo llave inversora doble marca SIPI o similar.

R1. Resistencia limitadora de corriente del led, 220 ohm, ¼ W.

R2-R5. Resistencia de pull-up para pines de programación del CPLD, 1K, 1/4W.

J7. Selector de fuente de reloj global al CPLD; tira de pines de 1 x 3 y jumper.

J8. Zócalo PLCC de 84 pines para alojar al CPLD.

C1. Capacitor de filtrado de alimentación; 1000 µF / 25 V.

C2. Capacitor de filtrado para fuente de tensión; 100 nF / 16 V.

C3. Capacitor de filtrado para CPLD; 100 nF / 16 V.

C4. Capacitor de filtrado para CPLD; 10 µF / 16 V.

C5. Capacitor de filtrado para el oscilador a cristal.

Varios: Disipador para el regulador de tensión U2, de aluminio tipo "U" de 20 x 20 x 20 mm. Fuente de alimentación 220 VAC - 12 VDC 1 ampere con terminación en ficha plug 3,5 mm con el terminal positivo en el centro.

Como mencionamos, el EPM7128 tiene, en total, 68 pines a disposición del usuario.

En el diseño del entrenador, contemplamos la inclusión de conectores tipo DB25; empleando 3 de ellos se tiene un total de 75 conexiones posibles.

Dado que la placa principal brinda la tensión de alimentación de +5 V y masa (GND), para simplificar el diseño de las placas de experimentación, algunos pines son utilizados para llevar dichas señales; éstos son los designados como 1, 13, 14 y 25 en cada uno de los conectores 3 (J1), 2 (J3) y 1 (J6).

Los pines 1 y 14 son de masa (GND), mientras que los pines 13 y 25 son de +5 V.

Cada conector lleva un conjunto diferente de pines de entrada/salida desde el EPM7128 y no se repiten.

Las excepciones son:

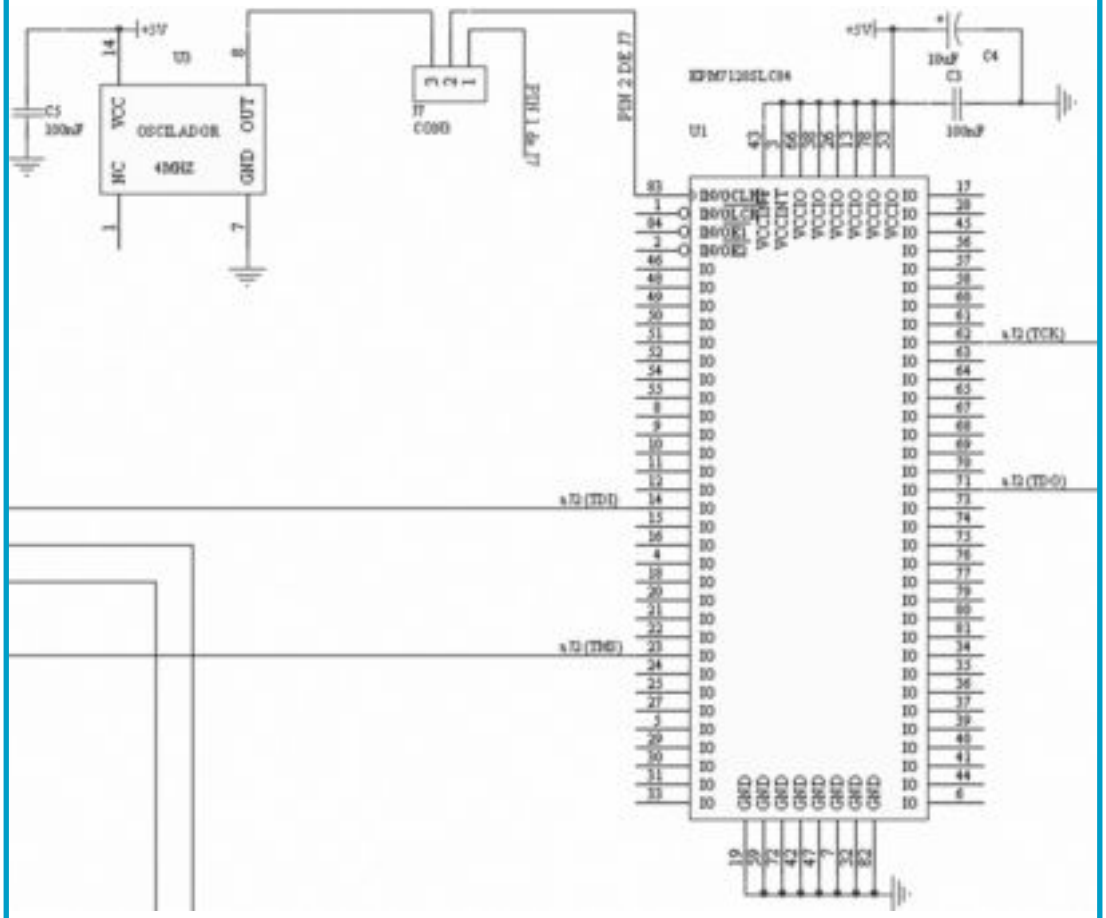
- Pin 1 del conector J7 que se conecta tanto al conector 1 (al pin 17) como al conector 2 (al pin 17). Si el jumper J7 lo habilita, este pin permite que una fuente de reloj externo a la placa principal entre al reloj global de la EPM7128 por el pin 83.
- Pin 2 del EPM7128 que se conecta tanto al conector 1 como al conector 2. Este pin puede ser configurado como entrada simple (I), de control para control de tri-state de buffers internos (OE2) o como una entrada adicional de reloj global (GCLK2).

Tabla de conexiones Conector 1

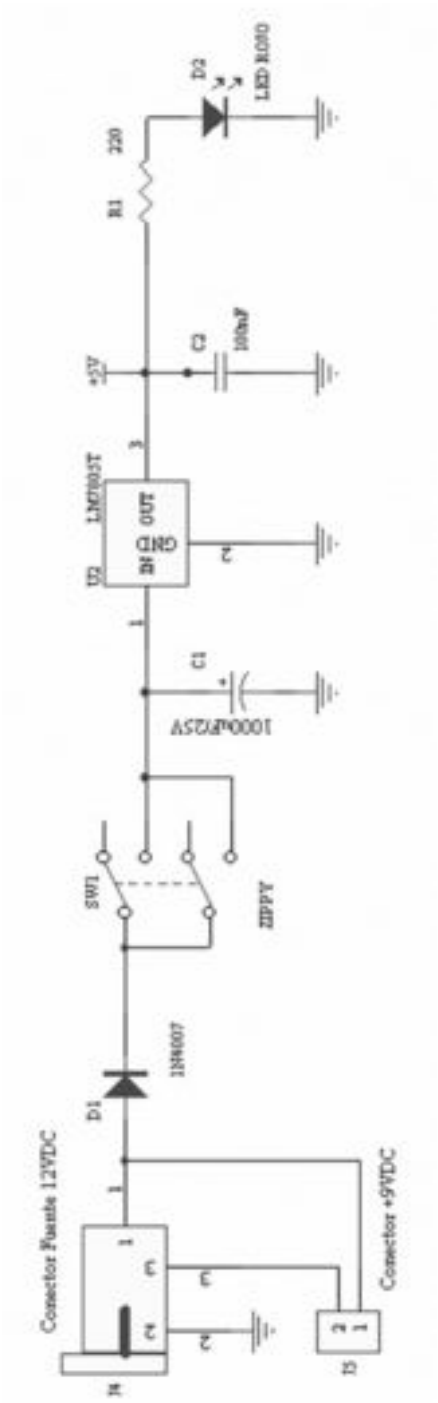
PIN	Dispositivo	
1.		GND
2.	77	EPM7128
3.	80	EPM7128
4.	84	EPM7128
5.	1	EPM7128
6.	4	EPM7128
7.	6	EPM7128
8.	9	EPM7128
9.	10	EPM7128
10.	15	EPM7128
11.	17	EPM7128
12.	20	EPM7128
13.		VCC
14.		GND
15.	79	EPM7128
16.	81	EPM7128
17.	1	ConectorJ7
18.	2	EPM7128
19.	5	EPM7128
20.	8	EPM7128
21.	11	EPM7128
22.	12	EPM7128
23.	16	EPM7128
24.	18	EPM7128
25.		VCC

Tabla de conexiones Conector 2		
PIN	Dispositivo	
1.		GND
2.	25	EPM7128
3.	27	EPM7128
4.	28	EPM7128
5.	29	EPM7128
6.	31	EPM7128
7.	33	EPM7128
8.	35	EPM7128
9.	39	EPM7128
10.	41	EPM7128
11.	45	EPM7128
12.	2	EPM7128
13.		VCC
14.		GND
15.	22	EPM7128
16.	21	EPM7128
17.	1	ConectorJ7
18.	30	EPM7128
19.	34	EPM7128
20.	36	EPM7128
21.	37	EPM7128
22.	40	EPM7128
23.	44	EPM7128
24.	46	EPM7128
25.		VCC

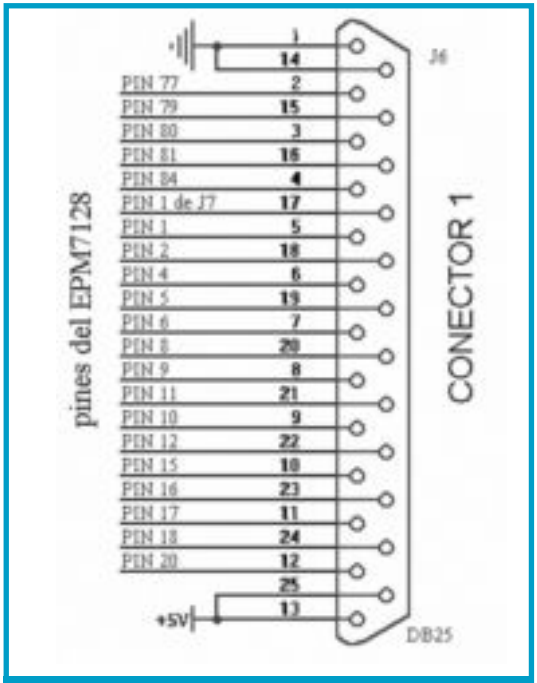
Tabla de conexiones Conector 3		
PIN	Dispositivo	
1.		GND
2.	50	EPM7128
3.	52	EPM7128
4.	54	EPM7128
5.	56	EPM7128
6.	58	EPM7128
7.	60	EPM7128
8.	65	EPM7128
9.	69	EPM7128
10.	70	EPM7128
11.	74	EPM7128
12.	75	EPM7128
13.		VCC
14.		GND
15.	49	EPM7128
16.	51	EPM7128
17.	55	ConectorJ7
18.	57	EPM7128
19.	61	EPM7128
20.	63	EPM7128
21.	64	EPM7128
22.	68	EPM7128
23.	73	EPM7128
24.	76	EPM7128
25.		VCC



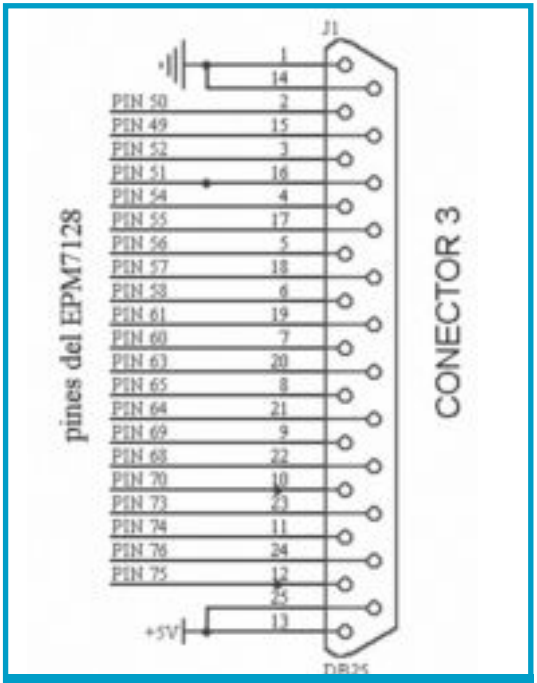
Esquemático del chip junto con el jumper selector de fuente de reloj y oscilador a cristal



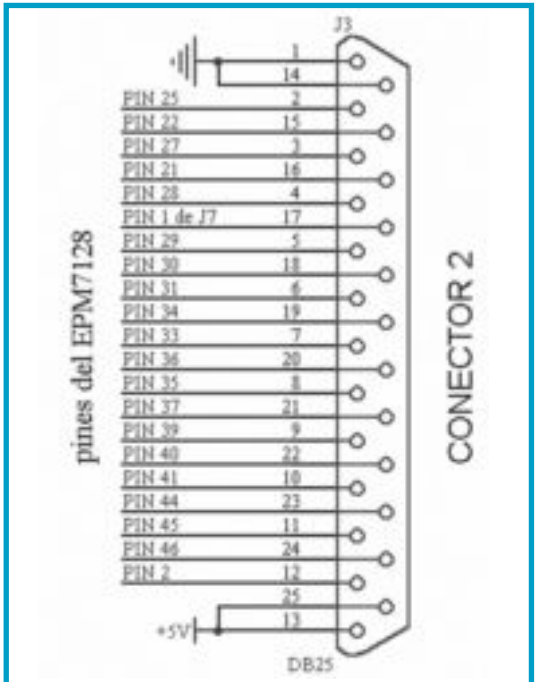
Esquemático de la fuente de alimentación



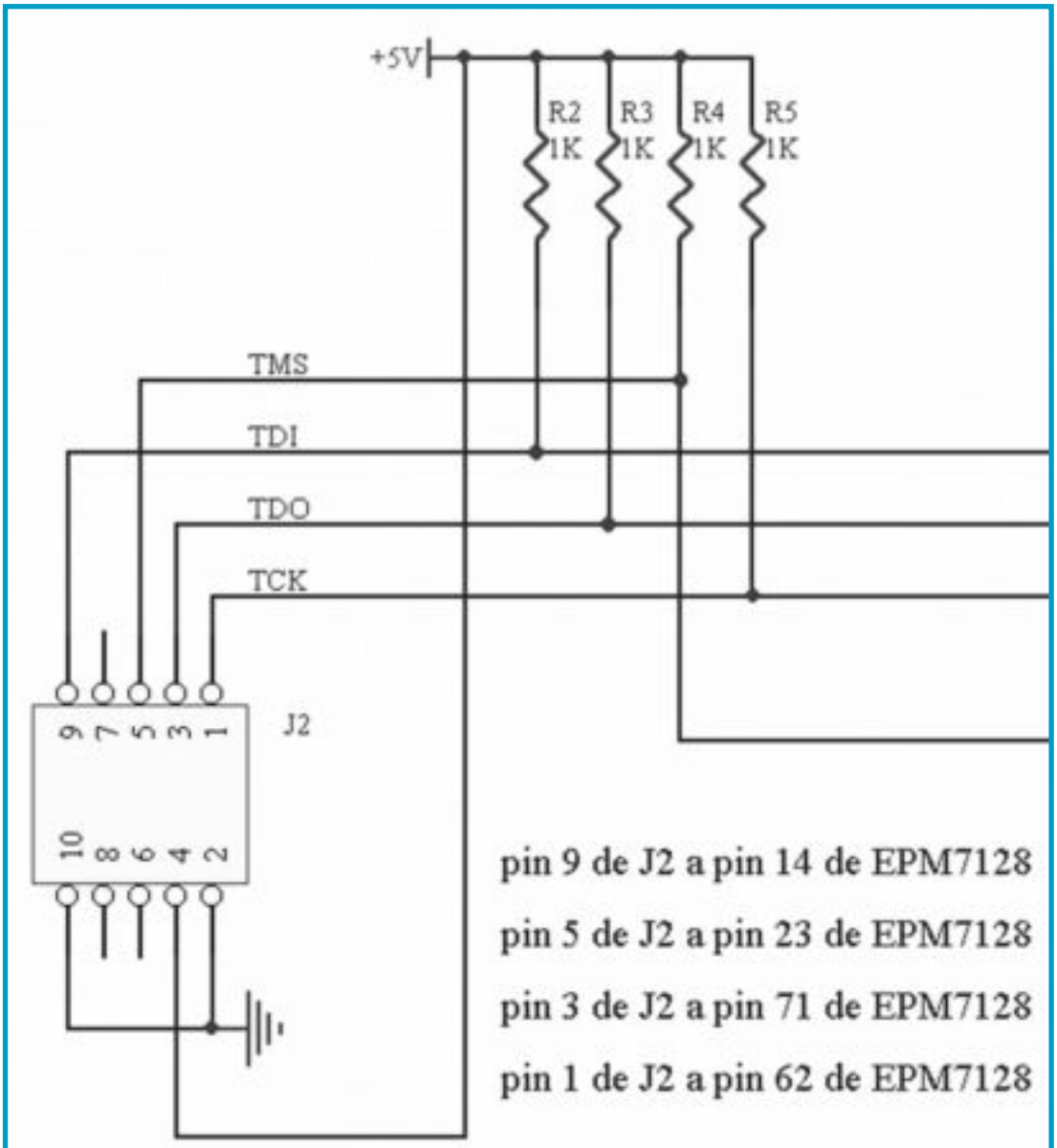
Esquemático del conector 1



Esquemático del conector 3



Esquemático del conector 2



Esquemático del conector para la interfaz de programación

La interfaz de programación. La tecnología empleada por los CPLD de la línea MAX7000 para almacenar la configuración de los chips una vez programados, es la EEPROM.

Los elementos de memoria fabricados con esta tecnología deben ser programados aplicando tensiones superiores a los 5 V (típicamente, alrededor de 12 V).

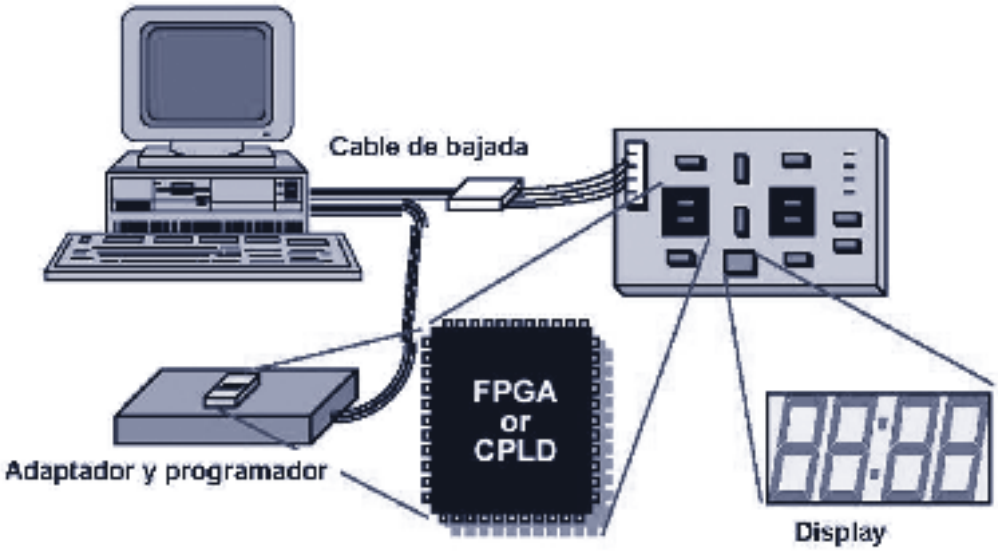
Existen dispositivos que, necesariamente, deben conectarse a los programadores para que éstos apliquen los niveles apropiados de tensiones de programación; otros, en cambio, como los de la línea MAX7000S, poseen electrónica interna que les permite generar dichas tensiones a fin de poder programar a sus elementos de memoria interna; este rasgo ofrece una gran ventaja, ya que no es necesario disponer de programador alguno.

En la siguiente figura vemos un ejemplo que sintetiza ambas opciones:

- Usar un programador conectado a la PC donde se aloja el CPLD o FPGA para su configuración y, luego de programado, conectarlo al impreso definitivo.
- Con la ayuda del cable de bajada, interconectar la PC directamente al circuito impreso donde está alojado el CPLD o FPGA -previo paso de haberlo soldado a él-.

Esta operación se realiza con sólo algunos cables (típicamente, 4) que se conectan a los pines de programación del chip. Una vez programado el chip, estos pines de programación pueden ser utilizados por el usuario en el circuito impreso.

> Esta última opción es la denominada ISP - *In-System Programmability*-.



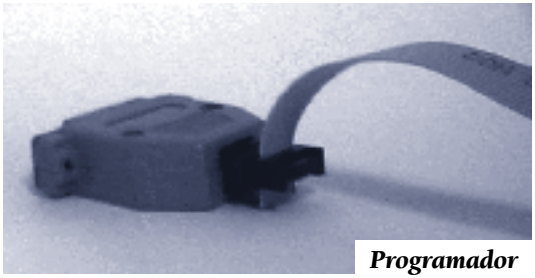
Esquema de programación en el sistema ISP

Para poder realizarlo, sólo son necesarios cuatro pines de la EPM7128S:

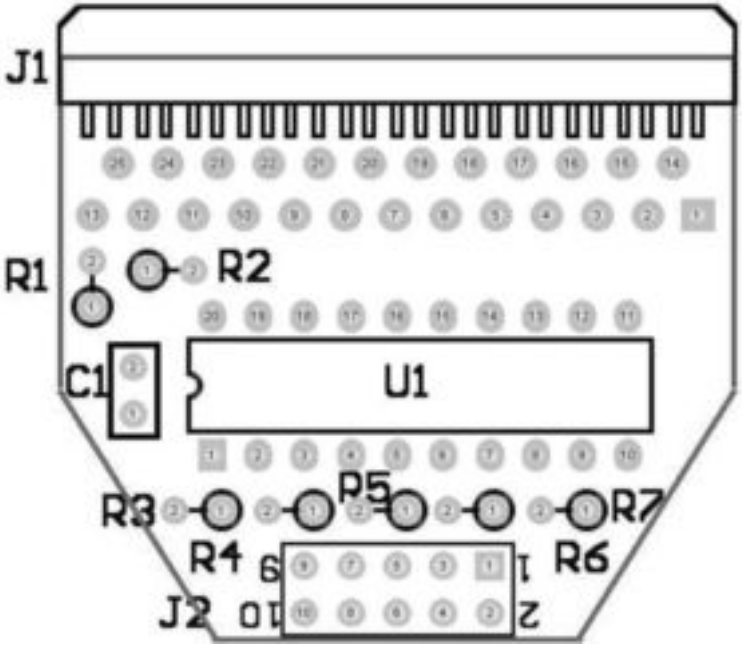
- TDI (pin 14)
- TDO (pin 71)
- TCK (pin 62)
- TMS (pin 23)

Cada uno se puede conectar directamente al puerto paralelo de una computadora personal. Sólo es necesario conectar -entre cada una de estas líneas y los +5 V de tensión de alimentación- una resistencia de 1 kΩ; estas resistencias están incluidas en la placa principal.

No obstante esto, la empresa proveedora recomienda interponer, entre el CPLD y el conector del puerto paralelo de la PC, un óctuplo *buffer* no inversor con tri-state tal como el 74LS244 y 7 resistencias asociadas. Esto es debido a que en el puerto pueden generarse niveles de tensión que perjudiquen las líneas de programación del CPLD.

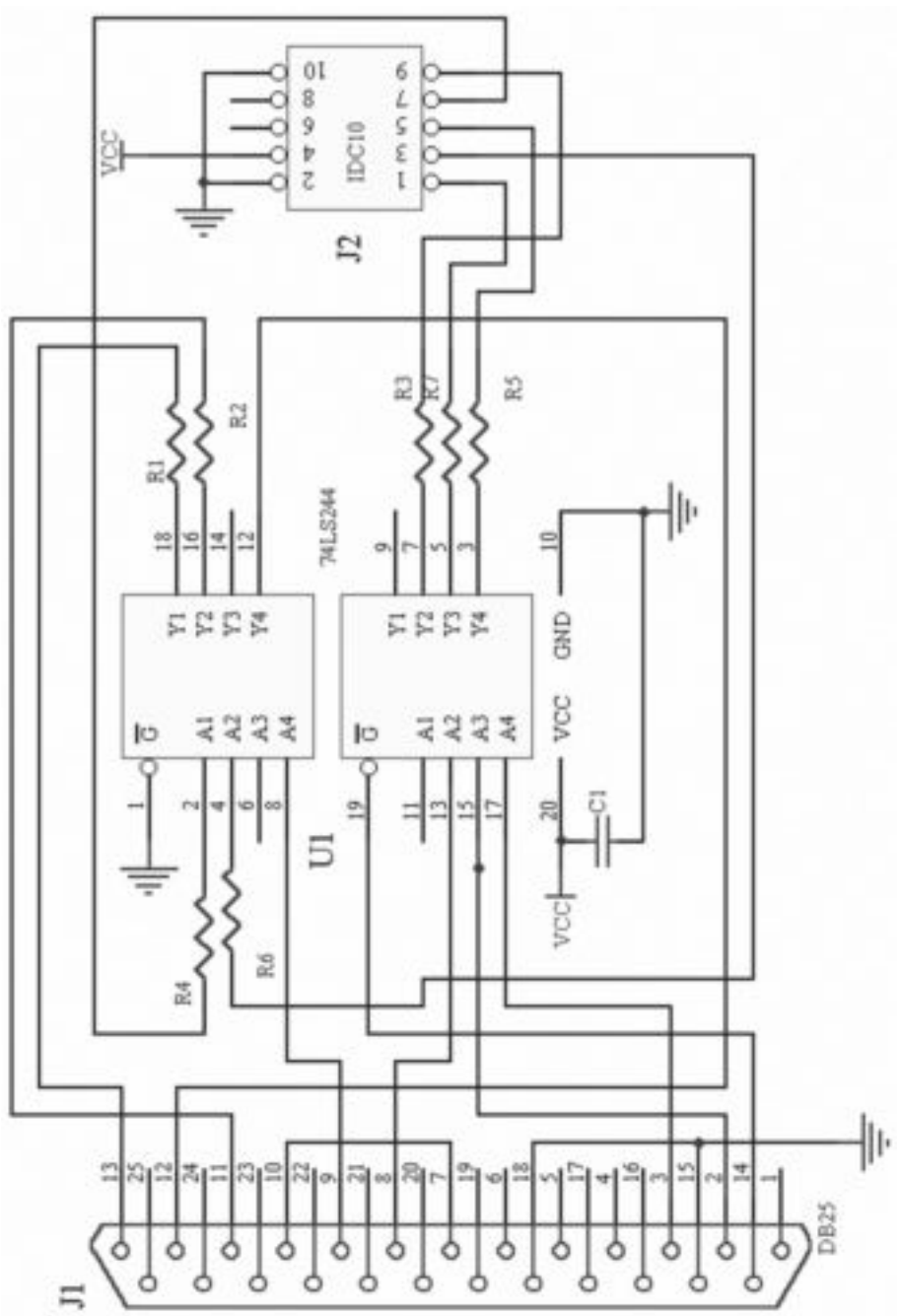


Programador



Disposición de componentes en el circuito

U1. Óctuplo buffer no inversor tipo 74LS244.
C1. Capacitor 100 nF / 16 V.
J1. Conector DB25 macho para cable.
J2. Conector para cable plano 2 x 5 tipo IDC10.
R1-R7. Resistencia de 33 ohm de ¼ W.
Varios: Cable plano de 10 cables. Conector hembra para cable plano de 10 (lado DB25). Conector hembra para cable plano de 10 (lado placa principal). Tapas para conector DB25. Tira doble de pines a 90° compatible con conector IDC10.

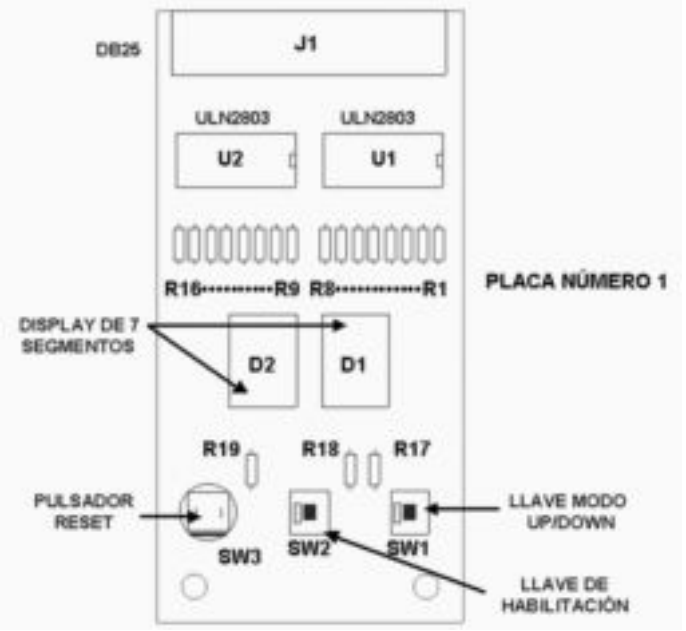


Esquemático del programador

La placa de experimentación número 1. Esta placa consta de dos buffers de corriente ULN2803 (U1 y U2) con capacidad para manejar hasta 500 mA de corriente cada uno.

Los buffers se usan para poder excitar a cada uno de los 2 displays de 7 segmentos (D1 y D2) que están en la placa. Esto es necesario, ya que la mayoría de los circuitos lógicos programables no posee la capacidad de corriente suficiente para excitar un led ellos mismos.

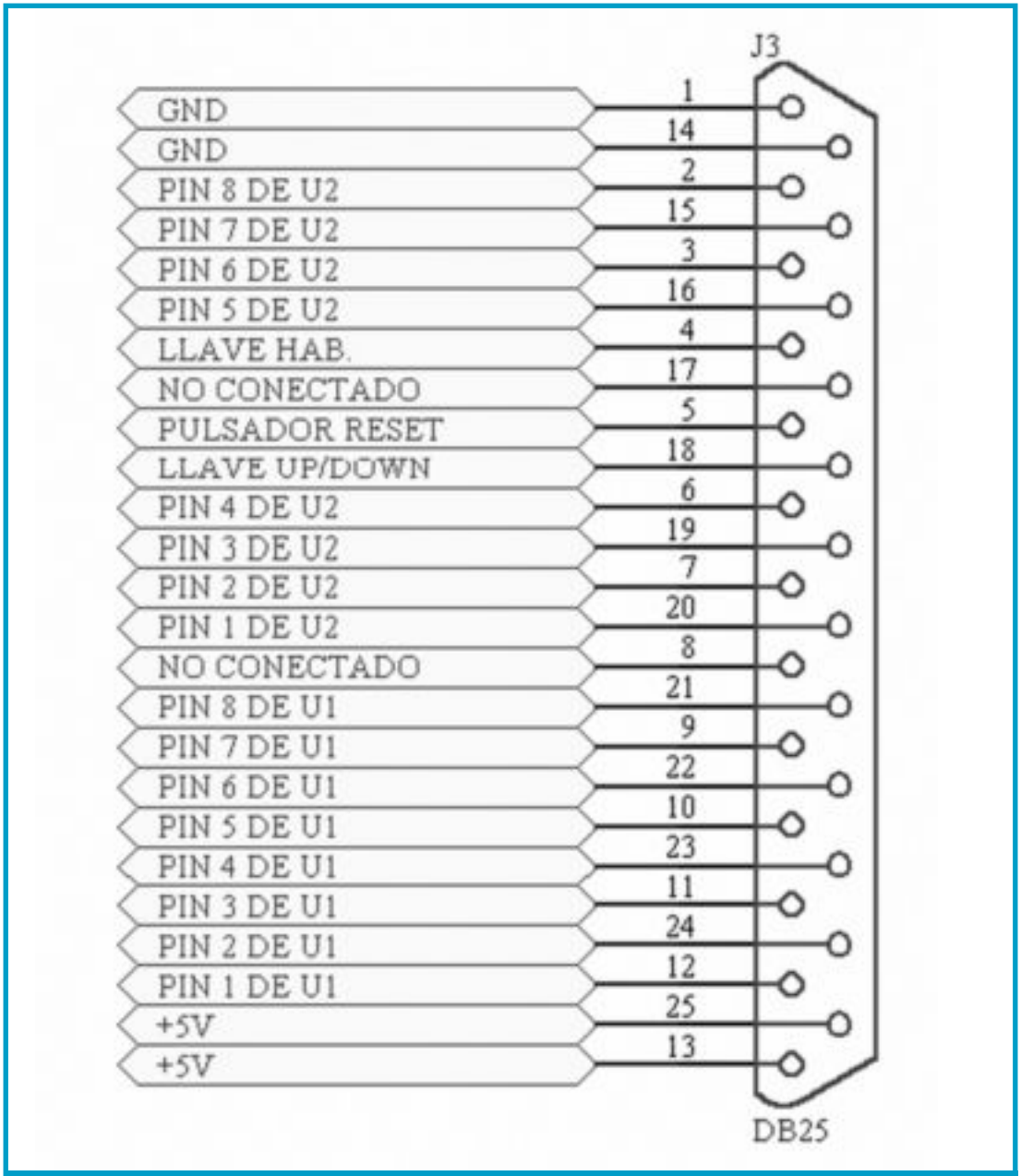
Además hay dos llaves doble inversoras (SW1 y SW2) y un pulsador (SW3) sin retención, a fin de proveer diversas entradas manuales al chip.



Disposición de componentes

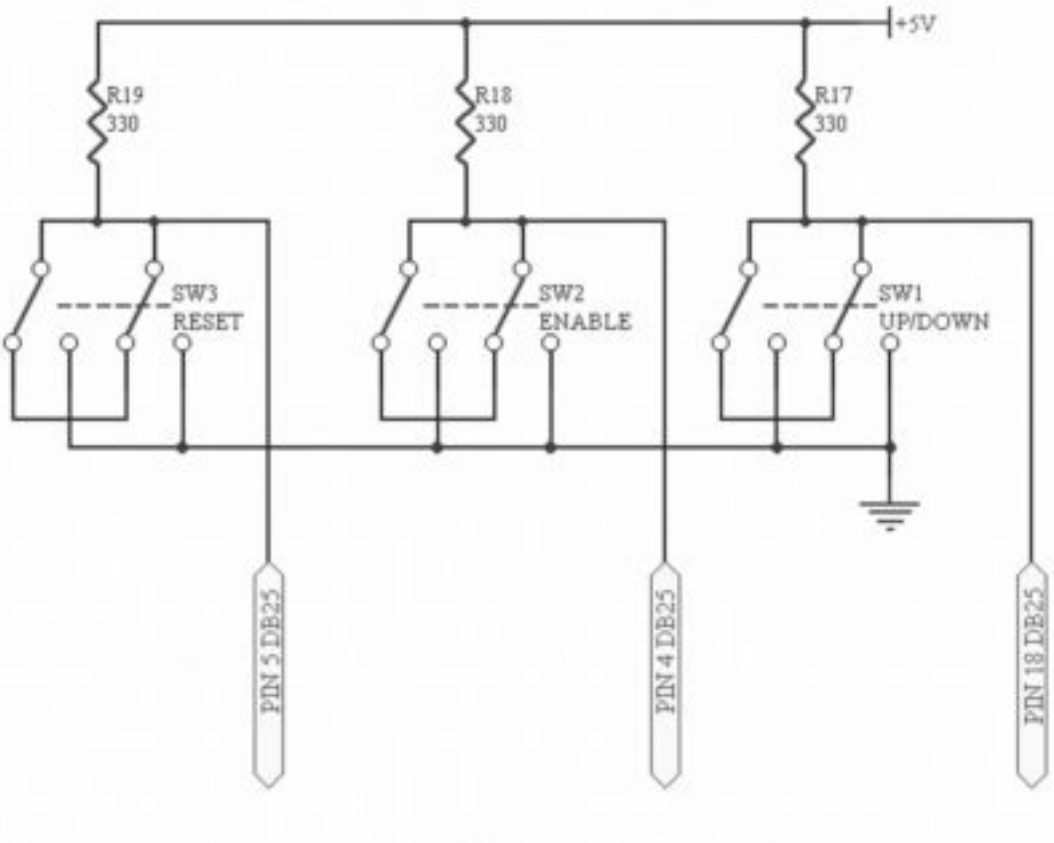
U1. Óctuplo buffer inversor tipo ULN2803 que comanda a display D1.
U2. Óctuplo buffer inversor tipo ULN2803 que comanda a display D2.
D1. Display de 7 segmentos ánodo común tipo TDSR5150 o TDSO5150.
D2. Display de 7 segmentos ánodo común tipo TDSR5150 o TDSO5150.
J1. Conector DB25 hembra a 90 grados para impreso.
SW1. Llave doble inversora, para uso como selector de modo de conteo up/down.
SW2. Llave doble inversora, para uso como habilitación de reloj.
SW3. Pulsador doble normal abierto, para uso como RESET.
R1-R8. Resistencia de 330 ohm de ¼ W limitadoras de corriente, para display D1.
R9-R16. Resistencia de 330 ohm de ¼ W limitadoras de corriente, para display D2.
R17-R19. Resistencia de 4K7 de ¼ W limitadora de corriente, para SW1, SW2 y SW3 respectivamente.

El conector DB25 hembra (J1) es el que interconecta esta placa con la principal. De esta manera, el EPM7128 puede excitar a los displays vía los ULN2803 y, además, usar entradas provenientes de las 2 llaves y el pulsador.



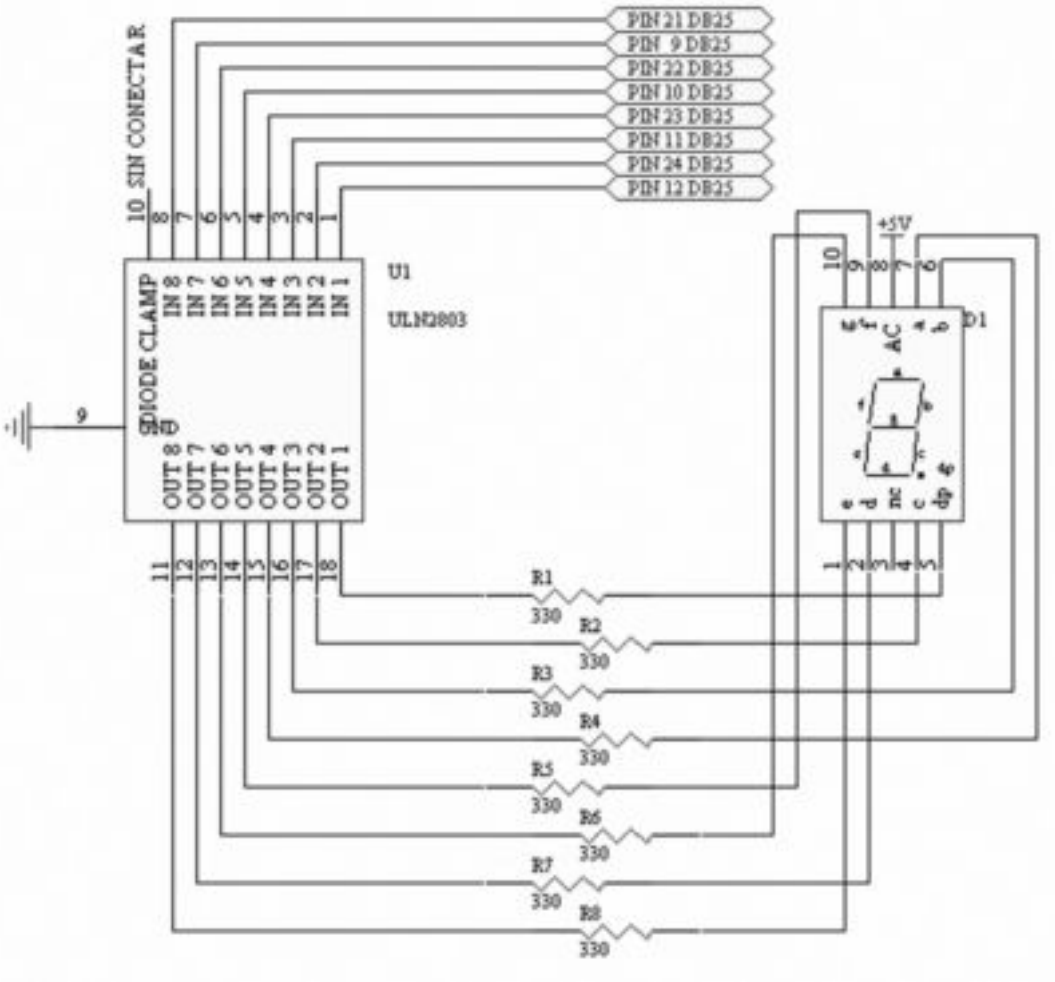
Esquemático mostrando el conector DB25

El pulsador se conecta al pin de global clear (GCLR1n) del EPM7128, teniendo un acceso directo para poder resetear a los flip-flops internos.

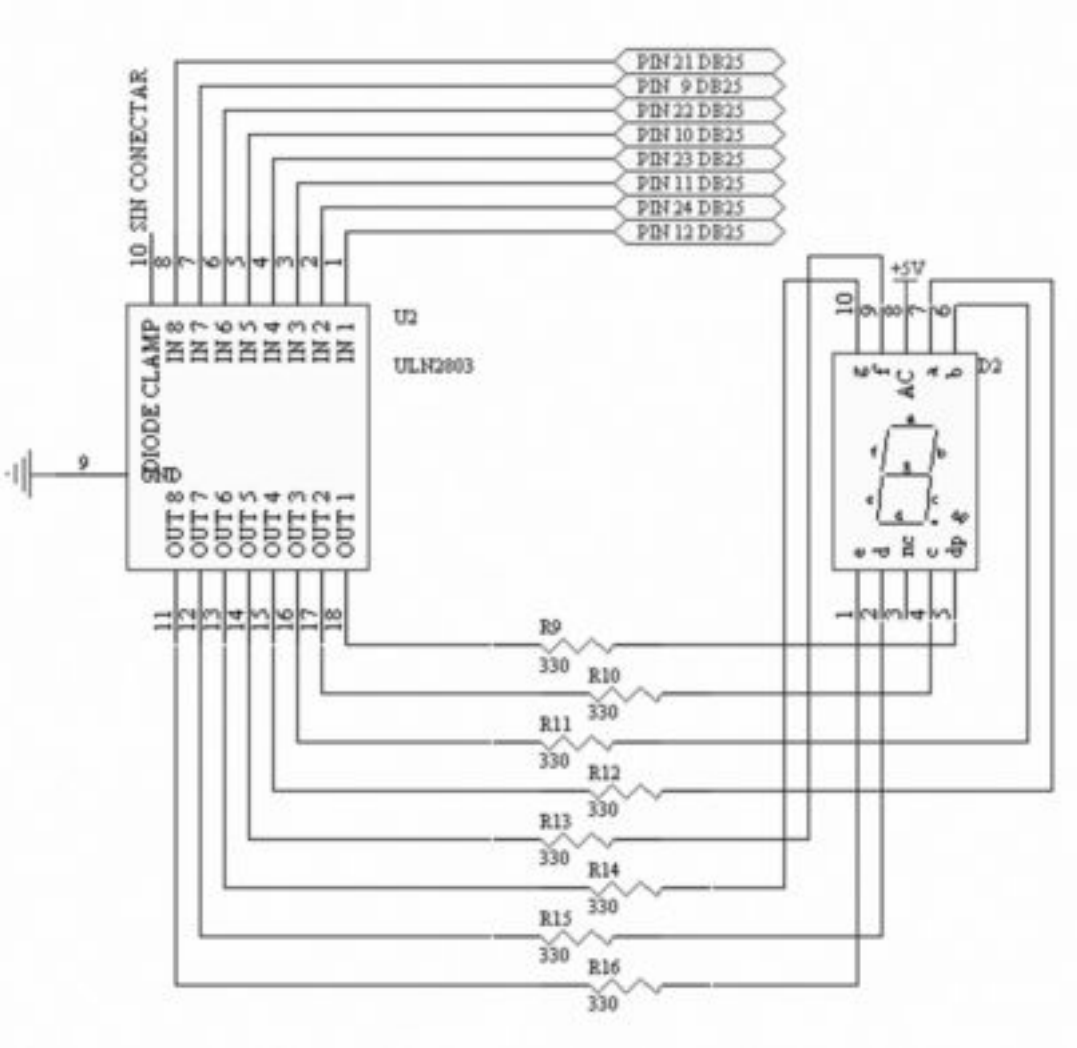


Esquemático mostrando las llaves

Los dos circuitos de excitación de cada display son idénticos. Las resistencias R1 a R16 se utilizan para limitar la corriente que pasa por los led y por cada uno de los inversores de los ULN2803.



Esquemático mostrando los componentes U1 y D1



Esquemático mostrando los componentes U2 y D2

Cada ULN2803 está compuesto de 8 inversores que sirven de buffer entre el dispositivo de salida (en este caso, el EPM7128) y los diodos *led* que conforman a los displays.

Cada uno de esos inversores tiene la capacidad de brindar una corriente de hasta 500 mA, suficiente en este caso, para alimentar a los diodos *led*.

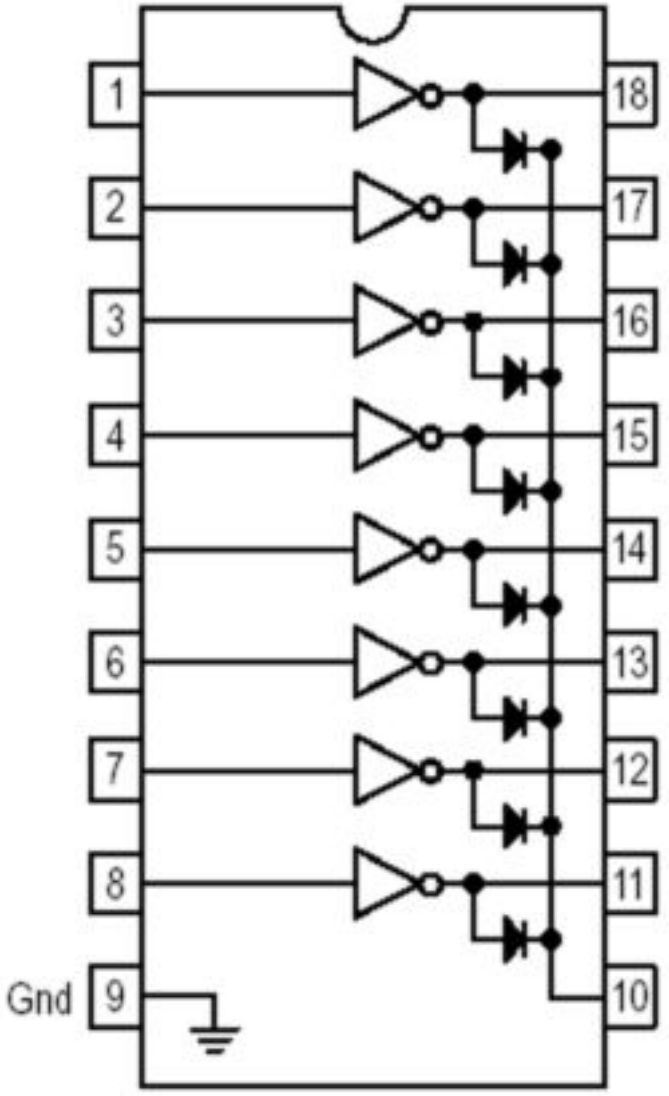


Diagrama del ULN2803

Tabla de conexiones entre placa principal y placa 1			
PIN	Dispositivo		Placa 1
1.		GND	GND
2.	77	EPM7128	Pin 8 de U2
3.	80	EPM7128	Pin 6 de U2
4.	84	EPM7128	Llave de habilitación
5.	1	EPM7128	Pulsador reset
6.	4	EPM7128	Pin 4 de U2
7.	6	EPM7128	Pin 2 de U2
8.	9	EPM7128	No conectado
9.	10	EPM7128	Pin 7 de U1
10.	15	EPM7128	Pin 5 de U1
11.	17	EPM7128	Pin 3 de U1
12.	20	EPM7128	Pin 1 de U1
13.		VCC	VCC
14.		GND	GND
15.	79	EPM7128	Pin 7 de U2
16.	81	EPM7128	Pin 5 de U2
17.	1	ConectorJ7	No conectado
18.	2	EPM7128	Llave de up/down
19.	5	EPM7128	Pin 3 de U2
20.	8	EPM7128	Pin 1 de U2
21.	11	EPM7128	Pin 8 de U1
22.	12	EPM7128	Pin 6 de U1
23.	16	EPM7128	Pin 4 de U1
24.	18	EPM7128	Pin 2 de U1
25.		VCC	VCC

La placa de experimentación número 2. Esta placa permite implementar tipos de circuitos que contienen diversos componentes, incluyendo circuitos integrados, ya que el paso de varias de las islas dibujadas en el impreso son de 100 mils (100 milésimas de pulgada).

Esta placa usa varios de los pines de entrada-salida del EPM7128.

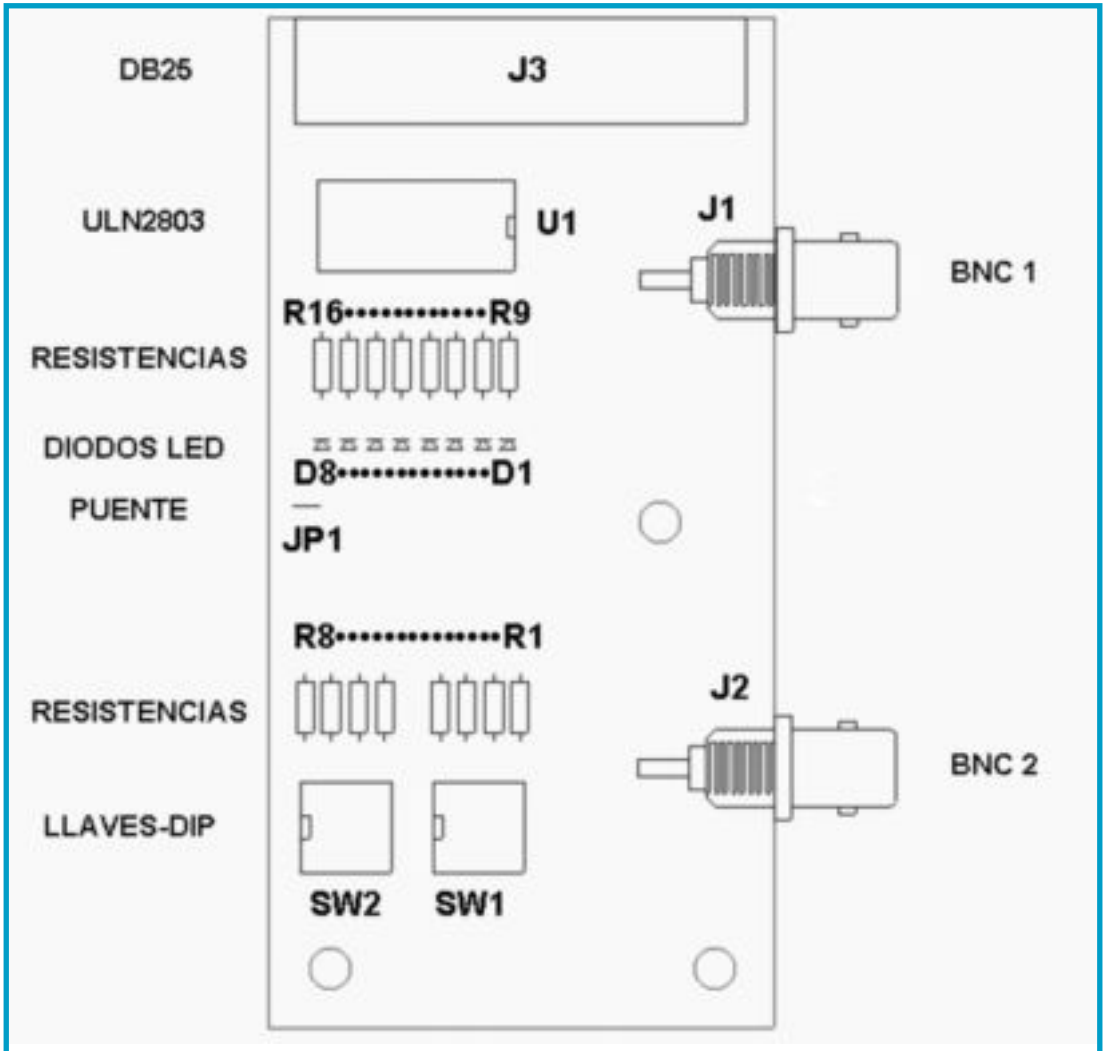
La placa de experimentación número 3. Ésta posee dos dip-switches (llaves en encapsulado DIP) de 4 llaves cada uno, lo que permite, en forma individual o en grupo, introducir hasta 8 señales digitales manuales directamente al EPM7128.

Un ULN2803 posibilita que, desde el EPM7128, se pueda excitar en forma independiente cada uno de los 8 diodos led incluidos en la placa.

A fin de poder realizar, por ejemplo, dos desarrollos diferentes en forma simultánea, se han elegido 4 led de color verde y 4 de color rojo.

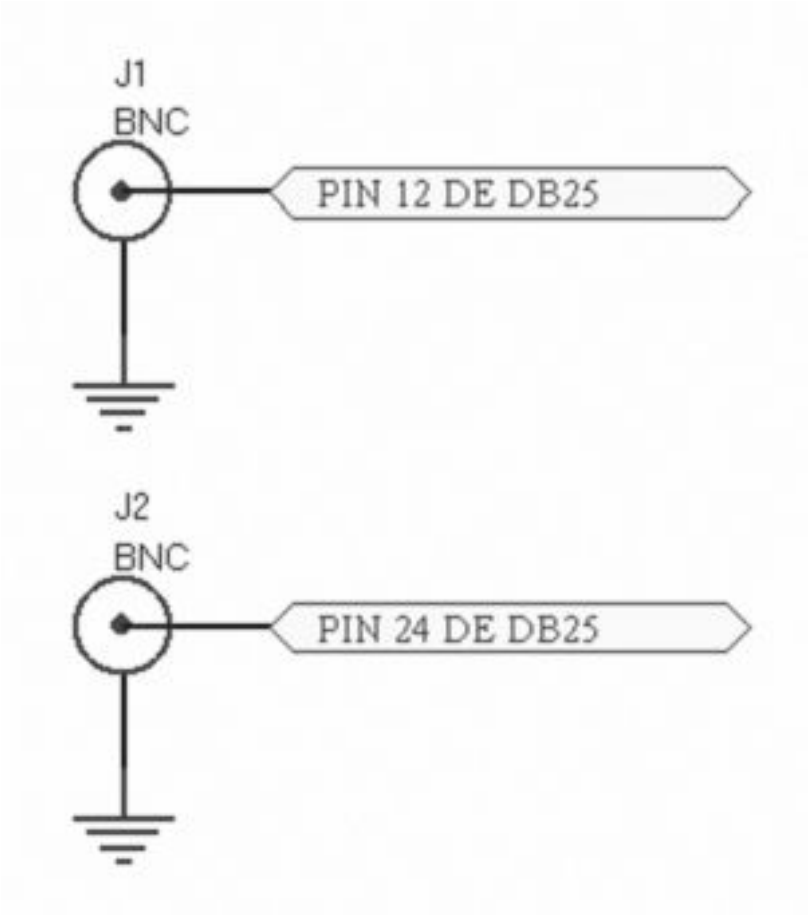
Dos conectores BNC hembra para chasis completan la placa; cada uno de ellos se encuentra conectado a un pin de entrada-salida del EPM7128 y pueden ser usados independientemente, como entrada o salida, según sea el proyecto.

Para el caso presentado en el siguiente título, "El ensayo y el control", se utiliza el conector BNC1 como entrada externa compatible con TTL hacia el EPM7128 y el BNC2 como salida compatible con TTL desde el EPM7128 hacia el exterior.



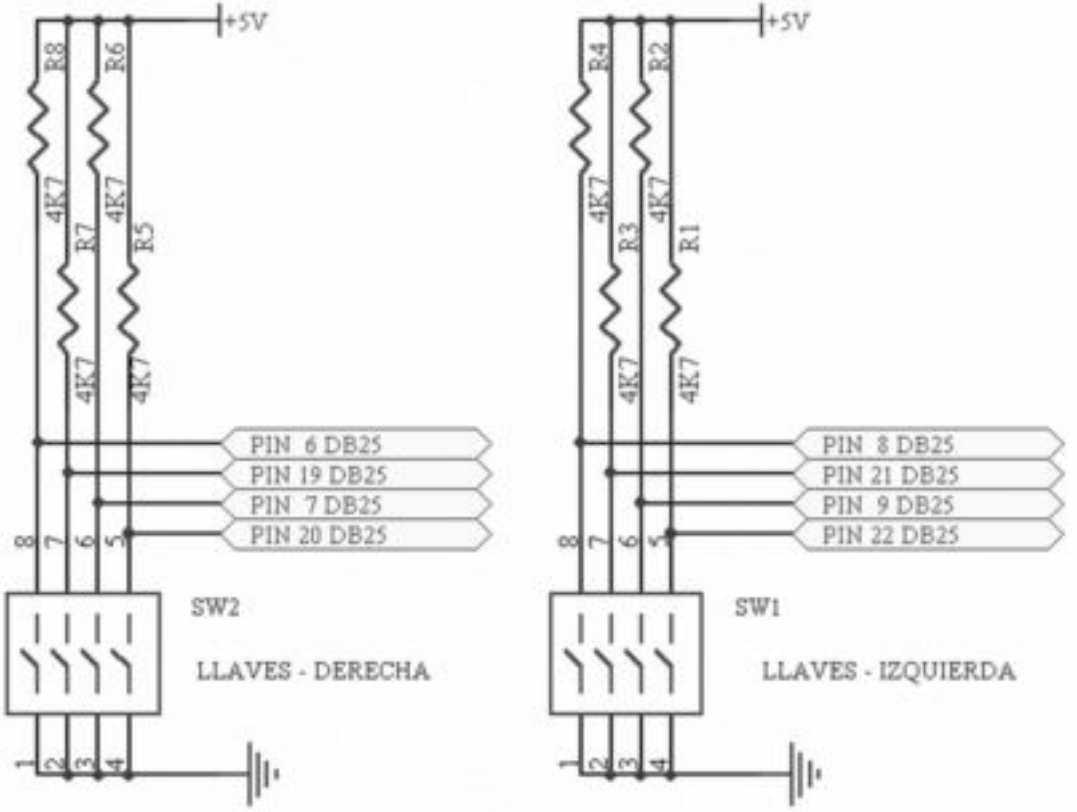
Disposición de componentes de la placa 3

***U1**. Óctuplo buffer inversor tipo ULN2803 que comanda a display D1.*
***D1-D4**. Diodo led de 3 mm de color rojo.*
***D5-D8**. Diodo led de 3 mm de color verde.*
***J1**. Conector BNC hembra, para chasis.*
***J2**. Conector BNC hembra, para chasis.*
***J3**. Conector DB25 hembra a 90 grados, para impreso.*
***SW1**. Llave cuádruple tipo DIP-SWITCH, para impreso.*
***SW2**. Llave cuádruple tipo DIP-SWITCH, para impreso.*
***R1-R8**. Resistencia de 4 K7 de ¼ W limitadora de corriente, para las llaves.*

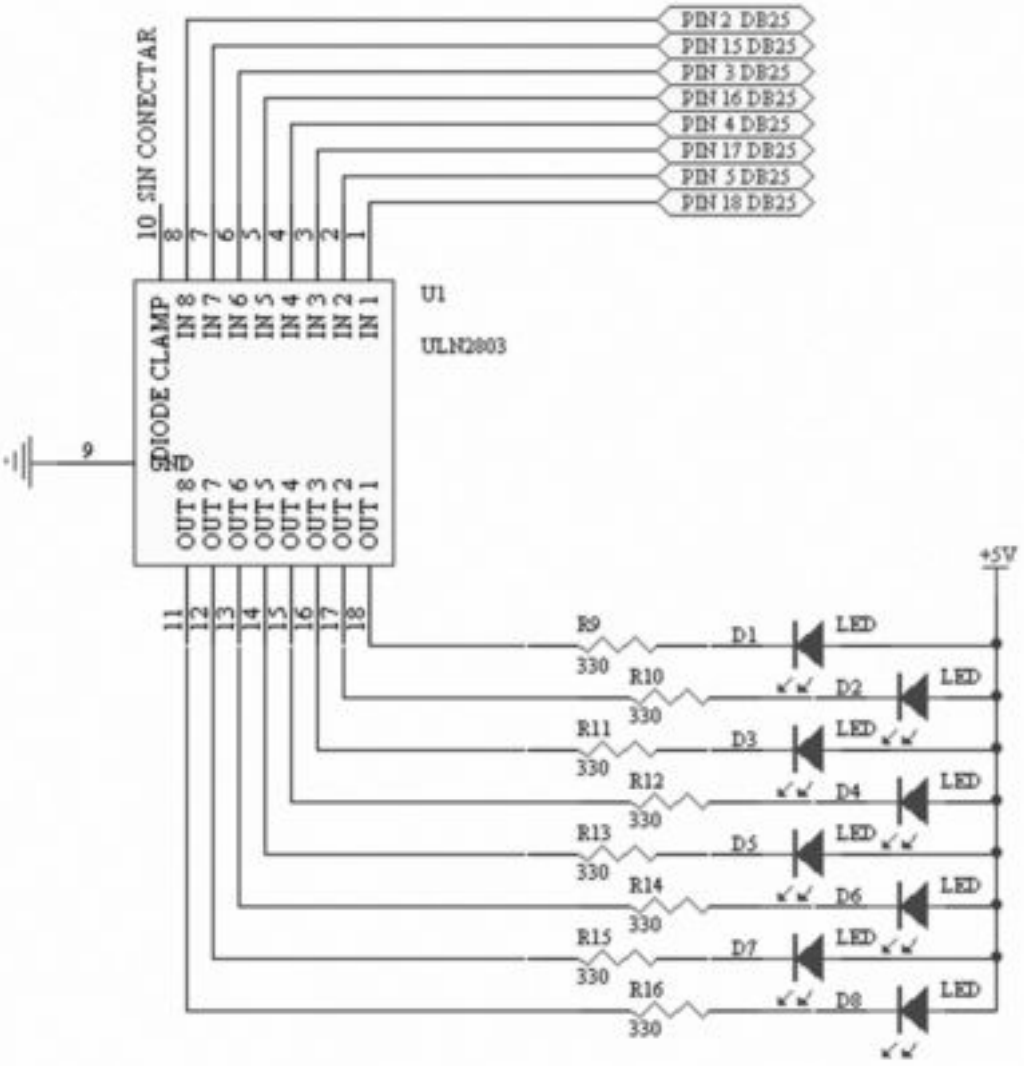


Esquemático del conexionado de conectores BNC

Esquemático del conexionado del conector DB25



Esquemático del conexionado de las llaves DIP-SWITCH



Esquemático del conexionado del ULN2803 y diodos LED

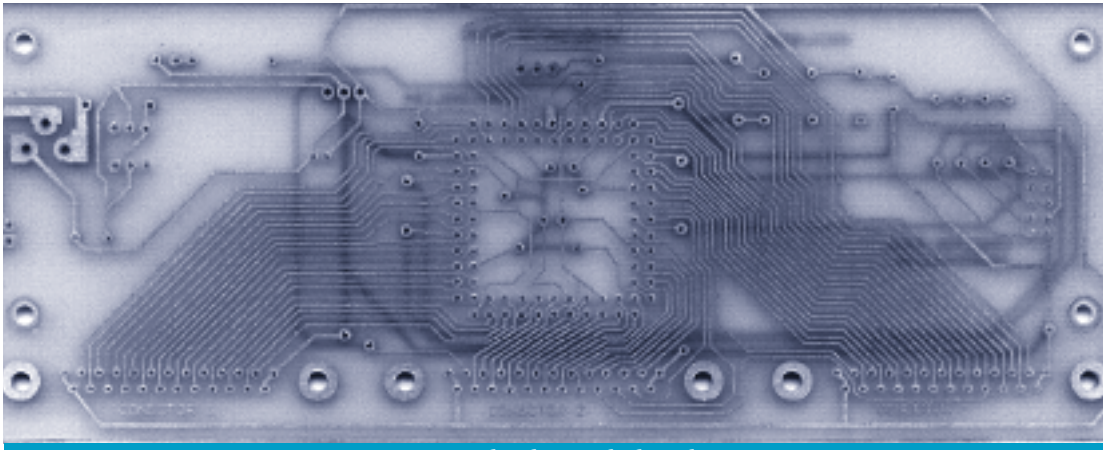
El armado del hardware

Vamos a organizar las tareas de armado en cinco etapas:

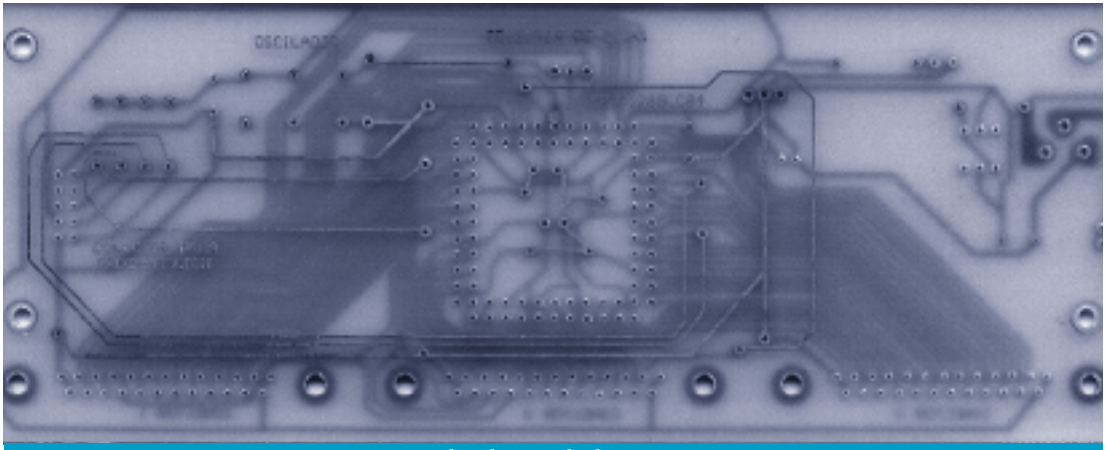
1. Armado de la placa principal.
2. Armado de la interfaz de programación.
3. Armado de la placa de práctica número 1.
4. Armado de la placa de práctica número 2.
5. Armado de la placa de práctica número 3.

1. Armado de la placa principal

Esta placa es un circuito impreso de doble faz, según se puede apreciar en las siguientes figuras:

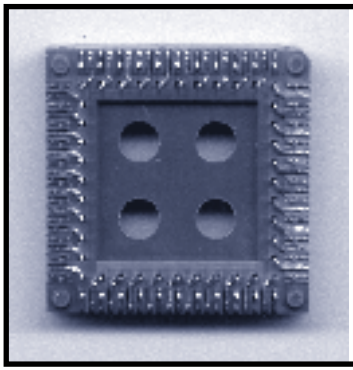


Impreso de placa 3 lado cobre

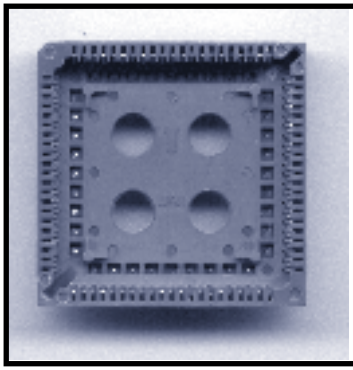


Impreso de placa 3 lado componentes

El chip empleado tiene un encapsulado tipo PLCC de 84 pines, distribuidos en un formato cuadrado de doble línea.



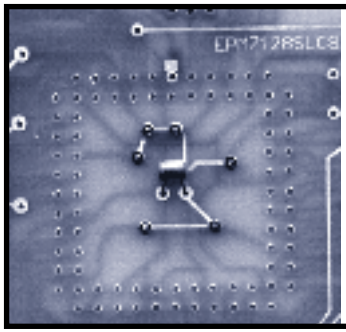
Zócalo PLCC; vista posterior



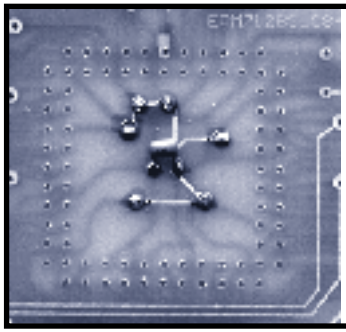
Zócalo PLCC; vista frontal

1.1. El EPM7128 tiene varios pines de alimentación de +5 V y GND que deben ser interconectados, además de los pines utilizados para entrar o salir con señales digitales. Es aconsejable, por ello, incluir un capacitor de 100 nF, usado como filtro de alimentación lo más cerca posible de los bornes de +5 V y GND. Para esto, optamos por ubicarlo en el centro del área del chip. Dado el tamaño reducido de dicha área (sobre ella va soldado el zócalo PLCC), empleamos un capacitor miniatura (puede ser de montaje superficial).

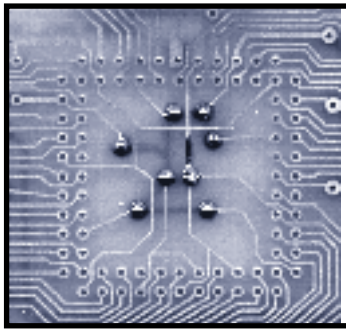
Primero, soldamos el capacitor en ambas caras del impreso y, luego, colocamos y soldamos *linking* pins -pines de interconexión- también en ambas caras.



Zoom de la zona del EPM7128: Ubicación del capacitor de 100 nF y linking pins; lado componentes

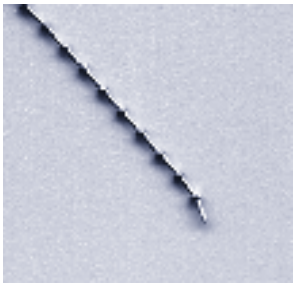


Zoom de la zona EPM7128: Soldado de componentes en dicha zona



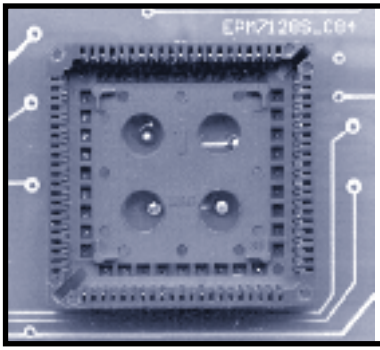
Zoom de la zona del EPM7128: Soldado en dicha zona lado cobre

Los *linking pins* son clavijas con cabeza sin punta; se presentan en tiras y son fácilmente removibles. Según su diámetro (el típico es de 0,7 mm), agujereamos cada una de las vías con una mecha acorde, a fin de que los linking pins entren a presión. Luego, soldamos ambos lados de la placa.



Tira de linking pins

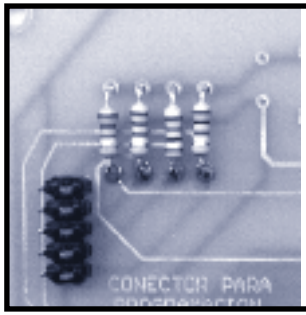
1.2. Una vez que están soldados el capacitor y los *linking pins*, montamos el zócalo PLCC, soldando los 84 pines del lado cobre.



Zoom zona del EPM7128: Ubicación del zócalo PLCC

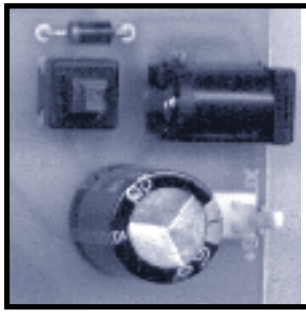
1.3. Ubicamos y soldamos los *linking pins* en el resto del circuito.

1.4. Soldamos la tira de pines doble de 2 x 5 y las 4 resistencias de 1kΩ.

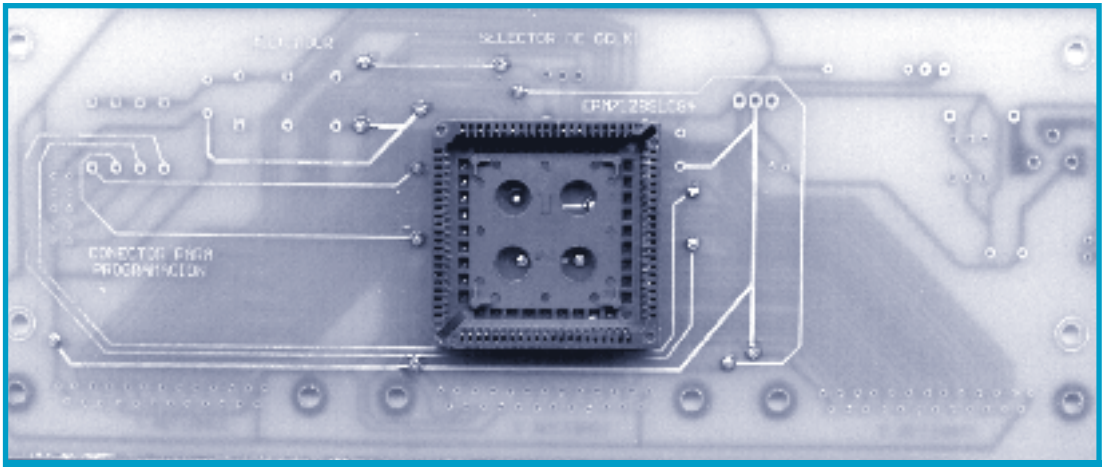


Zoom de la zona de entrada de programación

1.5. Soldamos el regulador de tensión -7805T-, la tira de pines de 1 x 3, la resistencia de alimentación de led y led. Lo mismo con los capacitores de fuente de 100 nF y 10 µF.

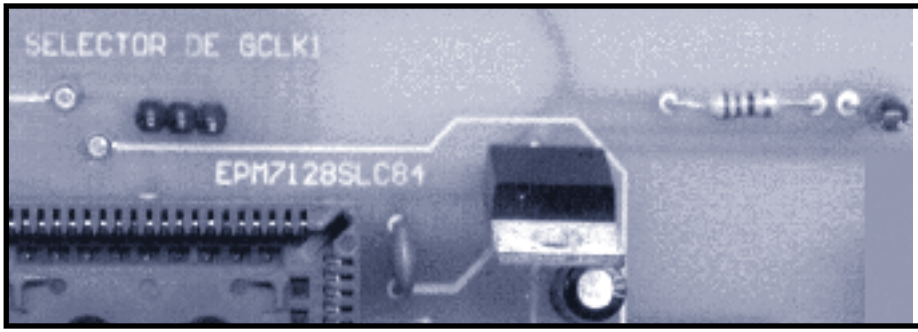


Zoom de la zona de entrada de alimentación



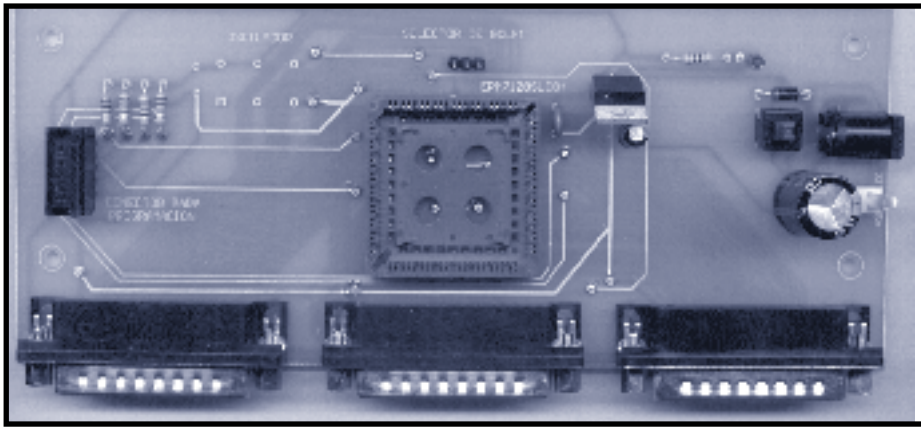
Disposición de vías y ubicación de los linking pins

1.6. Soldamos el jack de alimentación y el conector auxiliar de alimentación. Lo mismo con el diodo 1N4007 de protección contra inversión de polaridad, el capacitor electrolítico de filtrado y el interruptor de encendido.



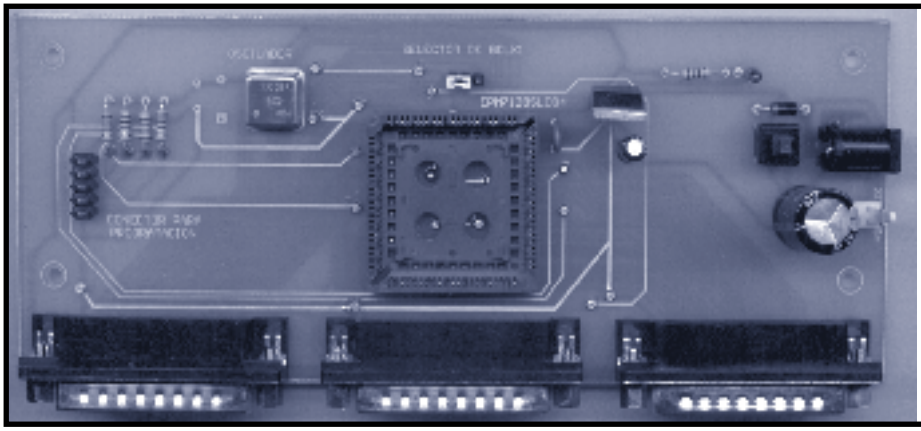
Zoom de la zona de salida de alimentación ya regulada

1.7. Soldamos los conectores DB25.



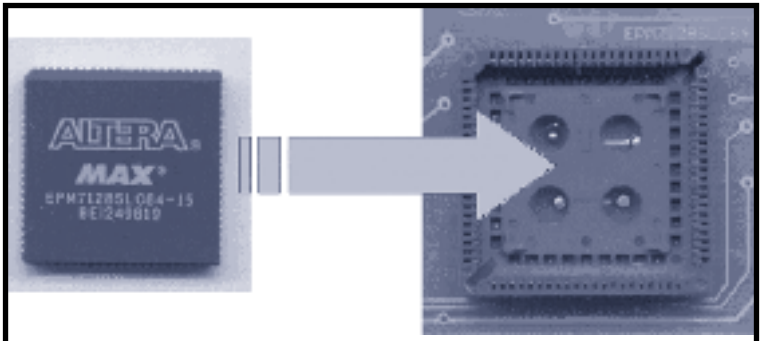
Emplazamiento y soldado de todos los conectores

1.8. Soldamos el oscilador a cristal.



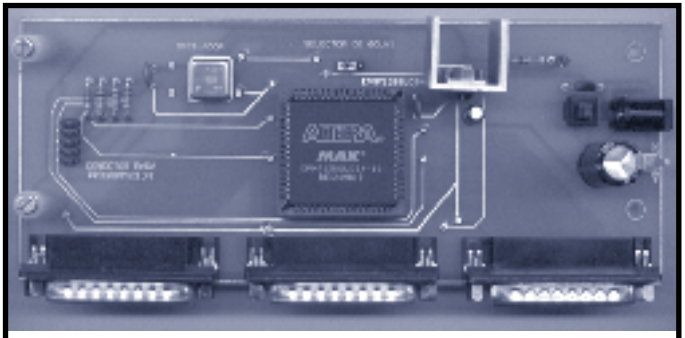
Emplazamiento del chip de oscilador a cristal

1.9. Emplazamos el chip EPM7128, presionando suavemente en las aristas del chip, de a una por vez, hasta que quede firmemente colocado.



Esquema mostrando la posición correcta para colocar el chip en el zócalo PLCC

1.10. Soldamos el capacitor de filtrado del oscilador y montar el disipador.



Placa principal ya armada

2. Armado de la interfaz de programación

La base de la interfaz se centra en:

- un conector DB25 macho para cable,
- un *buffer Schmitt Trigger* 74LS244,
- 7 resistencias de 30 ohm y
- un capacitor de 100 nF de policarbonato.

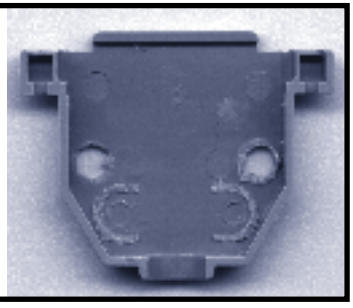
El extremo del conector DB25 se conecta al puerto paralelo de la PC y el otro a la placa principal, vía un conector hembra IDC10.

A fin de poder miniaturizar dicha interfaz, empleamos un circuito impreso doble faz, para ubicar todos los componentes dentro del espacio que existe entre las dos tapas de un conector DB25 para cable.

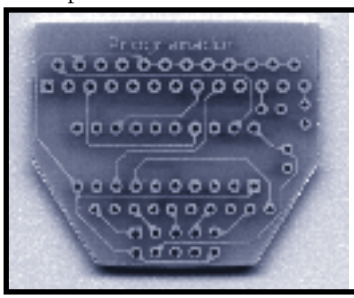
2.1. Cortamos los postes internos de cada tapa del conector DB25, a fin de dejar espacio para ubicar el circuito impreso.

Vista interna de la tapa del conector DB25, ya preparada

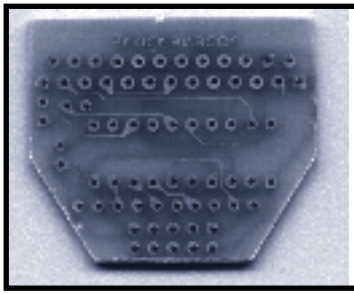
Vista externa de la tapa del conector DB25, ya preparada



2.2. Desarrollamos el circuito impreso doble faz para la interfaz de programación. Una vez fabricado, puede ser necesario limar los bordes, hasta que calce perfectamente dentro de las tapas.

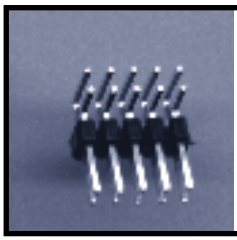


Impreso lado cobre



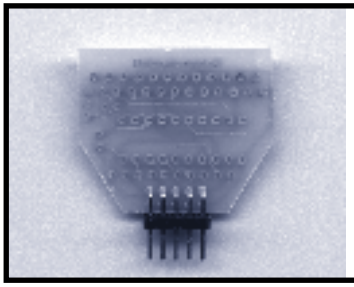
Impreso lado componentes

2.3. Cortamos una tira doble de postes a 90 grados, tal que queden dos líneas de 5 postes.



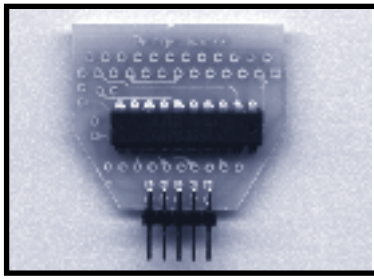
Corte de la tira de pines doble a 90 grados

2.4. Ubicamos y soldamos dicha tira al circuito impreso.



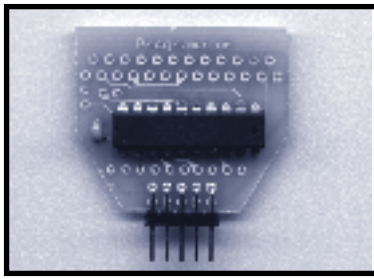
Ubicación y soldado de la tira de pines de 2 x 5

2.5. Soldamos el buffer 74LS244 al circuito impreso. Tenemos presente que es necesario hacer soldaduras en ambas caras del impreso.



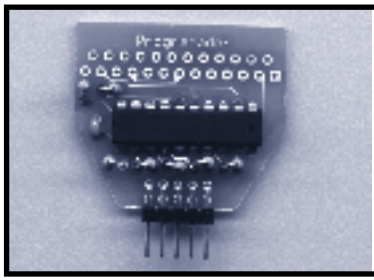
Soldado del 74LS244

2.6. Soldamos el capacitor de desacoplo al *buffer*.



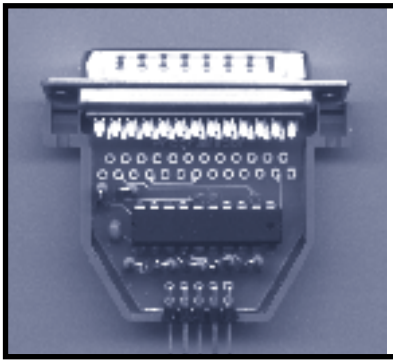
Soldado del capacitor de desacoplo

2.7. Soldamos las 7 resistencias de 30 ohm.



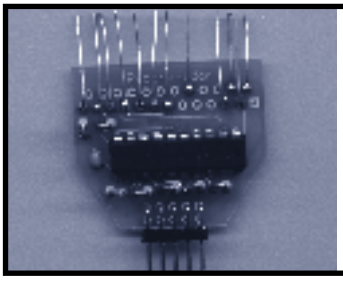
Soldado de las 7 resistencias de 30 ohm

2.8. Ubicamos el impreso dentro de una de las tapas (es indistinto cuál es, la superior o la inferior). Presentamos el conector DB25. Si es necesario, limamos la cara que lo toca a fin de que pueda acoplarse todo, al cerrar ambas tapas.



Emplazamiento del conector DB25

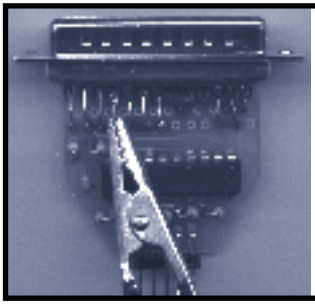
2.9. Retiramos el impreso y le soldamos alambres esmaltados de longitud suficiente como para que puedan soldarse, luego, al conector DB25.



Soldado de cables

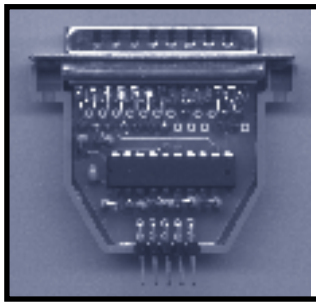
2.10. Ubicamos el conector DB25 contra el borde superior de la tapa y doblamos los alambres a la altura necesaria para cada contacto. Cortamos el alambre excedente y soldamos cada uno al pin que corresponda. Repetimos esto en cada contacto.

En la foto se muestra un clip "cocodrilo" que se usa a fin de que el calor del soldador no derrita el estaño ya soldado en el impreso.



Técnica para evitar el desoldado de la otra punta del cable

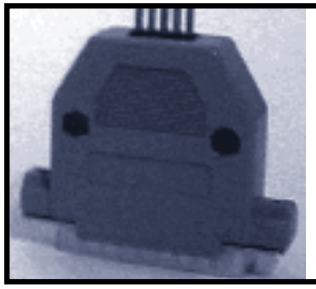
2.11. Ubicamos el impreso sobre una tapa, cortamos con trincheta y rebajamos a lima la zona donde está el conector IDC10, a fin de que pueda calzar dentro. Repetimos este paso para la otra tapa.



Vista final del soldado de cables al conector DB25

2.12. Ponemos ambas tapas y comprobamos que todo quede firme en el interior.

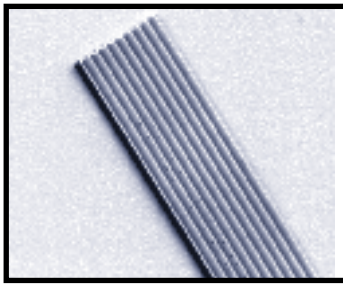
Provisoriamente, para las pruebas, podemos mantener las tapas unidas con cinta aisladora. Posteriormente, para su uso permanente, le recomendamos utilizar un adhesivo de contacto para fijarlas definitivamente.



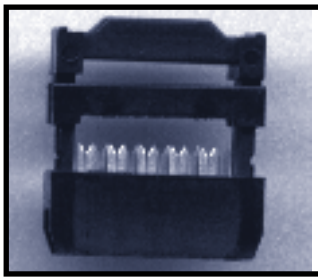
Vista final del programador, armado sin conector

Conviene identificar con un símbolo -en la parte posterior de la tapa- dónde está el pin número 1 del conector IDC10; éste se identifica en el impreso como un cuadrado. Esta identificación es importante para conectar correctamente el cable plano a esta interfaz.

2.13. Armamos el cable plano con los dos conectores hembra IDC10 en cada extremo. Dicho cable tiene, en general, uno de los hilos de color rojo que sirve como referencia para indicar cuál es el número 1. Asimismo, cada conector tiene una indicación (por ejemplo, un triángulo) para el mismo fin, lo que se respeta en el armado.



Cable plano de 10 cables

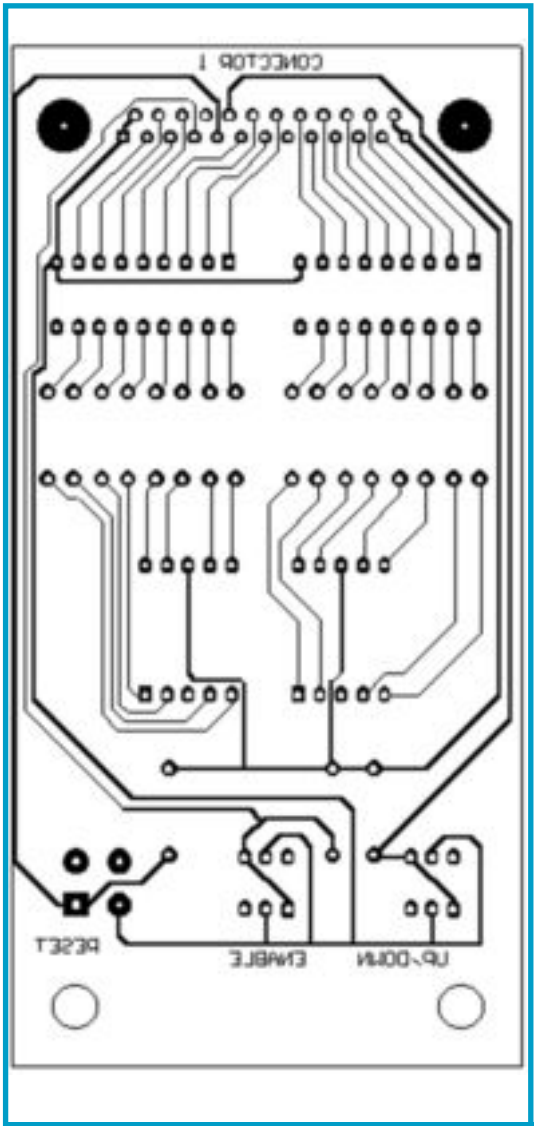


Conector IDC hembra de 10 pines

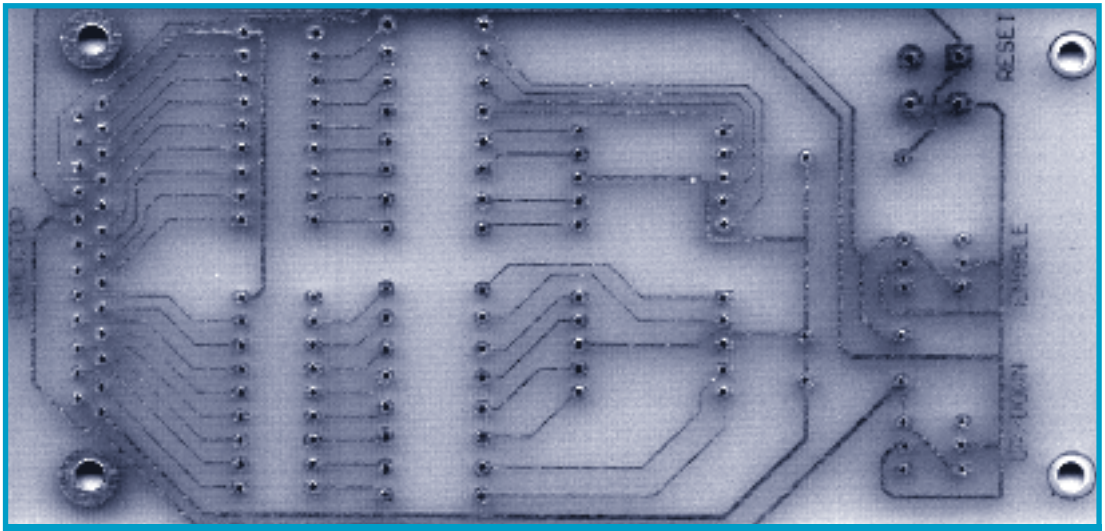
Programador con el cable completo

3. Armado de la placa de práctica número 1

La placa consiste en un circuito impreso simple faz, que se conecta a la placa principal a través del conector DB25.



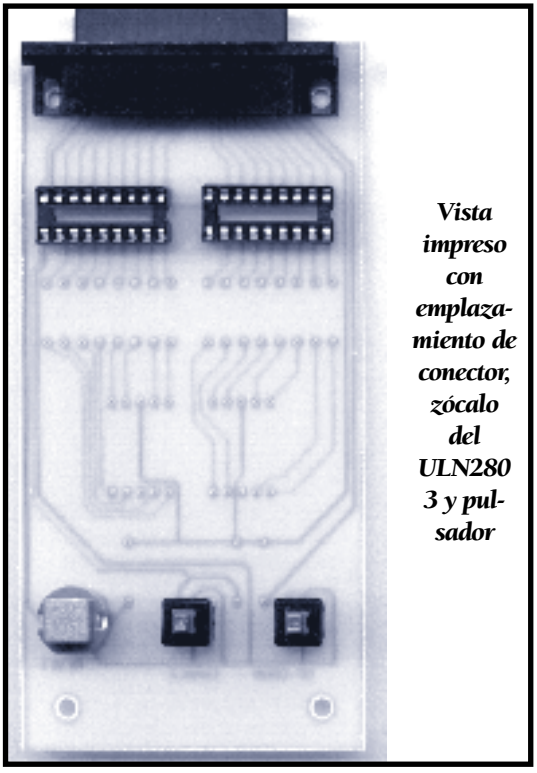
Croquis del circuito impreso de la placa 1



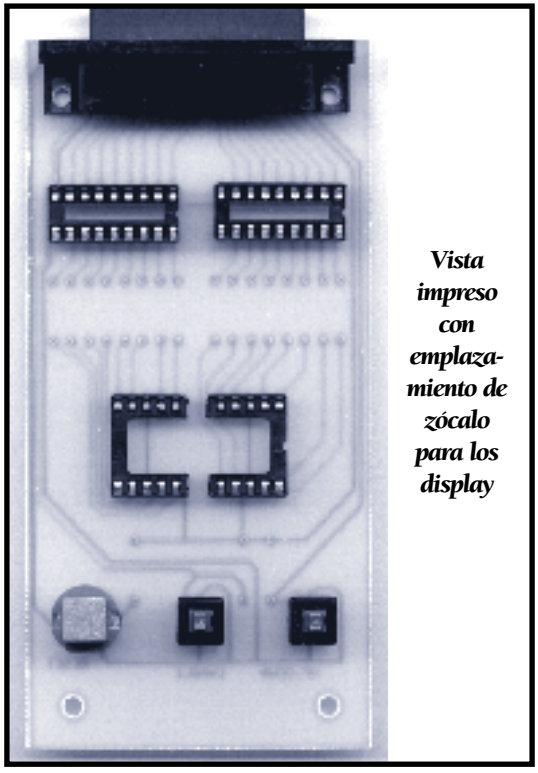
Impreso de la placa 1; lado cobre

3.1. Soldamos el conector DB25, zócalos DIP-18, el pulsador sin retención y las llaves.

3.2. En el impreso, dejamos ex profeso una separación de los displays de 7 segmentos, a



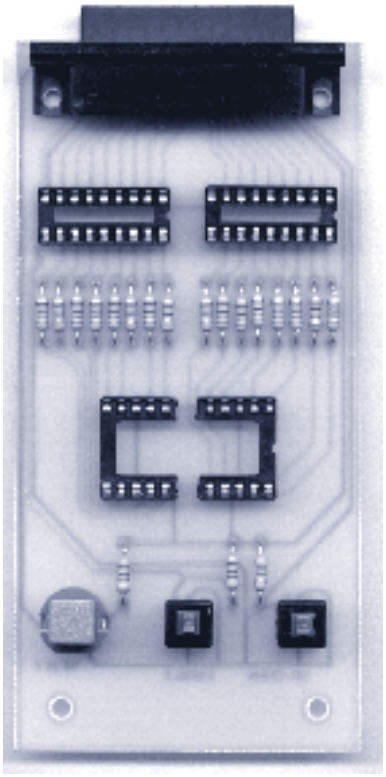
Vista impreso con emplazamiento de conector, zócalo del ULN2803 y pulsador



Vista impreso con emplazamiento de zócalo para los display

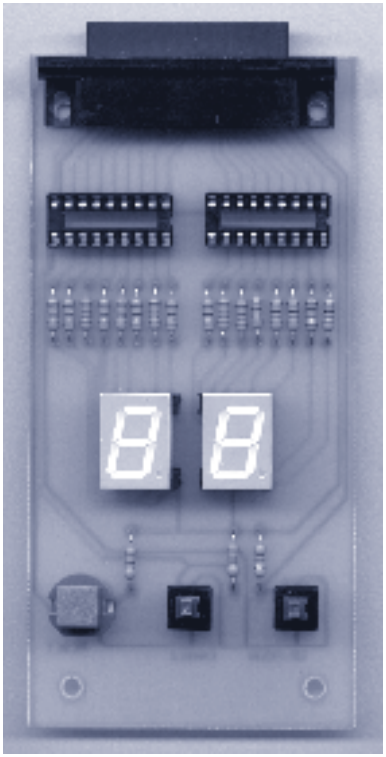
fin de poder utilizarlos en forma conjunta o individual. En caso de necesidad de reemplazarlos, podemos montarlos sobre un zócalo. Como no se proveen zócalos tipo DIP-10 del tamaño de dichos displays, es posible emplear uno DIP-24 (dos líneas de 12 contactos) y cortarlo.

3.3. Montamos las resistencias de 330 ohm y 4K7.



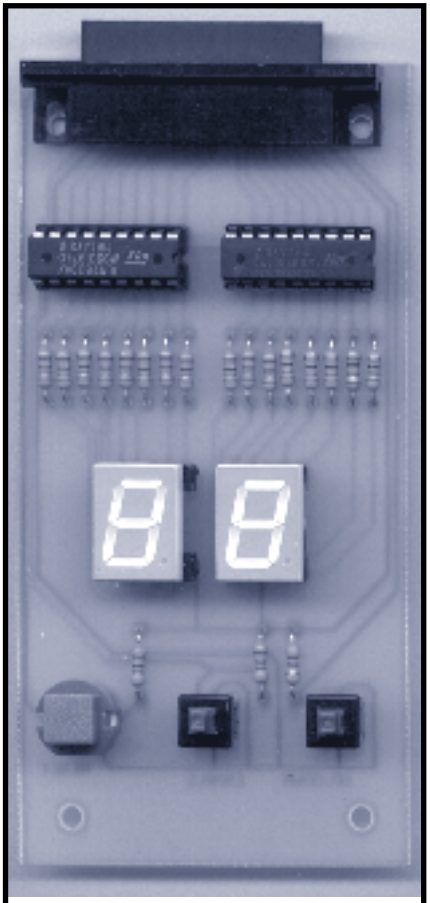
Vista impreso con el emplazamiento de las resistencias

3.4. Montamos los displays TDSR5150.



Vista impreso con los displays agregados

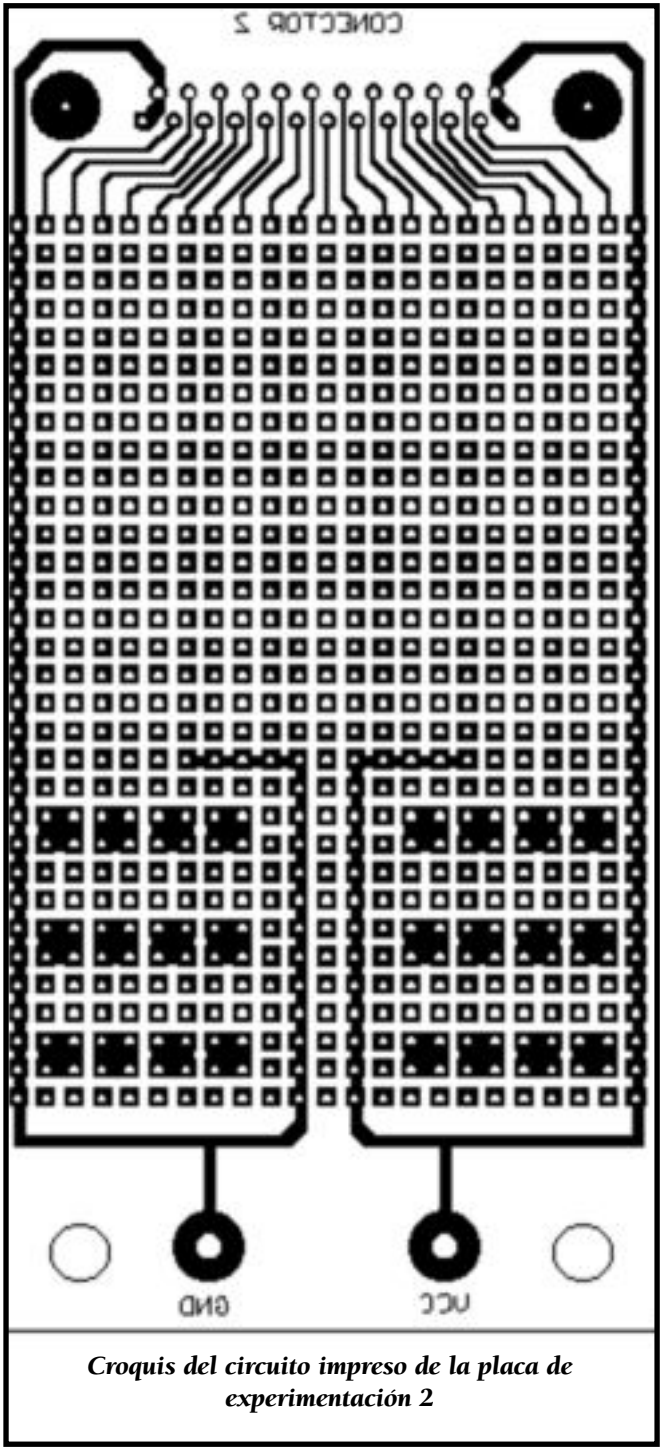
3.5. Por último, montamos los drivers de LED, ULN2803.



Vista definitiva del impreso de la placa de experimentación 1

4. Armado de la placa de práctica número 2

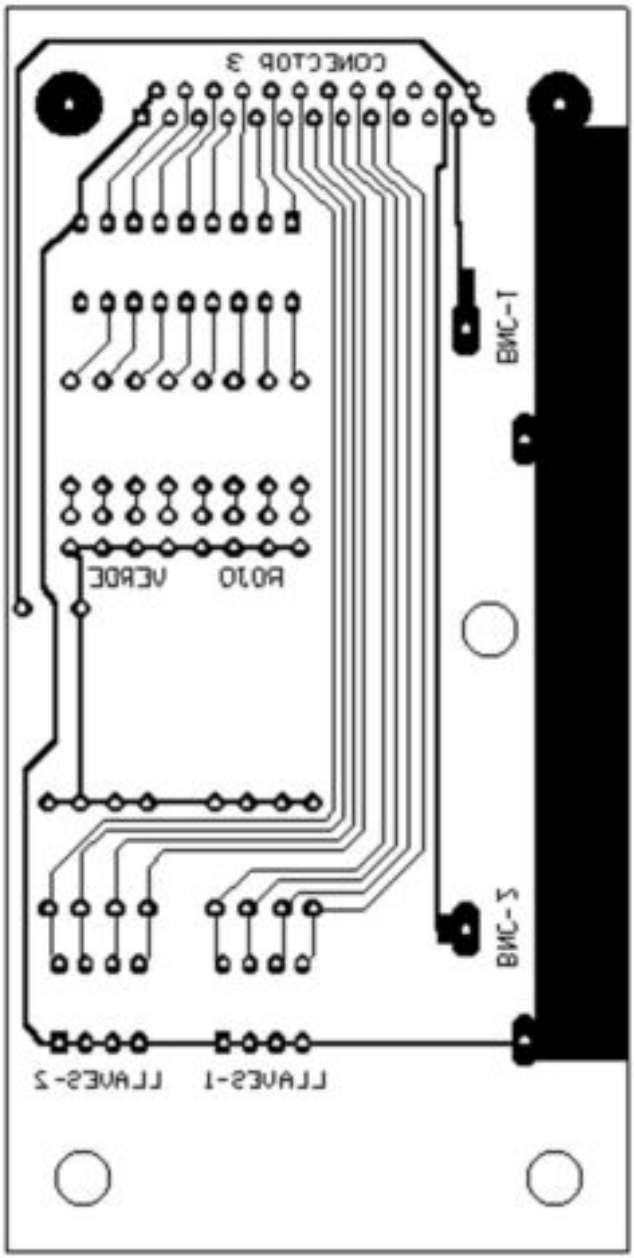
Esta placa está prevista para que los alumnos diseñen sobre ella, por lo que el montaje de componentes queda libre de explicación.



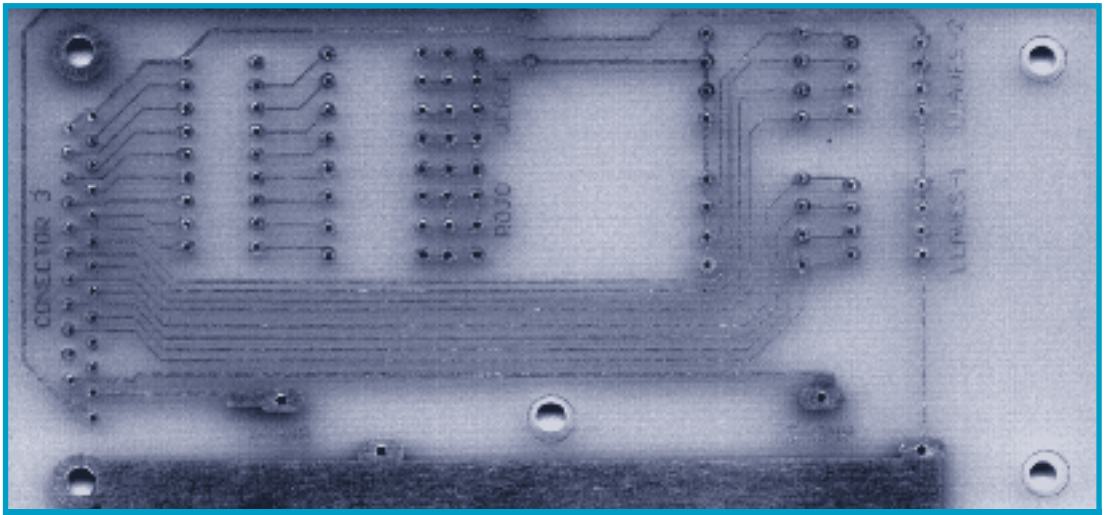
Croquis del circuito impreso de la placa de experimentación 2

5. Armado de la placa de práctica número 3

La placa consiste en un circuito impreso simple faz que se conecta a la placa principal a través del conector DB25.

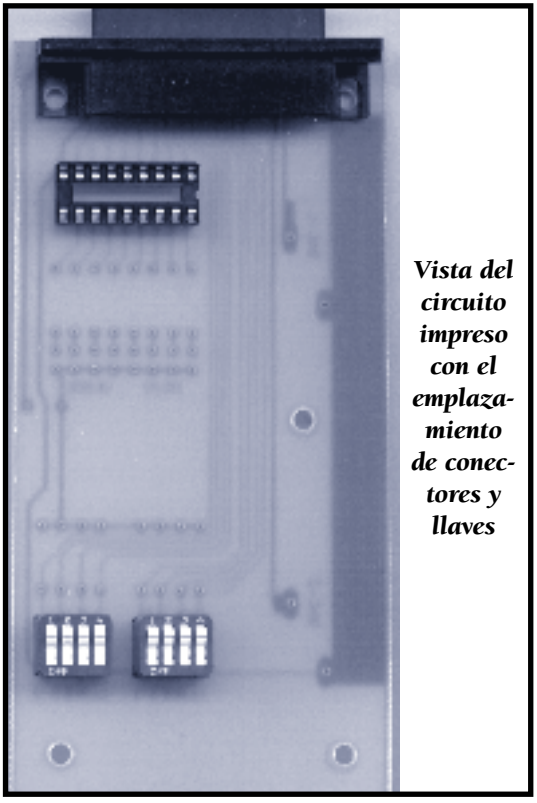


Croquis del circuito impreso de la placa de experimentación 2



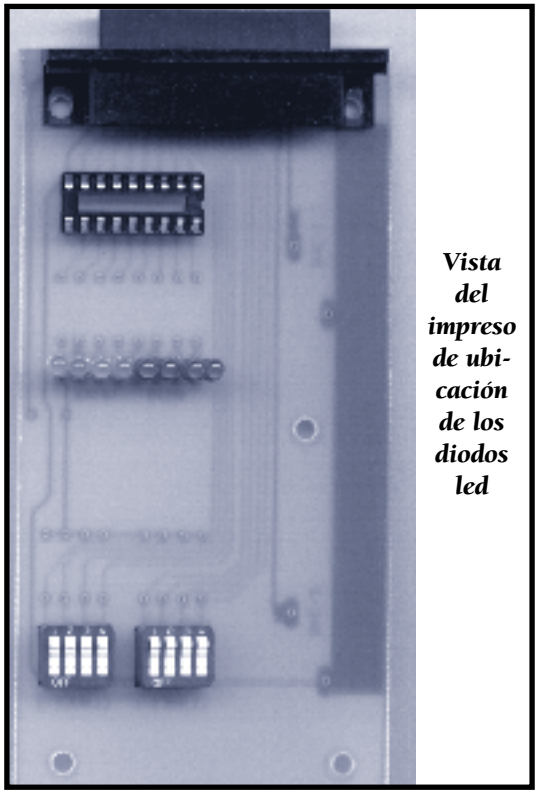
Vista del circuito impreso de la placa número 3

5.1. Soldamos el conector DB25, zócalo DIP-18 y las dos llaves *DIP-Switch*.



Vista del circuito impreso con el emplazamiento de conectores y llaves

5.2. Soldamos los 8 diodos led, 4 rojos y 4 verdes



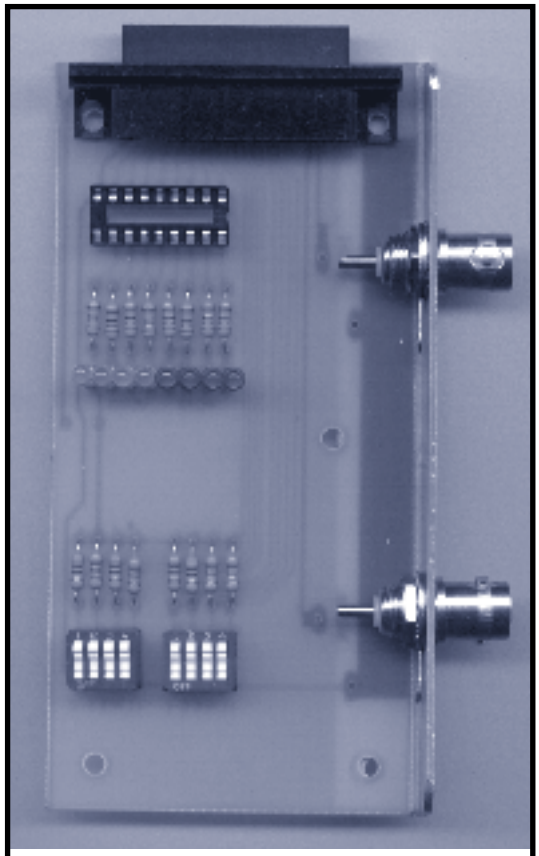
Vista del impreso de ubicación de los diodos led

5.3. Soldamos las resistencias de 330 ohm para el ULN2803 y las de 4K7 para las llaves *DIP-Switch*.

Vista del impreso con el emplazamiento de todas las resistencias

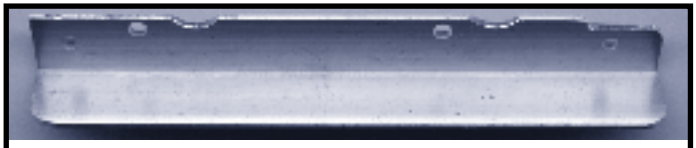
5.4. Realizamos perforaciones al perfil para poder amurarlo al circuito impreso y pasar dichos conectores. Se trata de un perfil en "L" de aluminio de 90 mm de largo, y de 15 x 15 mm de ancho, utilizado como soporte mecánico para los conectores BNC hembra tipo chasis.

5.5. Emplazamos el perfil "L" con los dos BNC ya colocados.

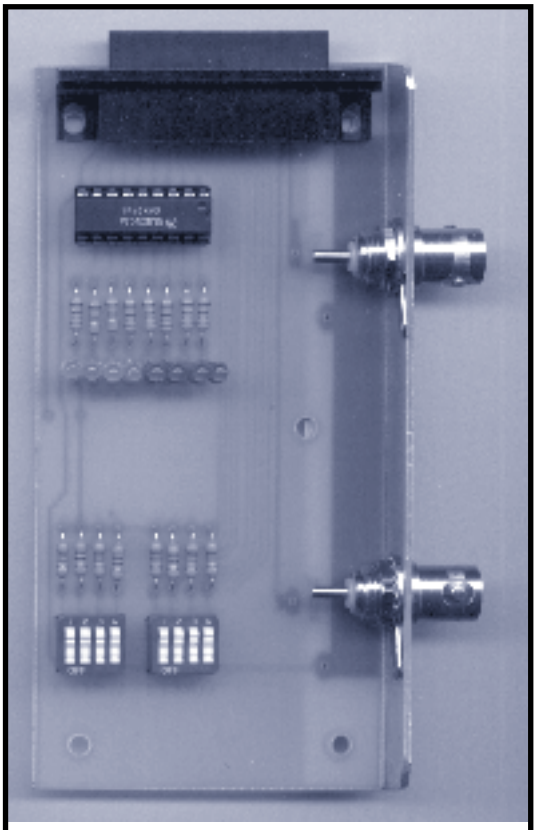


Vista del emplazamiento del perfil de aluminio con los conectores

5.6. Emplazamos el ULN2803.

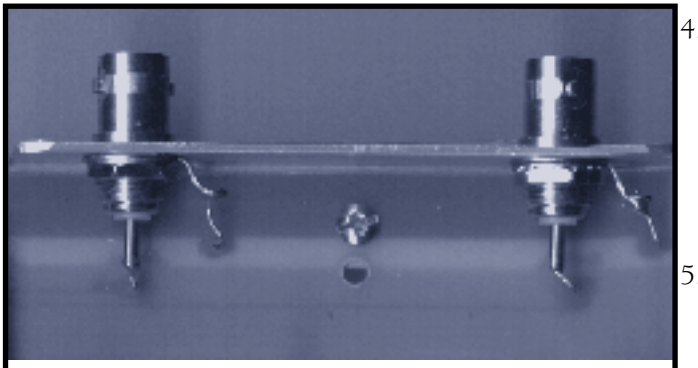


Perfil "L" de aluminio empleado para sujeción de los conectores BNC



Vista del impreso con el chip ULN2803

5.7. Soldamos los dos terminales de cada conector BNC al impreso.



Vista final del circuito impreso de la placa de experimentación 3

2. Software

El software para trabajar con el chip EPM7128 es el MAX+PLUS-II de la empresa *Altera*®. Es necesario instalarlo en una PC que corra bajo el sistema operativo Windows 98 en adelante.

> Si no dispone del archivo de instalación, puede bajarlo desde el sitio de Internet de la empresa.

Los pasos a seguir para realizar la instalación son:

1. Ingrese al sitio web de la empresa: http://www.altera.com

2. En la parte superior de la página web va a ver una serie de barras; haga clic en *Download*.

3. En *Legacy Design Software*, haga clic en la opción MAX+PLUS II *Student Edition*, la que lo conduce a una página del centro de descarga de la empresa.

4. Haga clic en *Download student10.0.exe* (archivo de 51 *Mbytes*, aproximadamente); va a ingresar a una nueva página con un formulario que es necesario llenar.

5. Una vez hecho esto, presione el botón de *Submit*; aparece una ventana de diálogo para bajar el software.

6. Presione en *Guardar*. Elija en qué subdirectorio quiere que se almacene el programa. Como son 51 Mbyte, depende de la velocidad de conexión de Internet saber cuánto tiempo dura la descarga (para una bajada a 30 Kbps, usted y sus alumnos tardarán, más o menos, 30 minutos).

7. Una vez descargado el software, pida una licencia a *Altera®* para usarlo. Para esto, vaya a la opción *Licensing* que puede estar en la parte izquierda de la última pantalla abierta; si no es así, vuelva a la página principal y haga clic en *Licensing*.

8. Una vez en esta página, vaya al final y haga clic sobre la opción MAX+PLUS II *software for student and universities*. Aparece otra página en la que debe elegir, dentro de *Legacy Software*, la opción MAX+PLUS II *Student Edition Software* Versión 10.2, 10.1 or 9.23.

9. Al presionar el botón de *Continue*, va a ir a otra página que le pide que ingrese el número de serie del disco rígido donde va a instalar el software. Este número puede obtenerse desde el sistema operativo DOS, escribiendo "dir/p".

10. Una vez ingresado, presione *Continue*. Aparece otra página con un formulario que es necesario llenar[25].

11. Presione *Continue*. La última pantalla da un mensaje de aprobación a lo solicitado y expresa que, dentro de las próximas horas, por e-mail, la escuela va a recibir el archivo de autorización para el uso del software.

12. Una vez que usted tiene el archivo de instalación, ejecútelo y siga las instrucciones que le solicita. Si usted no indica lo contrario, el programa es instalado en un subdirectorio creado en el directorio raíz, denominado maxplus2 y, además, es creado otro subdirectorio llamado max2work que contiene varios ejemplos de aplicaciones.

13. Luego de completada la instalación, todavía es necesario un paso más para emplear el software; éste es el ingreso del archivo *license.dat* que envía la empresa por correo electrónico. En este archivo hay información para que el software pueda funcionar -esto sucede sólo si coincide con el número de serie del disco rígido donde se ha instalado-. Una vez que tenga este archivo de licencia, ubíquelo en el subdirectorio maxplus2 y, luego, procedeaa correr el MAX+PLUS-II, ya sea desde el acceso directo que se crea al instalar o desde el subdirectorio maxplus2, haciendo doble clic en el archivo "max2win.exe".

14. Desde el menú principal, haga clic en *Options*. Se despliega una ventana que contiene la opción *License Setup*; haciendo clic sobre ella, aparece la ventana *License Setup*.

15. Presione sobre el botón *Browse*; va a entrar a una ventana similar a la del *Explorer de Windows®* donde debe ubicar el subdirectorio en el que se encuentra el archivo *license.dat*. Una vez hecho esto, presione el botón *OK*.

A partir de ese momento, usted y sus alumnos pueden utilizar el MAX+PLUSII.

[25] Es importante que controle que la dirección de correo electrónico sea la correcta, ya que la empresa envía por e-mail un archivo que, luego, debe ser ubicado en el subdirectorio "maxplus2", una vez que se ha instalado el software.

La descripción del software

Ya le hemos explicado cómo instalar el software; ahora, le presentamos las bases para poder emplearlo.

El *MAX+PLUS-II* es uno de los ambientes de desarrollo que nos permite diseñar proyectos de lógica digital con circuitos lógicos programables tanto SPLD, CPLD como FPGA.

Este software está formado por cuatro herramientas de diseño:

1. **Especificación del diseño**. Permite entrar las especificaciones del proyecto en forma gráfica o textual.

2. **Verificación del diseño**. Permite verificar, a través de simulaciones y análisis temporal, la viabilidad del proyecto.

3. **Procesamiento del diseño**. Permite, mediante un programa compilador y procesador de mensajes, sintetizar la lógica requerida dentro del chip seleccionado.

4. **Programación de los dispositivos**. Permite programar físicamente al dispositivo seleccionado, vía puerto serie o paralelo, según la interfaz que se disponga.

Especificación del diseño. Existen varias formas de entrar un diseño en el software.

- *Modo gráfico*. El proyecto se va armando a través de la inclusión, en una ventana gráfica, de símbolos que representan los componentes digitales a usar (compuertas, multiplexores, sumadores, contadores, etc.), cables de interconexión y bornes de entrada y salida; es decir, de todo lo necesario para especificar el circuito que se desea implementar.

- *Modo textual*. A través de lo que se denomina HDL -*Hardware Description Language*; lenguaje de descripción del hardware- se puede ordenar al programa que sintetice (implemente) el circuito pedido dentro del chip; se realiza mediante sentencias escritas [26].

Verificación del diseño. En dispositivos PLD, la manera tradicional de poder saber si el proyecto puede implementarse (una vez de haberlo descrito en forma gráfica o textual) es mediante una simulación en el tiempo.

Para esto, es necesario realizar la compilación (la explicamos en el punto siguiente) y, luego, crear un gráfico que incluya todas las señales de interés: las de entrada y las de salida del circuito.

Este gráfico muestra la evolución en el tiempo, en el eje de las abscisas (eje X). Se dan valores (0 y 1) a las señales de entrada a lo largo del eje X y se ejecuta el programa de

[26] Existen variaciones de este tipo de lenguaje HDL básico, como el AHDL de *Altera*® o el ABEL de *Xilinx*®, y versiones mucho más poderosas como VHDL y VERILOG, que son utilizadas por los diseñadores expertos

simulación, obteniendo entonces, la respuesta de las salidas a dichas entradas.

Procesamiento del diseño. Para poder simular y, luego, programar el chip se requiere compilar -sintetizar internamente el circuito dentro del chip-. En esta tarea se usa el programa *Compiler* -compilador- que se encarga de convertir en hardware el circuito que ya se ha descrito.

Programación de los dispositivos. Una vez que se está seguro de que el circuito funciona, pasa a la etapa de programación del chip. Para ello se usa el programa *Programmer* -programador- en el que se especifica qué dispositivo se desea programar (en nuestro caso, el EPM7128SCL84).

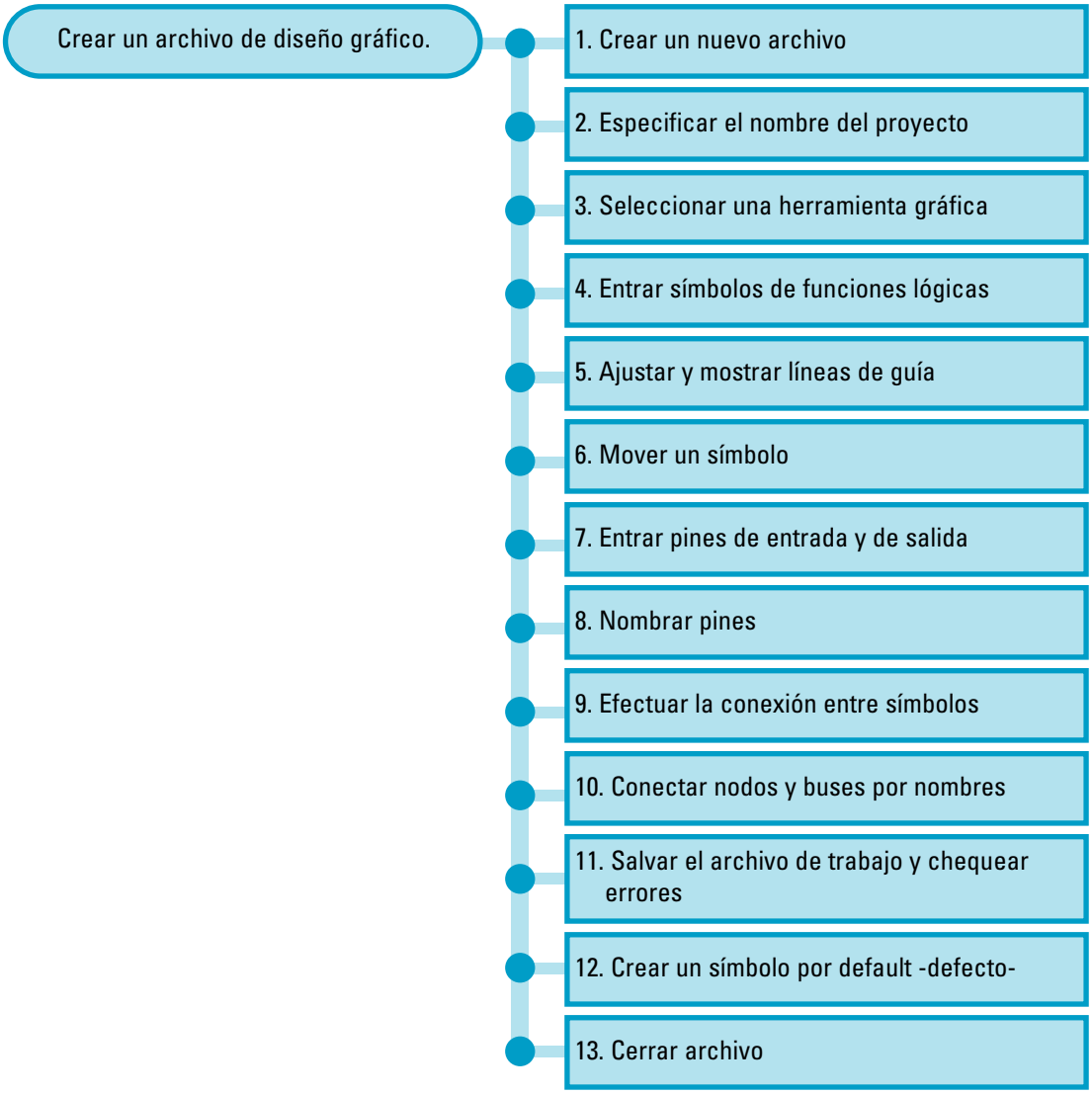
Consideremos los distintos pasos:

- Se plantean las especificaciones del circuito a implementarse. Esto, generalmente, se hace en un papel y forma parte del diseño. Por ejemplo, si va a implementarse un contador de décadas (BCD), se define de cuántos bits consta, si se necesita un reset asincrónico o sincrónico, etc.
- Se ingresan esos datos al programa, ya sea en forma gráfica (dibujando un esquemático) o en forma textual (empleando algún lenguaje HDL).
- Se compila el proyecto, a fin de que el programa lo sintetice como lógica dentro del chip. Este paso es muy importante y puede dar información referente a posibles errores en el diseño (por ejemplo, alertarnos de entradas o salidas no definidas, de partes del circuito que -por mal diseño- nunca van a cambiar con las señales especificadas, etc.).
- Una vez que se ha pasado con éxito la etapa de compilación, se procede a realizar la simulación temporal. Para esto, se ingresan a un gráfico de tiempos, las señales de entradas y las de salida, dándoles valores a las primeras; al correr la simulación aparecen las formas de onda que corresponden a las señales de salida.
- Si todo sale bien, es decir, si el gráfico responde a lo esperado, se puede pasar a la última etapa. Si los resultados no son los esperados, entonces se debe volver al paso primero y replantearse el diseño para ver donde está el error.
- El último paso es el de programación. Seleccionado el chip a emplear, se conecta a la PC, ya sea a través de una interfaz serie (vía puerto RS-232) o paralelo (vía el puerto paralelo, como en nuestro caso).
- Se prueba el diseño con el chip ya inserto en el circuito definitivo.

Especificación del diseño

Explicaremos cómo realizar la entrada del diseño en forma gráfica.

> Cada vez que se mencione la frase "hacer clic o doble clic" se refiere al uso del botón izquierdo del mouse. Cuando se requiera usar el botón derecho, se menciona expresamente esa frase.



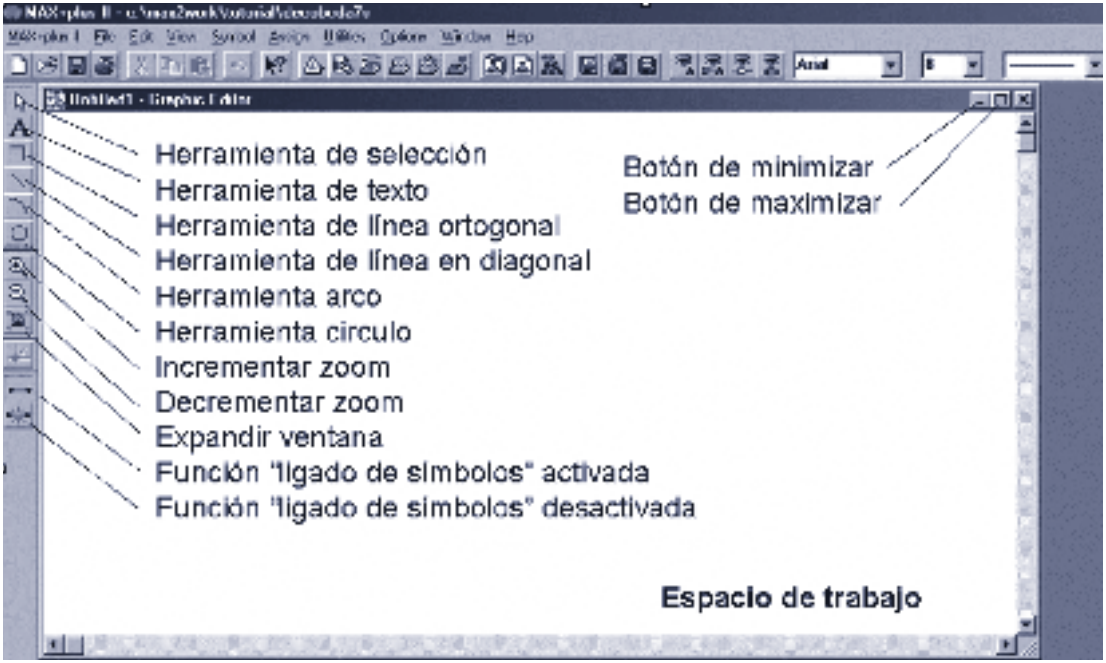
1. **Crear un nuevo archivo**. Creamos un proyecto que se llama "decbcda7s.gdf", en alusión a un decodificador BCD a 7 segmentos.

   1.1. En *File* -archivo- elija *New* -nuevo-.

   1.2. Elija *Graphic Editor File* -archivo de editor gráfico-.

   1.3. Seleccione la extensión ".gdf" en el menú desplegable.

   1.4. Elija OK. Se abre una ventana sin título.



1.5. Si es necesario, puede maximizar la ventana del diseño gráfico.

1.6. Para salvar el archivo, en *File*, elija *Save As* -salvar como- y, en *File*, *Name* -nombre del archivo-; en este caso, es "decbcda7s".

2. **Especificar el nombre del proyecto**. En el MAX+PLUS II se pauta un archivo de diseño -como su proyecto actual, antes de poder realizar una compilación (proceso por el cual el programa sintetiza la lógica necesaria en base a la información que le dio el usuario)-.

2.1. Elija, en el menú, *File*, la opción *Project Name*, donde aparece una ventana de diálogo.

2.2. Seleccione el archivo "decbcda7s".

2.3. Oprima *OK*. En la barra superior de la ventana principal aparece el nombre del proyecto (por ejeplo: c:/max2work/tutorial/decbcda7s).

3. **Seleccionar una herramienta gráfica**. Es posible seleccionar diferentes tipos de herramientas para dibujo -en particular, para el puntero-. Por ejemplo, haciendo

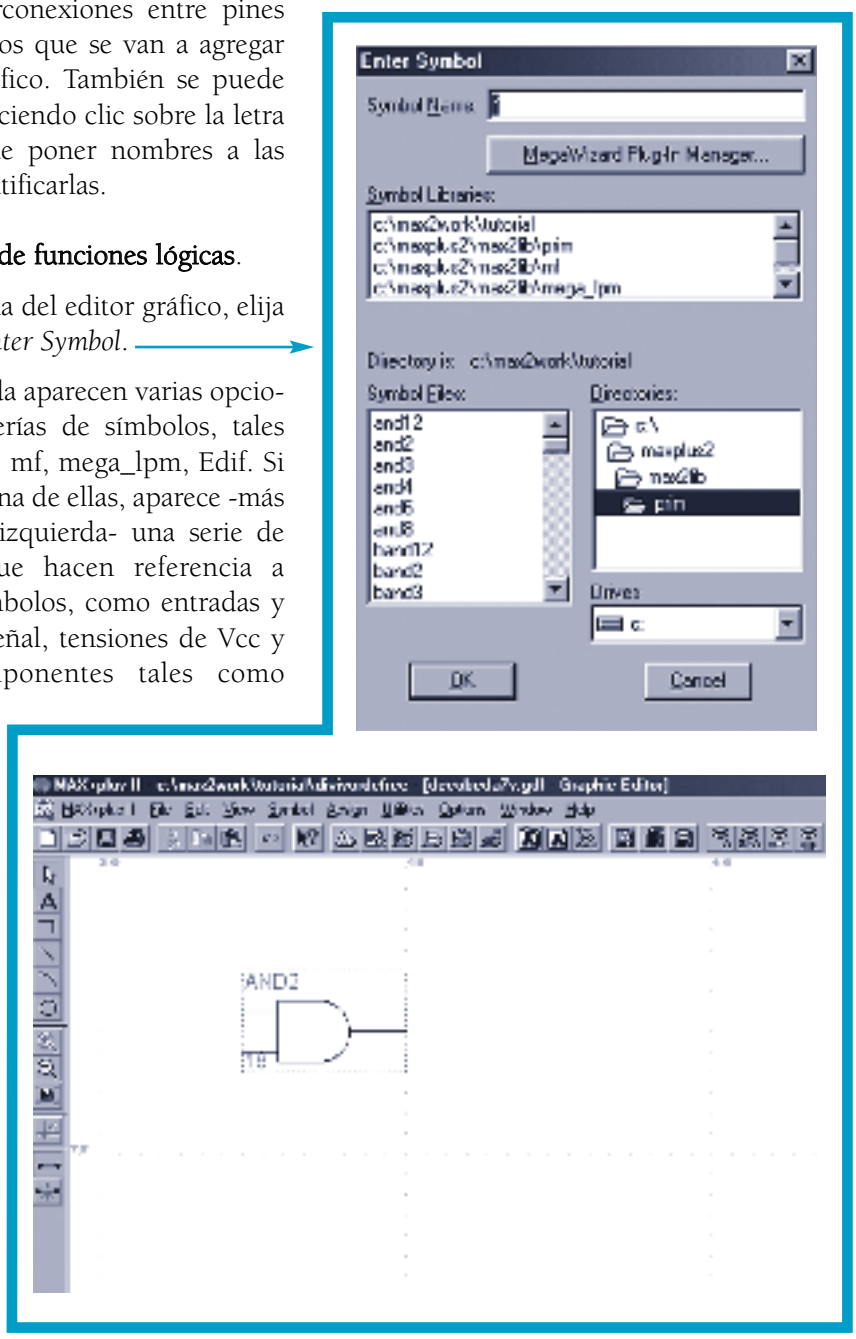
clic en el icono *Orthogonal Line Tool*, el puntero se convierte en una cruz y se puede dibujar una línea; esto sirve para realizar las interconexiones entre pines de los dispositivos que se van a agregar en el diseño gráfico. También se puede agregar texto, haciendo clic sobre la letra "A"; esto permite poner nombres a las líneas y, así, identificarlas.

4. Entrar símbolos de funciones lógicas.

4.1. En la ventana del editor gráfico, elija la opción *Enter Symbol*.

4.2. A la izquierda aparecen varias opciones de librerías de símbolos, tales como: prim, mf, mega_lpm, Edif. Si selecciona una de ellas, aparece -más abajo, a la izquierda- una serie de nombres que hacen referencia a diversos símbolos, como entradas y salidas de señal, tensiones de Vcc y GND, componentes tales como compuertas simples (and, or, etc.) o dispositivos complejos (decodificadores, sumadores, contadores, etc.).

La librería de símbolos "prim" contempla los símbolos básicos de compuertas, pines de entrada, salida y entrada-salida, como así también las etiquetas de "Vcc" (que equivale a un 1 lógico) y "GND" (que equivale a un 0 lógico).

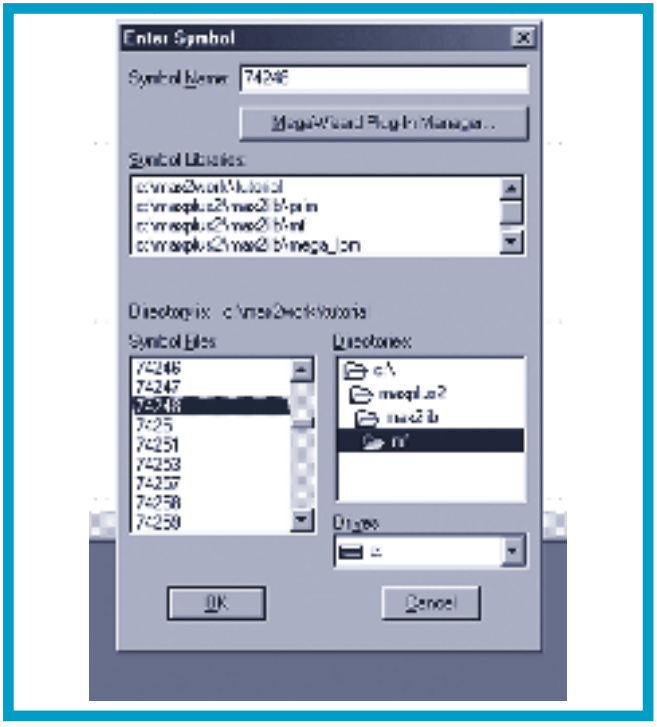


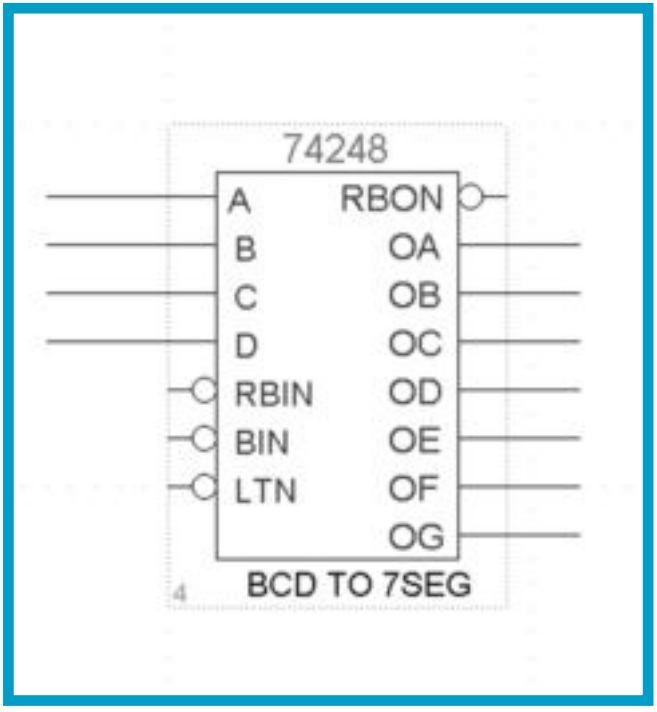
La librería de símbolos "mf" -megafunciones- contempla, fundamentalmente, componentes que figuran en los manuales TTL, como son 7400 (*nand* cuádruple de 2 entradas), 74161 (contador binario de 4 bits), etc.

La librería "mega_lpm" -componentes parametrizados- es muy poderosa, ya que contempla la inclusión en el diseño de componentes que pueden ser armados "a medida". Por ejemplo, un contador en esta librería llamado "lpm_counter" se puede configurar como uno quiera; es decir, es posible definir la cantidad de bits que se necesiten, si va a contar en binario o BCD, si va tener o no entrada de RESET, ya sea asincrónica o sincrónica, etc.

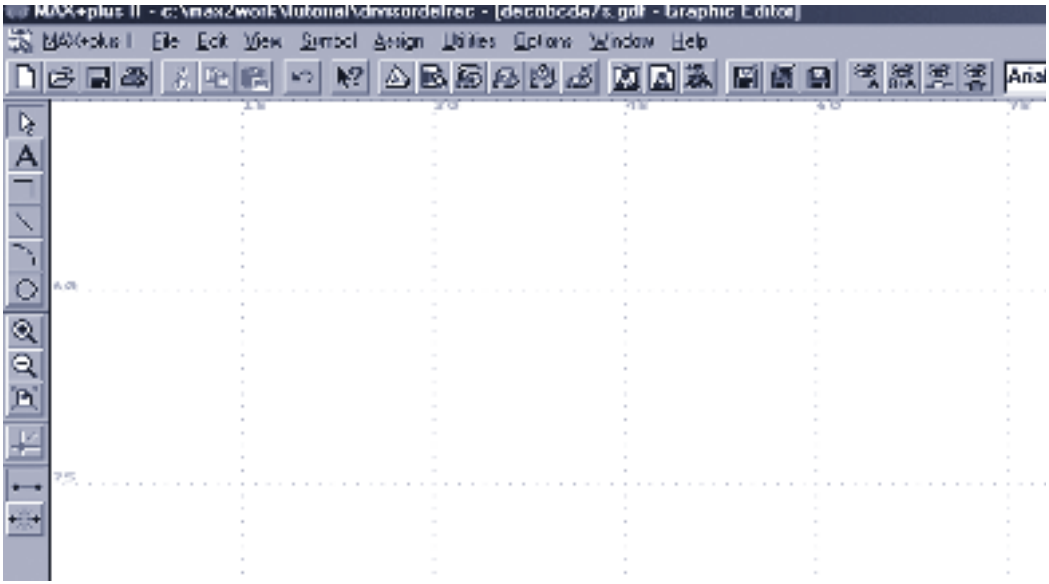
Lo mismo vale para otros componentes: sumador, multiplexer, decodificador, latch, etc.

Para este ejemplo, hemos decidido seleccionar el decodificador 74248 de la librería "mf que tiene las mismas características de funcionamiento que el chip comercial de lógica TTL denominado 74248. Para acceder a él, entramos en *Enter Symbol* y, eligiendo la librería "mf", optamos por "74248" en el menú desplegable y hacemos clic.





5. **Ajustar y mostrar líneas de guía**. Usted puede ubicar líneas horizontales y verticales que sirven de guía (grilla). Para ello, en el menú *Options* vaya a *Guideline Spacing* y, por ejemplo, ponga tanto en "X" como en "Y" en número 15; se forma, entonces, una grilla con cuadrados delineados por líneas de puntos de 15 x 15 unidades.

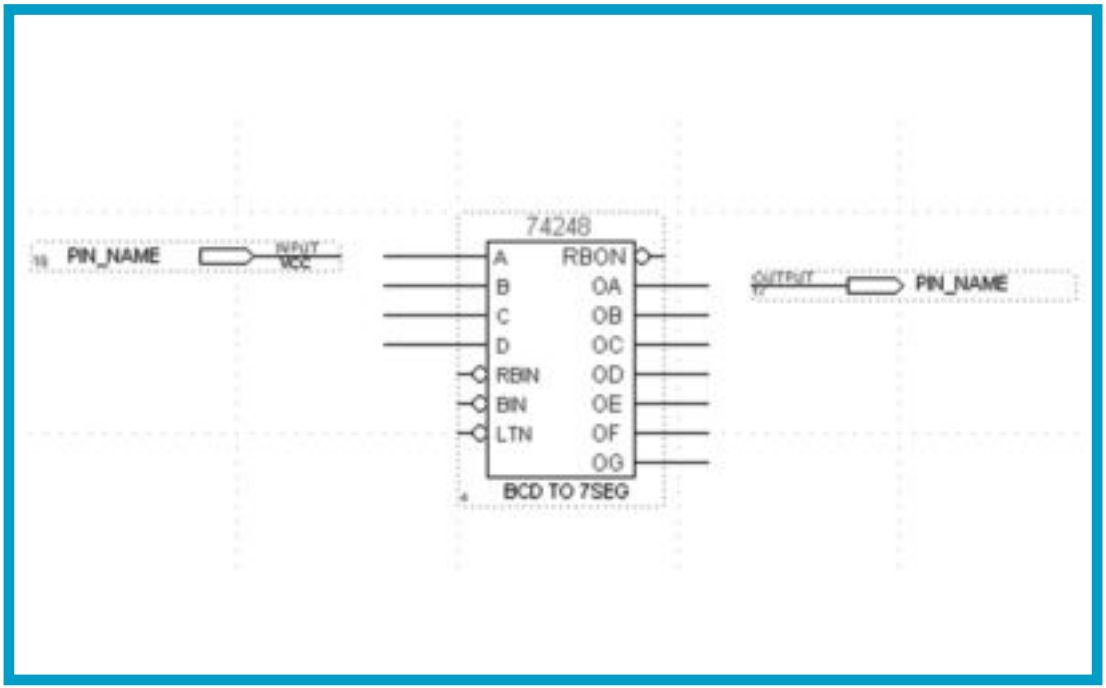


6. **Mover un símbolo**. Posicione el puntero sobre el símbolo y, presionando el botón izquierdo del mouse, muévalo; el símbolo seleccionado se va corriendo hasta el lugar que se elija.

7. **Entrar pines de entrada y de salida**. Vaya a *Enter Symbol* y, en /prim, seleccione *Input* -entrada-. Al hacer clic en *Input*, aparece en la pantalla el correspondiente símbolo.

Lo mismo, para entrar un pin de salida: En /prim, busque la opción *Output* -salida- y haga clic. En el área de trabajo aparece el símbolo correspondiente.

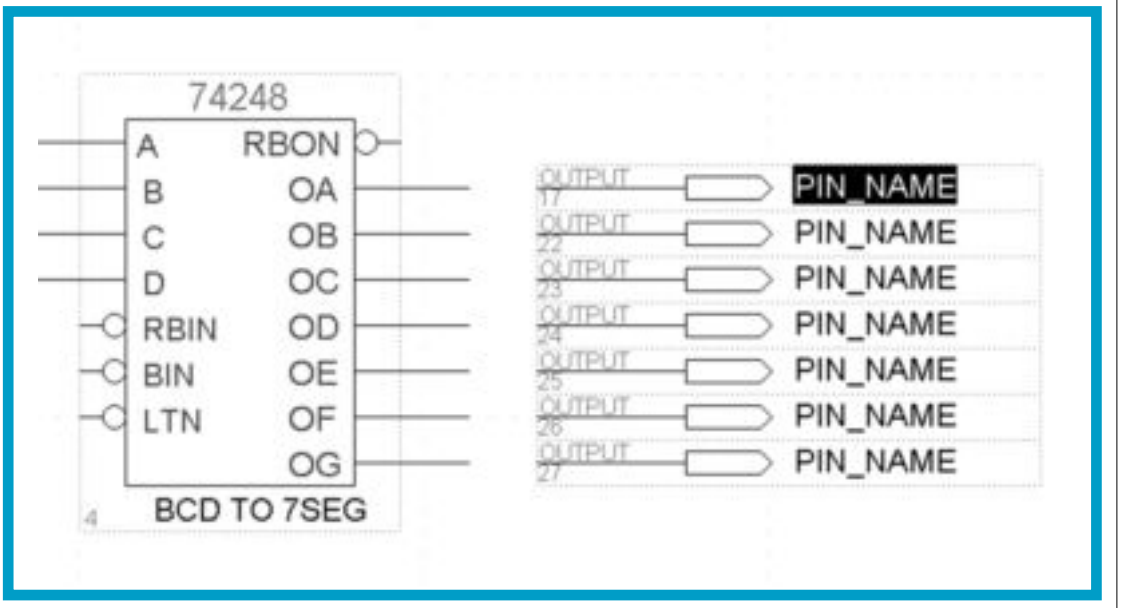
Ambos se ven así:



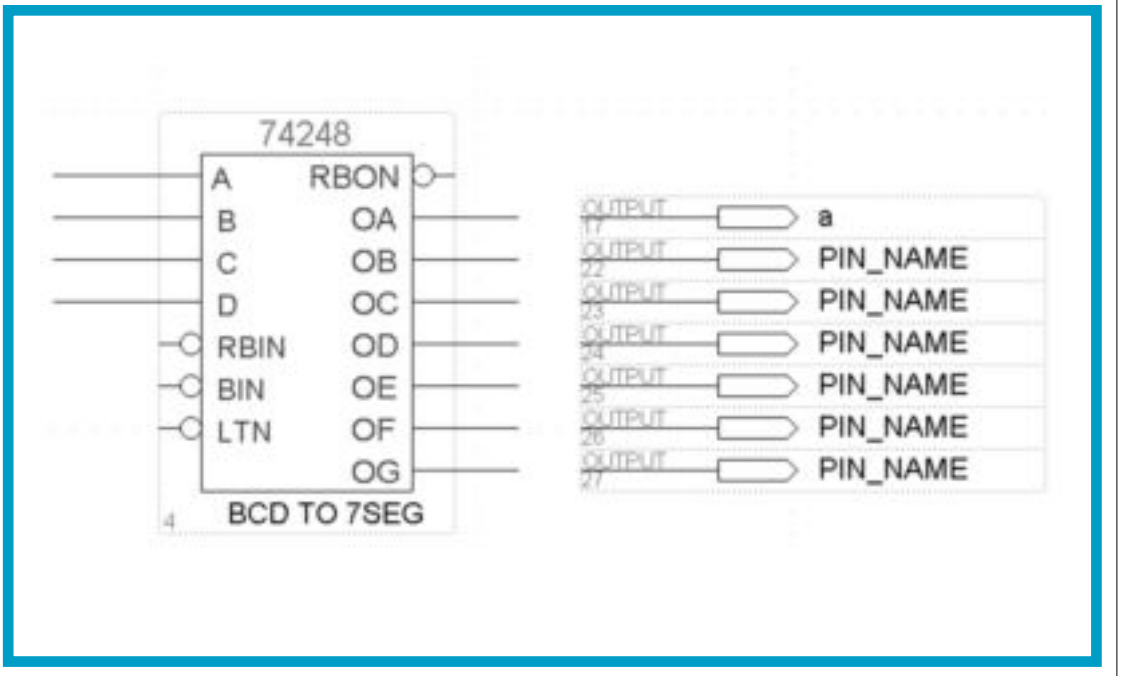
Esta operación se repite las veces que sean necesarias (en este caso, tenemos 4 entradas y 7 salidas).

Una forma más práctica es la de entrar sólo una entrada y una salida, y luego usar las herramientas de *Copy* y *Paste* -copiar y pegar-, a las que puede usted acceder desde el menú *Edit* -edición- o desde la barra superior de herramientas. De esta manera, por ejemplo, si hace clic sobre el símbolo del pin de entrada, éste queda recuadrado en color rojo; en esta condición, si selecciona *Copy*, queda cargado en la memoria y, luego, basta sólo posicionar el puntero en donde quiera que aparezca una copia e indicar *Paste*. Puede realizar la misma operación con el símbolo del pin de salida o con cualquier otro objeto que desee reproducir.

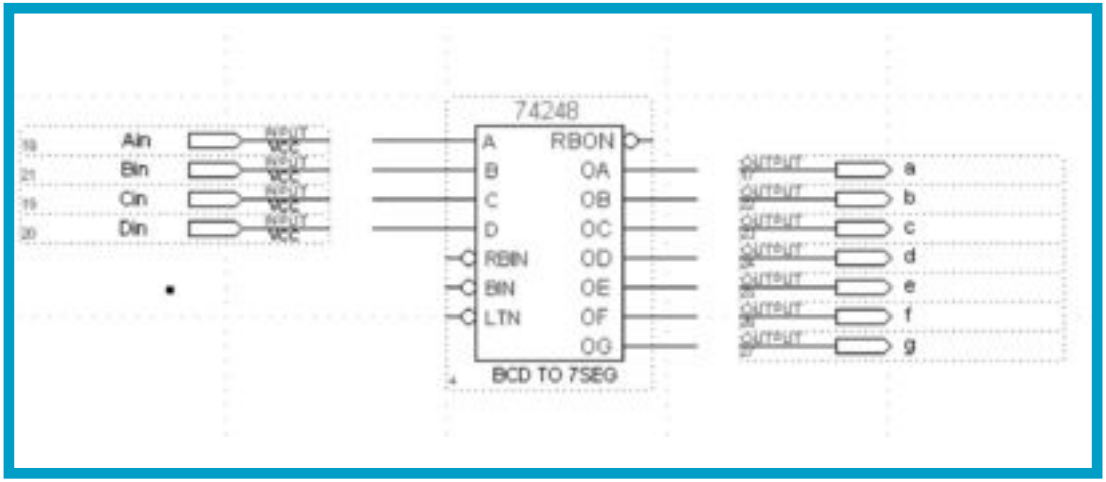
8. **Nombrar pines**. Posicione el cursor sobre Pin_name y haga doble clic; el cartel se ha puesto negro y usted puede cambiar el nombre por el que elija. Como regla, el nombre siempre empieza con una letra.



La primera salida del decodificador queda, entonces:

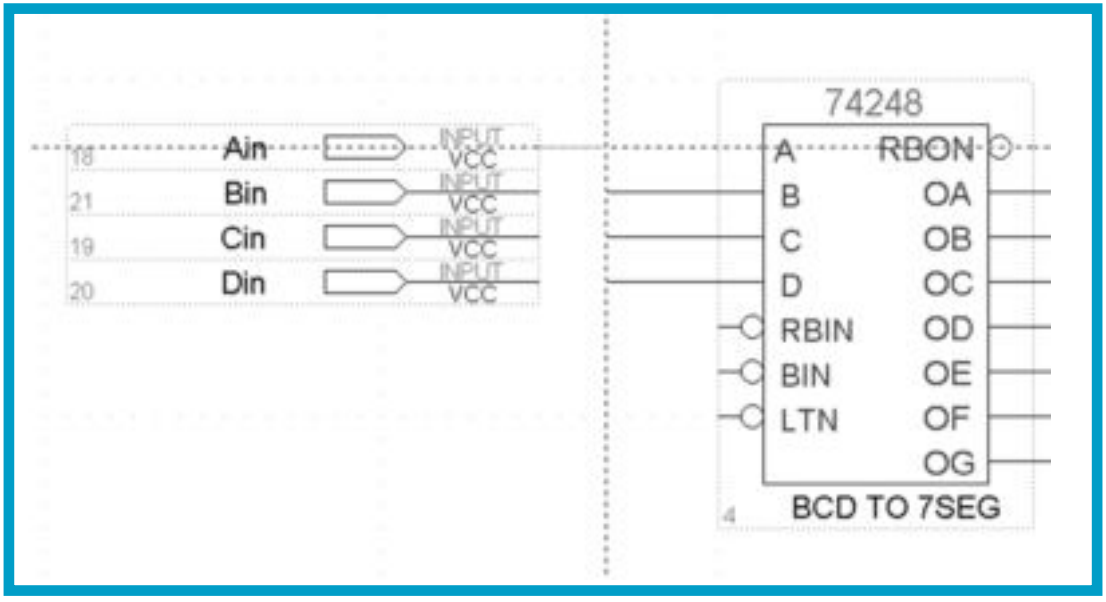


Y, el esquema completo:

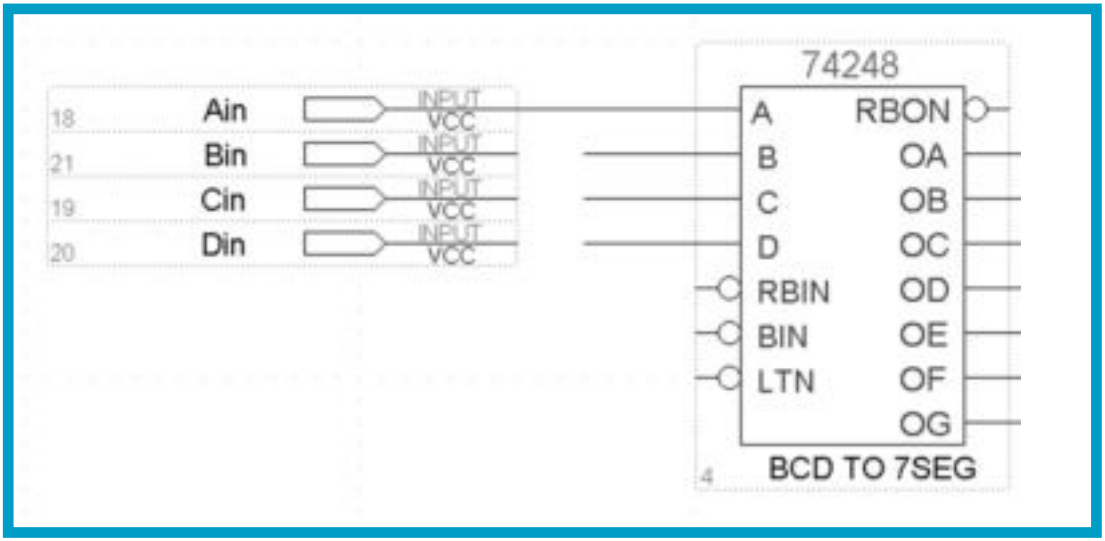


9. **Efectuar la conexión entre símbolos**. Desde la barra izquierda de herramientas, presione el ícono con la flecha. Las conexiones entre símbolos se hacen haciendo clic con el botón izquierdo del mouse y, desde un terminal, arrastrando el puntero hasta el otro terminal que se quiere unir.

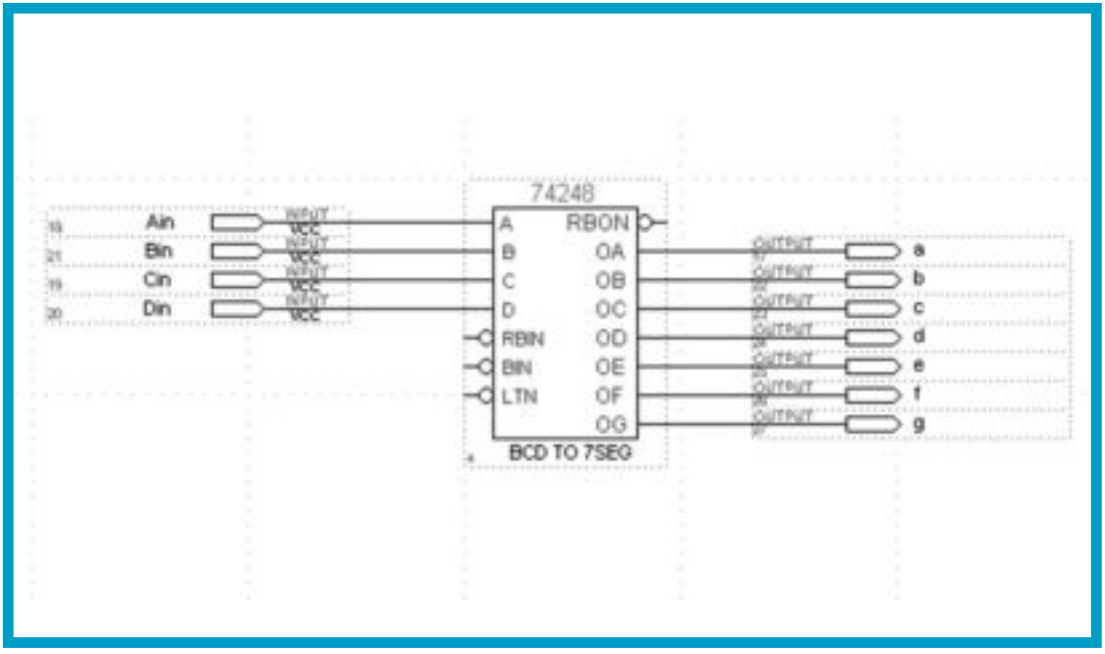
Por ejemplo, la unión de la entrada A comienza así:



Y, termina así:



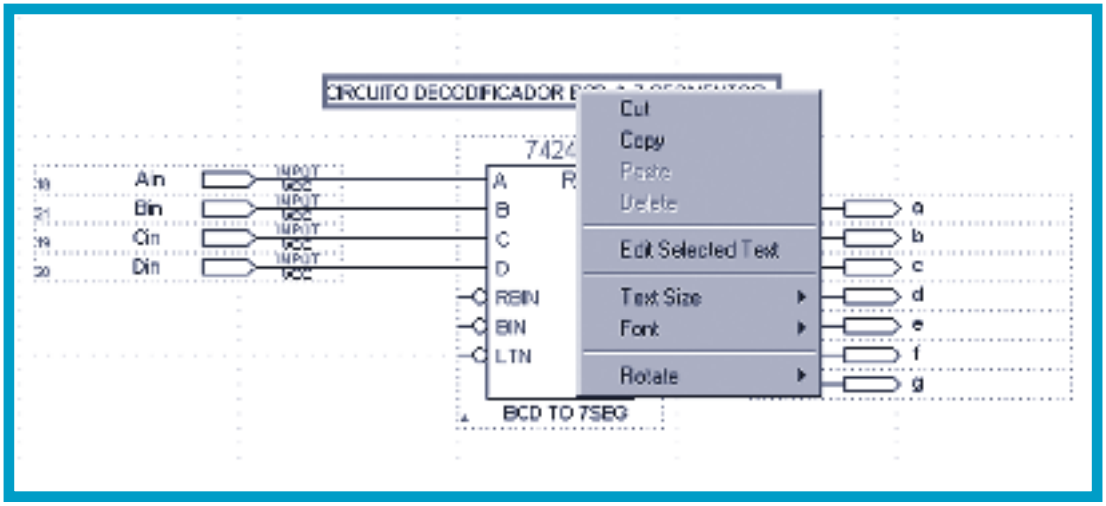
El circuito completamente conectado, queda:



Para escribir algún texto, acceda a la herramienta *Texto* que está ubicada a la izquierda de la ventana de trabajo. Haciendo clic en ícono "A", puede escribir a partir de donde está posicionado el cursor. En caso de querer cambiar el tipo de letra, tamaño o necesitar girar el texto 90°, puede posicionarse en él y, con la herramienta de *Selección*, hacer clic con el botón derecho del mouse.

10. **Conectar nodos y buses por nombres**.

11. **Salvar el archivo de trabajo y chequear errores**. Para guardar el diseño se debe ir a *File* y a *Save* -guardar-, donde aparece una ventana que permite almacenar los últimos cambios realizados. Otra opción es hacer clic en el icono del disquete que está sobre la barra superior.



Existen dos comandos más en el menú de herramientas de la barra de la izquierda de la ventana de trabajo; corresponden a dos íconos denominados "rubberbanding function on" y "rubberbanding function off". Conforman una herramienta que, si está activada (primera opción), permite que al moverse un símbolo, las conexiones asociadas a él se mantengan; es decir, las líneas de conexión se modifican para que siga conectado con los otros componentes. En cambio, si la opción está en "off", al mover el objeto, sólo éste se cambia de lugar, quedando el resto del circuito tal como estaba antes de moverlo.

12. **Crear un símbolo por default -defecto-**.

13. **Cerrar archivo**. En *File*, haga clic sobre la opción *Close* -cerrar-.

Procesamiento (compilación) del diseño.

Hasta aquí hemos especificado las tareas necesarias para realizar la entrada del diseño. Consideremos, ahora, cómo efectuar la **compilación del proyecto**.

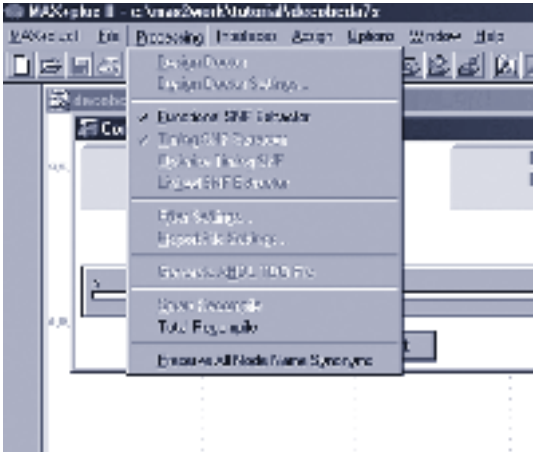
Para realizar la compilación, abrimos la ventana del proyecto que queremos sintetizar

(en este caso, el denominado "decobcda7s.gdf").

Dentro del ambiente de diseño del MAX+PLUS II, podemos acceder a los programas de **compilación**, **simulación** y **programación** del proyecto, empleando los íconos que aparecen en la barra vertical superior o accediendo al menú de la izquierda que dice "MAX*PLUS II.

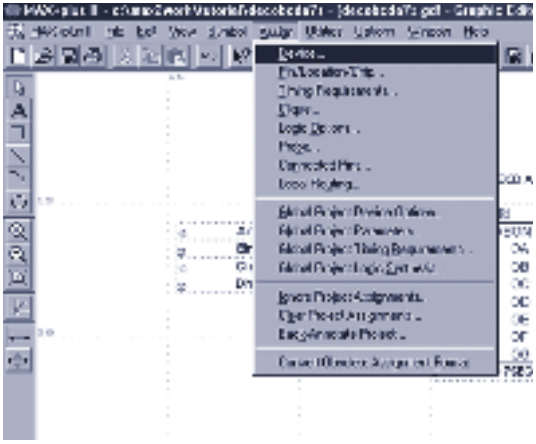
Se pueden realizar dos compilaciones diferentes: una funcional y otra total.

La **compilación funcional** *-Functional SNF Extractor*; extractor SNF funcional- permite sintetizar la lógica requerida para el proyecto pero no tiene en cuenta ningún requerimiento temporal; es decir, no optimiza la velocidad del circuito. Si bien esta opción sirve para una primera aproximación -para saber si el funcionamiento del circuito es el correcto-, es bastante rápida en implementarse.



Una vez que no hay errores en la especificación del diseño, se puede pasar a la opción de **compilación total o completa** *-Timing* SNF[27] *Extractor*; extractor SNF con información de temporización-. Esta opción tiene en cuenta, por ejemplo, qué tipo de chip se va a usar, ya que existen muchas familias de dispositivos SPLD, CPLD y FPGA, cada una con diferente performance y velocidad de respuesta. Es necesario, entonces, asignar al proyecto la familia de dispositivo que se quiere usar y, dentro de ella, especificar de qué modelo se trata.

Dentro del menú *Assign* -asignación-, está la opción *Device* -dispositivo- desde donde usted puede seleccionar qué familia de chip[28] va a emplear:



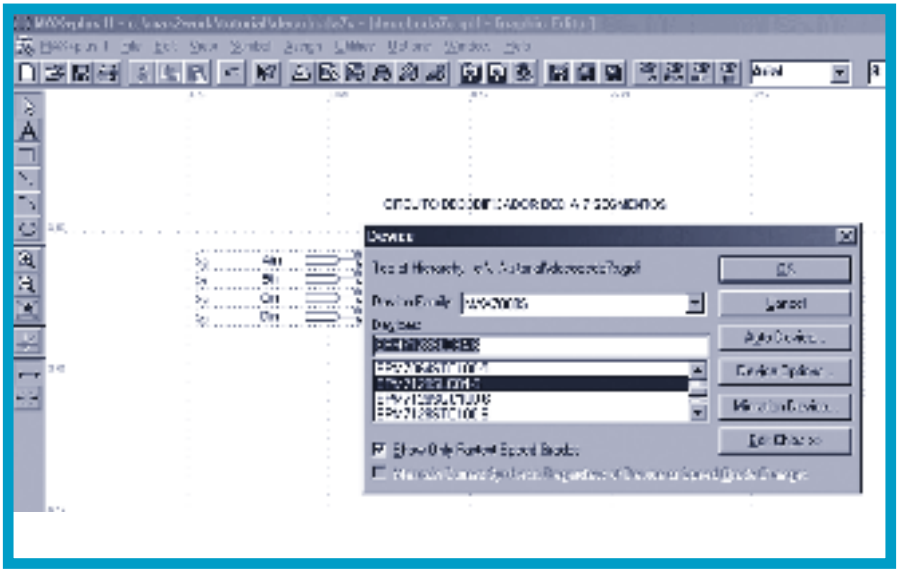
En la ventana desplegable *Device Family*, seleccione la MAX7000S.

---

[27] SNF *-Simulator Netlist File-* es un archivo de listado del simulador que contiene toda la información requerida para sintetizar el circuito especificado.

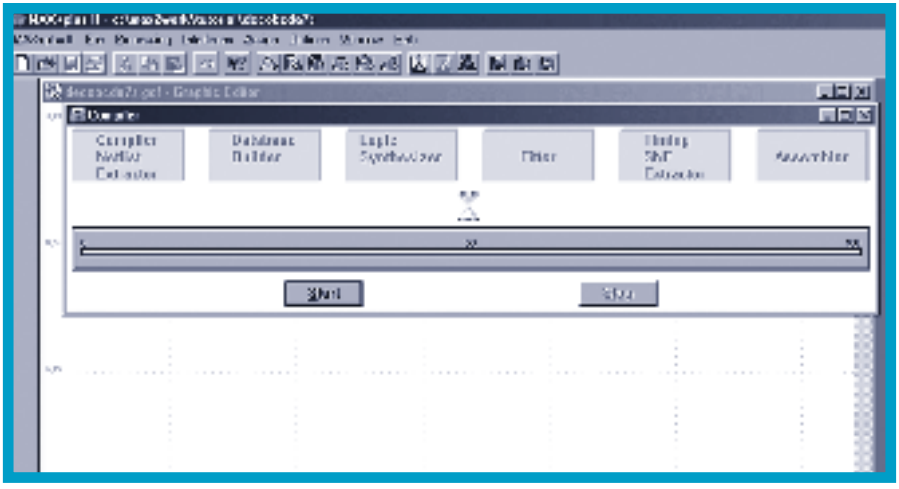
[28] Puede haber variaciones en el chip que se consiga luego del guión (en lugar de 6 puede aparecer 15); esto se refiere a una especificación de velocidad del chip que no es relevante, por ahora, para lo que estamos explicando. Dichas opciones se encuentran en el menú *Processing* -procesamiento- que aparece una vez que se hace clic sobre el icono *Compiler*.

Y, dentro de la familia de dispositivos CPLD MAX7000S, elija el chip modelo EPM7128SLC84-6 que es el que le proponemos usar en el entrenador.
Una vez seleccionado, presione el botón de OK.



Ya designado el dispositivo, el paso siguiente es comenzar la compilación. Para ello, presione en *Start*.
Si todo está correcto, aparece una ventana con un aviso de 0 *errors* y 0 *warnings* -ningún mensaje de error y ninguno de advertencia-.

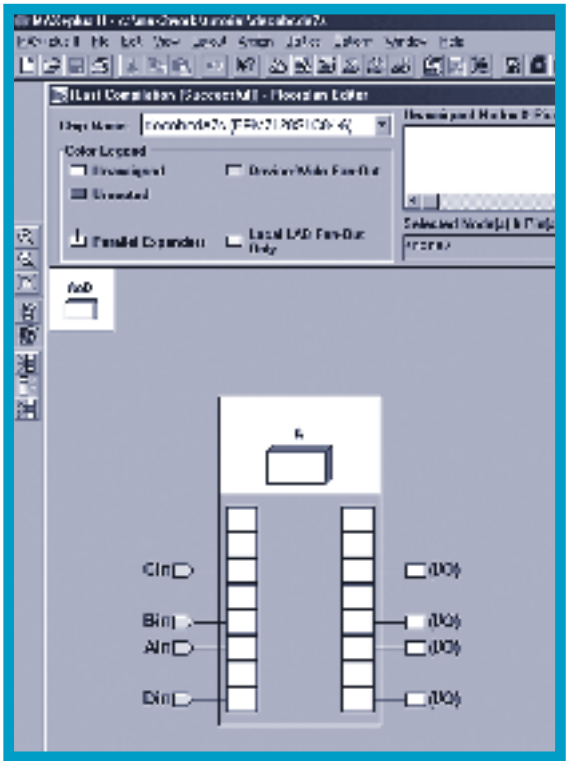
El mensaje de advertencia no es un mensaje de error; sólo indica alguna condición que el usuario puede tener en cuenta o no (por ejemplo, que no asignó ningún chip y que el compilador lo hizo por sí solo). Los mensajes de error, en cambio, sí son para considerar, ya que la síntesis no se puede realizar hasta no tener ningún error. En esas condiciones es imposible simular y, menos, programar el chip hasta eliminar por completo todos los errores que aparezcan.



Si usted hace clic en los símbolos etiquetados como CNF, RPT, SNF y POF, se abren archivos que contiene información sobre la compilación realizada. En particular, para nuestro pro-

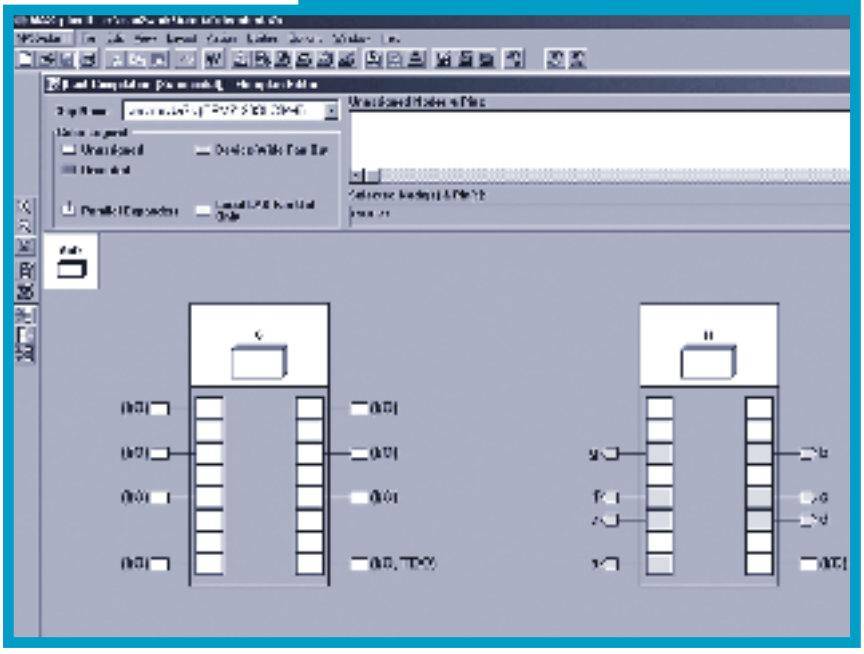
yecto, el icono RPT abre el archivo de texto de reporte "decobcda7s.rpt" que contiene información de cómo se realizó la compilación; generalmente, es un archivo extenso que brinda datos de cuántas y cuáles macroceldas se utilizaron, cuántos y cuáles pines se emplearon, qué espacio queda disponible todavía, etc. (El archivo denominado "decobcda7s.pof" contiene información necesaria para poder, luego, programar al chip).

Dentro del ambiente de desarrollo gráfico, existe también una parte denominada *Floorplan Editor* que es un editor gráfico que muestra la disposición física de las macroceldas y de los pines dentro del dispositivo. Allí se encuentra información que sirve de complemento al archivo de reporte en el que se encuentran marcados los componentes que se han sintetizado luego de la compilación.

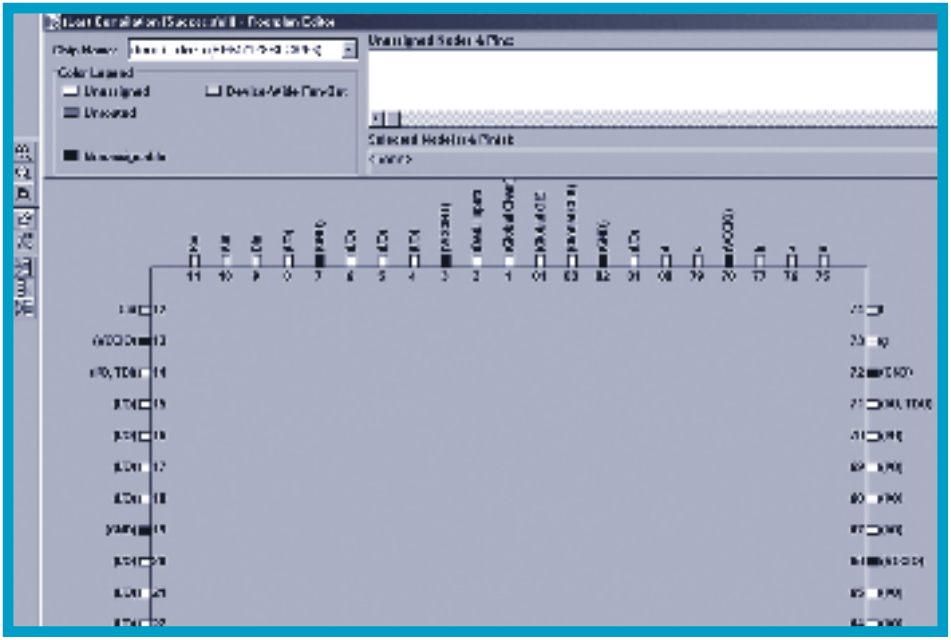


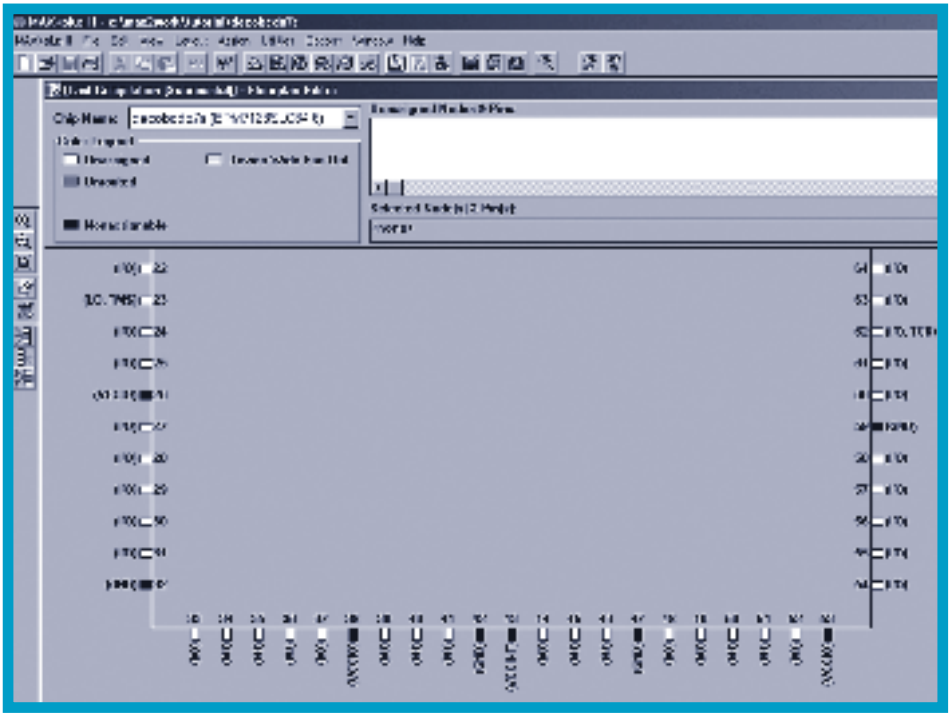
Al entrar a este editor aparecen las macroceldas agrupadas de a 8. Cada bloque forma el LAB -*Logic Block Array*; arreglo de bloques lógicos- que ya hemos descrito en apartados anteriores.

En las siguientes dos figuras podemos ver aquellas macroceldas que han sido utilizadas para este proyecto:



Si usted hace doble clic sobre esta ventana, pasa a otro modo de ver el chip que muestra la disposición física de los pines con sus designaciones respectivas.



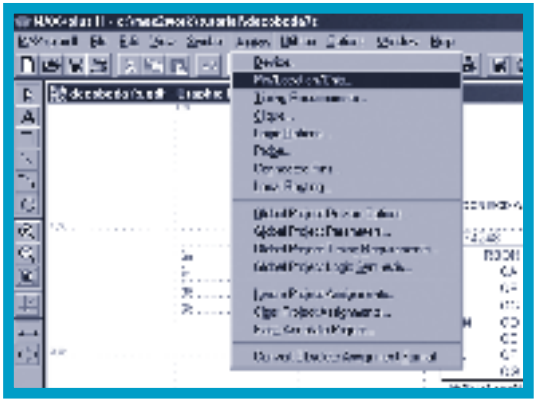


Hasta aquí hemos explicado cómo se puede compilar un proyecto para que el programa del chip sintetice las funciones requeridas según los datos entrados en forma gráfica; y hemos planteado cómo se puede asignar un modelo dentro de una familia de dispositivos lógicos programables.

El siguiente paso para tener un diseño completamente flexible es el de asignar a cada una de las entradas y salidas del proyecto un número de pin en el encapsulado del chip.

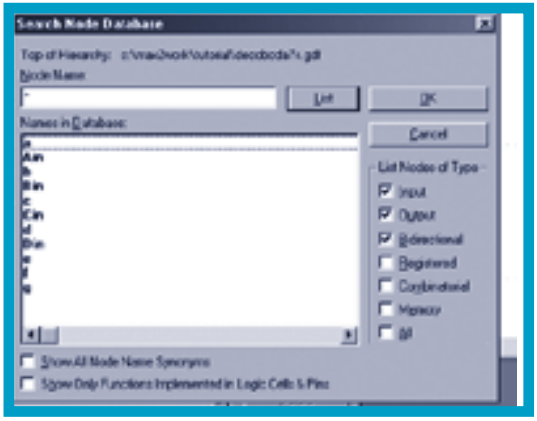
En nuestro caso, tenemos 4 entradas y 7 salidas.

Para comenzar las asignaciones, entre en el menú *Assign* y haga clic en la opción *Pin/Location/Chip*



Aparece una ventana de diálogo. Presione el botón *Search* -búsqueda- para encontrar la lista de los nombres que hemos puesto en las entradas y salidas (Ain, Bin, Cin y Din para las entradas; a, b, c, d, e, f y g para las salidas).

Presione en *List* -listado- para que aparezcan:

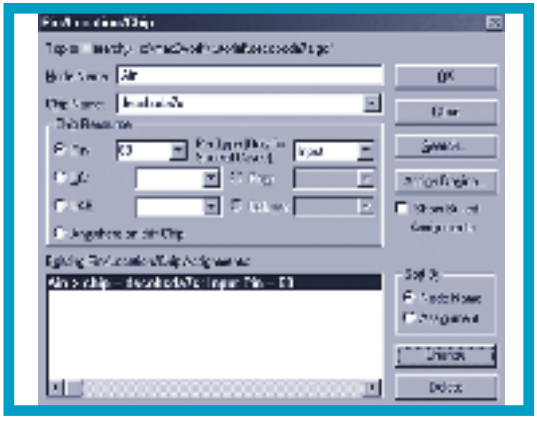


Una vez allí, haga clic sobre, por ejemplo, la entrada Ain y presione OK.

Vuelva a la primera ventana. Ahora, en *Node Name* -nombre del nodo- aparece "Ain".

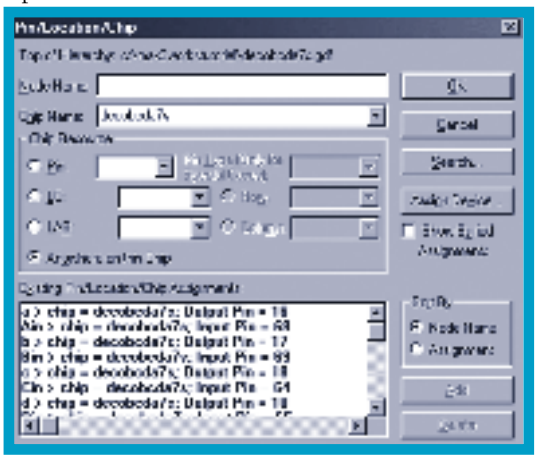
Haga clic en la opción *Pin*. Aparece una ventana desplegable que contiene todos los pines del dispositivo seleccionado (en este caso, el EPM7128SLC84).

Hemos elegido el pin número 68. Luego, presione el botón *Add* -agregar-. En la ventana denominada *Existing Pin/Location/Chip* Assignment aparece la asignación ya hecha de la entrada *Ain* al pin número 68.



Repita esto para cada una de las señales restantes.

En las siguientes dos figuras se resume dicha operación:



Salve el proyecto nuevamente y vuelva a correr el programa de compilación, ya que ahora se han cambiado -por decisión del usuario- las asignaciones realizadas en pasos anteriores.

> Recuerde que cuando hizo por primera vez la compilación, fue el mismo programa el que decidió dónde ubicar las entradas y las salidas.

Verificación del diseño

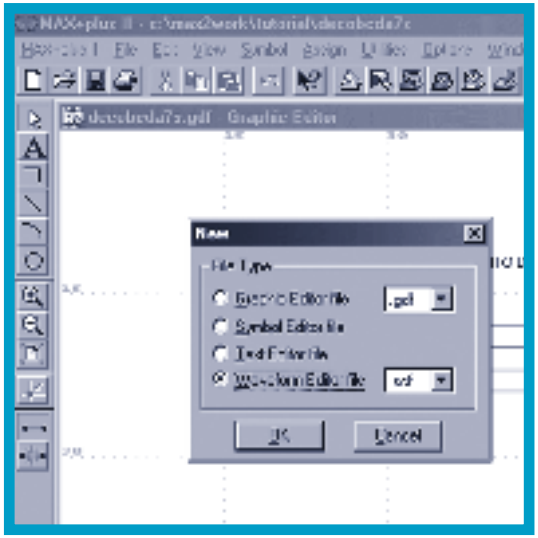
Una vez que tiene el proyecto compilado, puede pasar a la etapa de simulación.

Para esto, haga correr el programa denominado *Simulator* -simulador-. Puede acceder a él a través del menú "MAX+plus II en la opción *Simulator* o haciendo clic sobre el icono correspondiente ubicado en la barra superior horizontal de la hoja de trabajo del proyecto abierto.

Comience creando un archivo de extensión "scf" que, en un gráfico temporal, ubique las señales de entrada y salida para poder simular a estas últimas.

Para esto, en *File* seleccione la opción *New*.

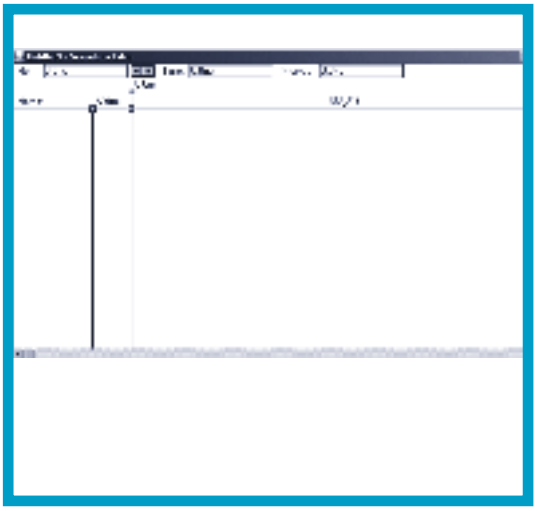
Haciendo clic aparece una ventana; en ella, seleccione *Waveform Editor File* -archivo del editor de formas de onda- y haga clic en OK.



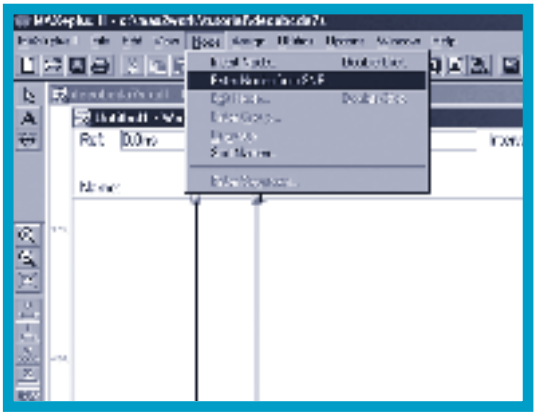
Al hacer esto, aparece una nueva ventana con

un gráfico X-Y; en él, el eje de las abscisas corresponde a tiempo y el de las ordenadas a los estados lógicos de las variables en juego.

Esta ventana aparece con el nombre "untitled1.scf" -sin título-.

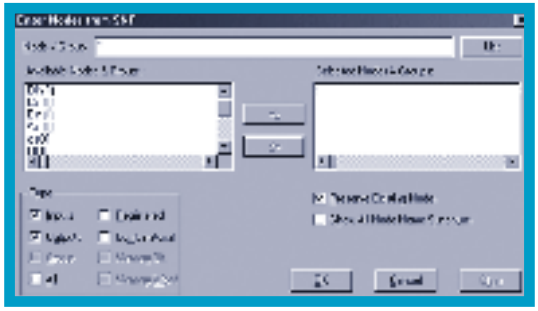


Ahora, ingrese las señales que intervienen en la simulación: 4 entradas y 7 salidas, en nuestro proyecto "decobcda7s". Para esto, en el menú *Node*, elija *Enter Nodes from SNF* -entrar nodos desde SNF, *Simulator Netlist File*-.

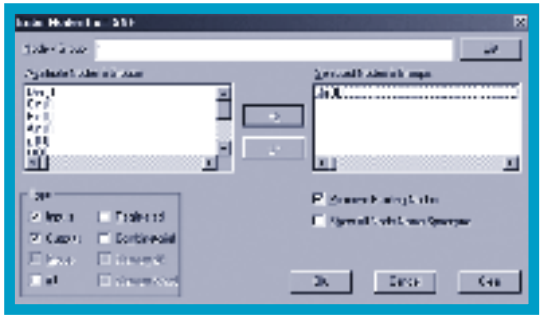


Al hacer esto, aparece una ventana que tiene dos cuadros de diálogo; en el de la izquierda aparecen las variables disponibles en el proyecto -*Available Nodes&Groups*-; en el de la derecha, las variables que uno quiere simular -*Selected Nodes&Groups*-.

Presione en *List*. En el cuadro de la izquierda, aparecen todas las señales disponibles:



Marcando, por ejemplo, la entrada *Din* y haciendo clic en el botón ⇒ en el cuadro de la derecha, aparece la entrada en la simulación.



Repita esto para todas las señales. Cada una de éstas debe aparecer en el cuadro de la derecha.

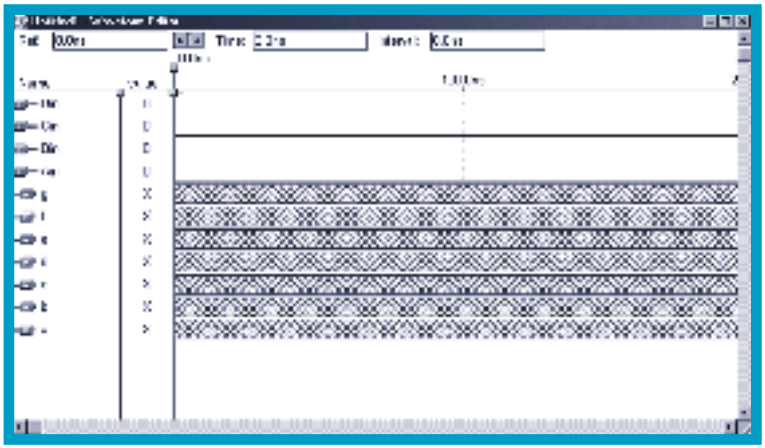
Una manera más rápida es hacer clic sobre una de las señales y arrastrar el cursor hacia abajo y, luego, soltarlo. Así, quedan marcadas en negro las señales requeridas. Luego, pre-

sione en el botón ⇒.

Cuando esta tarea está terminada, marque OK. Aparece, entonces, nuestra ventana inicial pero, ahora, con las 11 señales.

Los gráficos de las salidas se ven rayados; esto se debe a que aún no tienen valores asignados, ya que no se realizó la simulación.

Las entradas por *default* -defecto- están todas en cero a lo largo del gráfico.



Deténgase a analizar el menú File. Allí tiene las opciones normales más otras; por ejemplo:

- *End Time* es la opción que permite entrar el tiempo máximo de simulación, desde 0 segundos en adelante. Por defecto, el programa siempre ajusta este parámetro en 1.0 µs (1 microsegundo).
- Otro menú importante es el *View* -vista-.
- La opción *Fit in Windows* -ajustar pantalla- permite que todo el gráfico entre en la pantalla.
- *Time Range* posibilita seleccionar el tiempo inicial y final que se muestra en pantalla.
- Mediante *Zoom in* -lupa con signo más- y *Zoom out* -lupa con signo menos-, se puede expandir o comprimir el gráfico en el eje de tiempos.
- A través del menú *Node*, entran las señales tanto desde el archivo SNF como de a una variable por vez, empleando la opción *Insert Node* -insertar nodo-.

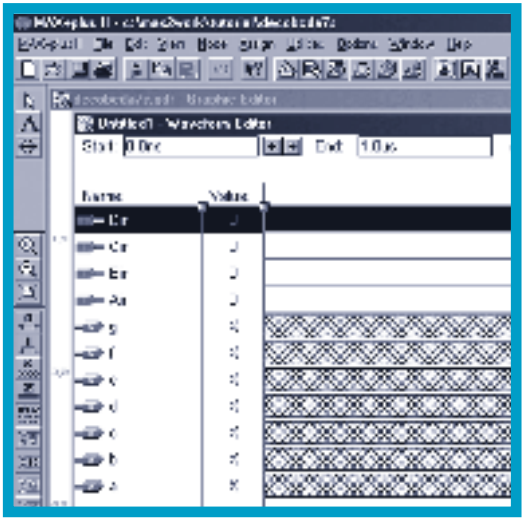
Sigamos con el armado de los gráficos temporales. Teniendo las entradas, es necesario asignarles valores en el tiempo.

Para ello, use la barra de herramientas que está a la izquierda de la ventana de trabajo. Las opciones 0, 1, X, Z son las que definen "0" lógico, "1" lógico, estado no definido y alta impedancia respectivamente.

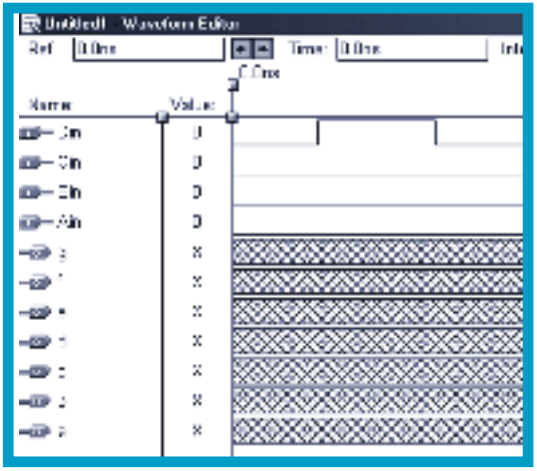
Nosotros aquí sólo usamos las opciones "0" y "1".

Si usted hace clic sobre el nombre *Din*, por ejemplo, el gráfico correspondiente se vuelve negro, lo que significa que puede trabajar sobre él.

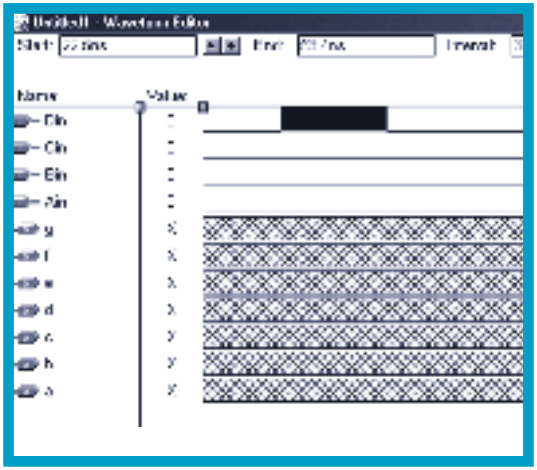
Si, en esta condición, usted presiona el botón "1", queda la señal *Din* en "1" lógico en todo el gráfico. Esto se evidencia con un cambio en el nivel de la línea horizontal.



Para asignar, en el tiempo, un nivel lógico a la entrada *Din*, marque una zona determinada con el cursor del *mouse*; hágalo presionando el botón izquierdo en la zona en que quiere modificar el nivel lógico, mientras corre el cursor horizontalmente. Va a obtener la zona a modificar destacada con negro.



Si, ahora, presiona sobre el botón "1" de la barra vertical de la izquierda, la entrada *Din* cambia su nivel lógico a "alto".



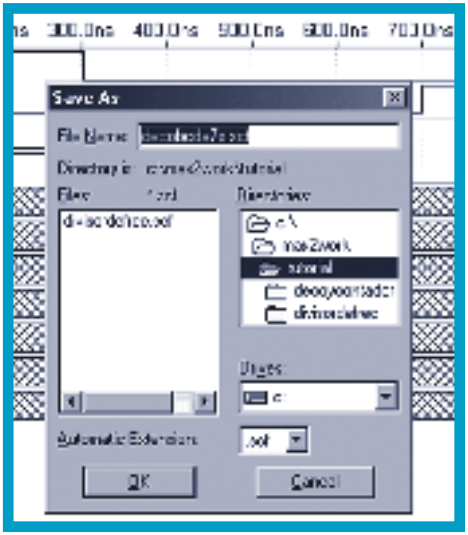
Para observar todo el gráfico -hasta el tiempo final de la simulación marcado como *End time*-, puede ir a *View* y, desde allí, elegir la opción *Fit Windows*. Va a apreciar, entonces, todo el rango de tiempo completo (para este caso, desde 0 segundos hasta 1.0 µs).

Complete el resto de los gráficos de las otras señales de entrada de la misma manera que hizo con *Din*.

Si necesita hacer un cambio de estados "muy fino" y la resolución de la pantalla no es la adecuada, puede hacer un *zoom in* (la lupa con el signo más) y cambiar los niveles con más precisión.

Una vez completados los gráficos de las entradas, salve el archivo con el mismo nombre que tiene el del diseño gráfico (en nuestro caso, "decobcda7s"). Para ello, diríjase al menú *File*.

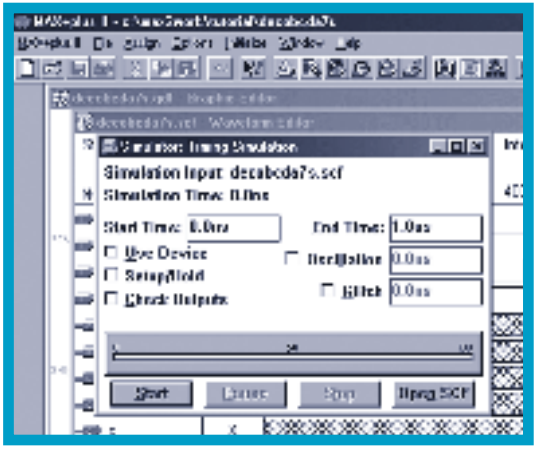
Aparece una ventana de diálogo con el nombre de nuestro proyecto, por defecto. Presionando, registre en *File Name*. Finalmente, dé OK.



Ya está listo para realizar la simulación.

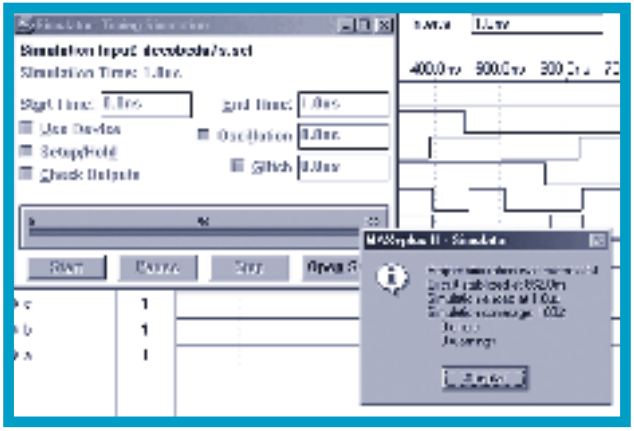
En el menú MAX+plus II, elija la opción *Simulator*.

Al hacer clic, aparece una nueva ventana que se prepara para iniciar con la simulación. Presione, entonces, en *Start*.



Esta ventana cuenta con una barra horizontal que visualiza el grado de avance de la simulación. Cuando ésta ha concluido, aparece un cartel. Si todo resultó correcto, el mensaje indica 0 *errors* y 0 *warnings*.

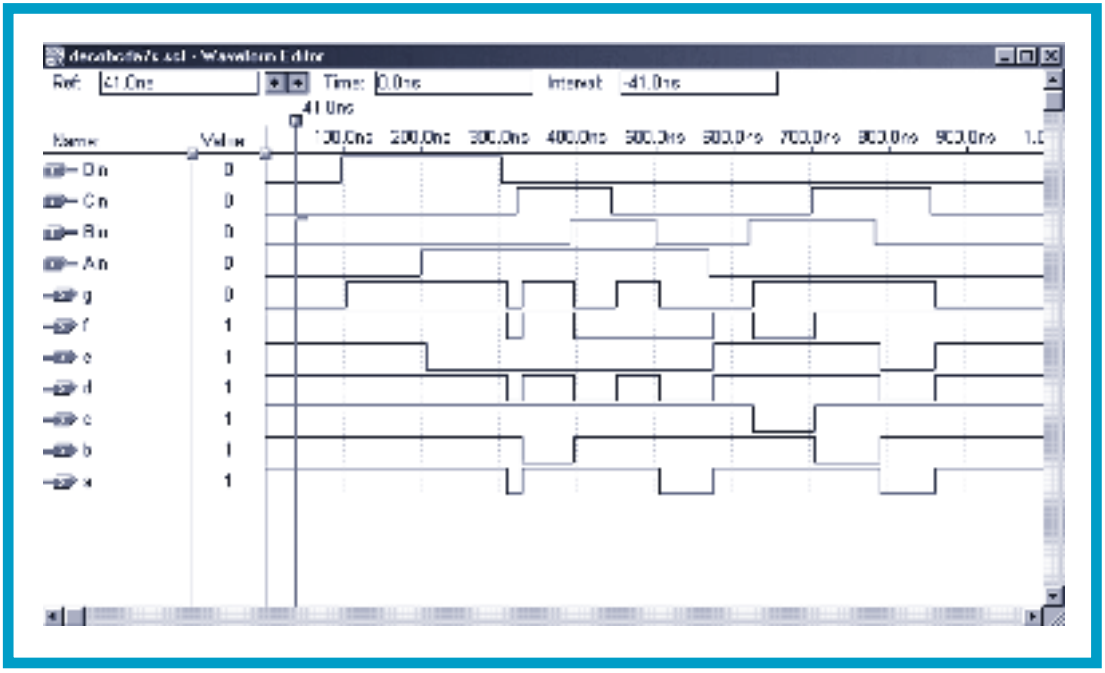
Al igual que cuando usted hizo la compilación, pueden aparecer errores o mensajes de alerta derivados de desajustes cometidos o de los valores que se utilizaron. Un aviso de error puede ser, por ejemplo, porque asignó valores a una variable desde 0 hasta 0.2 µs y el tiempo final de simulación es 1.0 µs -como en este caso-.



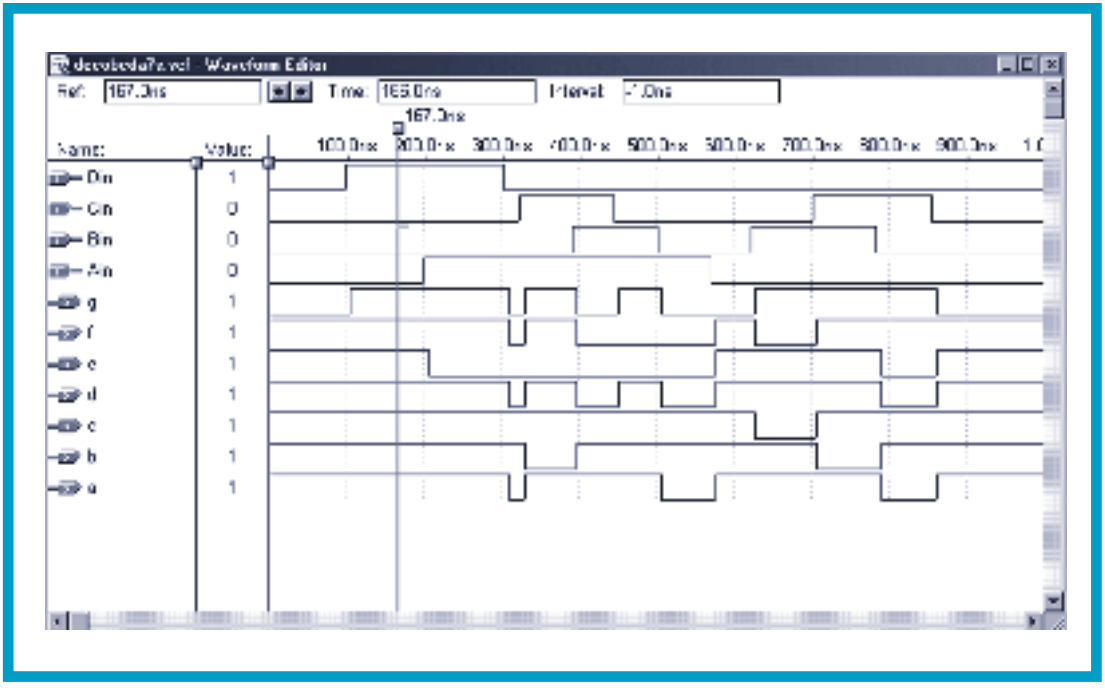
Si todo es correcto, va a ver completados los gráficos de las salidas.

Como usted puede notar, si hace clic sobre una coordenada horizontal, aparece una barra vertical de color azul, indicando a qué tiempo corresponde esa posición (en la figura, a 41 ns).

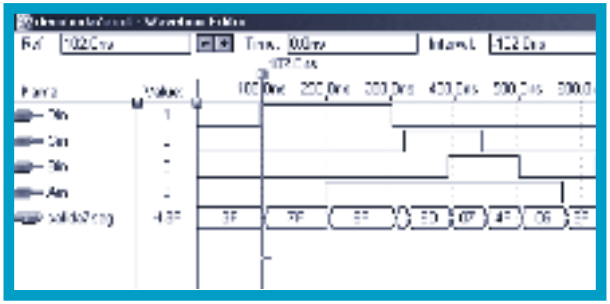
Al lado de cada una de las variables del circuito aparecen valores ("0" y "1") que indican los valores que toman las señales en ese instante.



Cambiando el lugar de esa barra vertical, se cambia la información (en la figura, 167 ns).

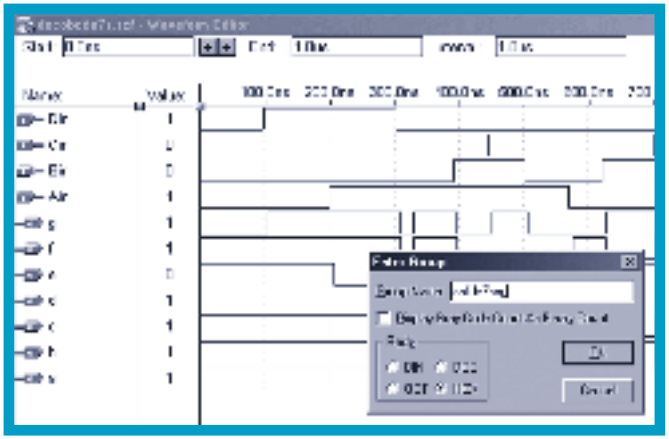


En algunos proyectos puede resultar interesante disponer de una forma más compacta para la representación de señales. Por ejemplo, si usted lo desea, puede agrupar las 7 salidas -que están indicadas en formato binario- en una sola salida con notación en hexadecimal. Para ello, presione el botón del mouse marcando todas las salidas y, luego, libérelo: Quedan las salidas marcadas en negro.



Paso seguido, presione el botón derecho del mouse; aparece un menú. En él, elija *Enter Group* -entrar grupo-.

En la ventana de diálogo que se ve, ingrese un nombre para esta nueva salida combinada (aquí, hemos optado por "salida7seg") y elija el formato de dicho grupo (en el ejemplo, HEX = Notación hexadecimal).



Al hacer esto, va a ver que desaparecen las 7 salidas y que se visualiza una nueva, denominada "salida7seg", con información sobre los valores de las 7 salidas agrupadas.

Si repite lo mismo para las entradas, obtiene un gráfico más compacto que puede ser más práctico a la hora de analizar los resultados de la simulación.

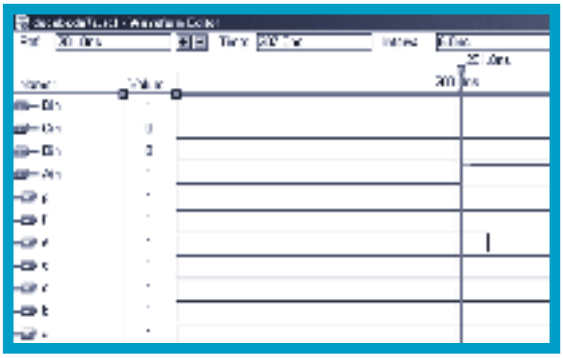
Para revertir el proceso de agrupación de señales, basta con que se pare sobre la señal y, haciendo clic con el botón derecho, seleccione la opción *Ungroup* -desagrupar-.

Una vez que cuenta con el formato deseado, proceda a verificar que los valores lógicos de las salidas que se obtienen en el tiempo sean los correctos, en función de los valores establecidos por las entradas.

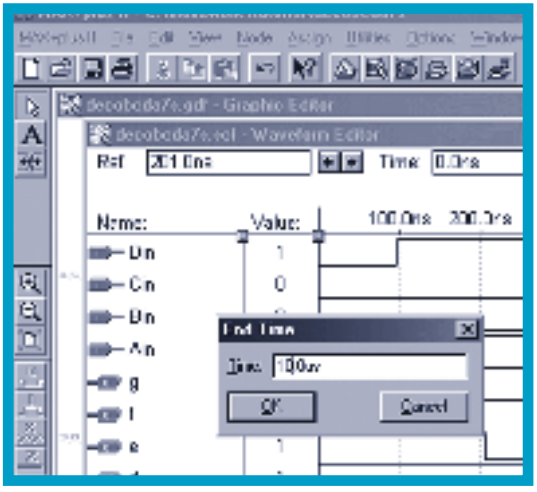
La simulación a través de la opción *Zoom* va a permitirle observar con bastante precisión, los cambios producidos por las salidas ante cambios en las entradas.

A continuación vemos un ejemplo donde, en

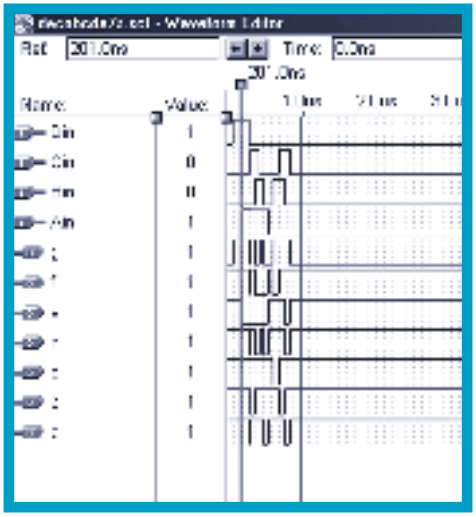
una parte del gráfico, la señal de entrada *Ain* produce un cambio en la salida "e". La barra vertical se ha posicionado en el instante justo del cambio de *Ain* y, si se mueve el cursor hasta el momento en que cambia la señal "e", se puede medir en el cuadro denominado Interval -intervalo- qué tiempo ha transcurrido desde el instante en que *Ain* cambió hasta que lo hizo "e" (en este caso, 6.0 ns).



En las siguientes dos figuras se muestra cómo puede usted hacer para cambiar el tiempo final de la simulación. Desde *File*, al seleccionar la opción *End Time*, aparece una ventana de diálogo; en este caso, ponemos en ella el valor 10.0 µs.



Expandiendo el gráfico desde 0 hasta 10.0 µs, puede usted ver que las señales no están definidas desde 1.0 µs hasta 10.0 µs. Esto es debido a que es necesario correr nuevamente la simulación que paró en 1.0 µs. Además, obviamente, se requiere dar valores a las entradas para este nuevo segmento de tiempo.



Programación de los dispositivos

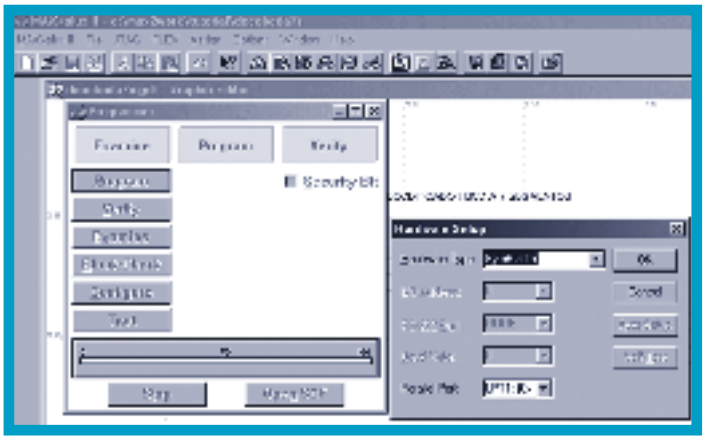
Esta herramienta de diseño tiene como objetivo programar al chip seleccionado, a fin de que pueda sintetizarse la lógica -que fue previamente definida al hacer la compilación del proyecto-.

Para lograr esto, comience accediendo al programa *Programmer*; esta opción se encuentra en el menú "MAX+plusII" y, además, en el ícono correspondiente de la barra horizontal superior.

Al hacer clic sobre esta opción, aparecen dos ventanas: la primera (*programmer*) está aso-

ciada con las operaciones de configuración del chip, y con operaciones de programación y verificación. La segunda (*Hardware Setup*) le permite realizar la selección de la interfaz que empleará para conectar el chip a la computadora personal.

En esta última, elija la opción *ByteBlaster* en *Hardware Type* -tipo de hardware-. Esta interfaz es la que permite trabajar con el puerto paralelo de la PC y es donde va usted a enchufar el cable de programación. Luego de esto, puede dar alimentación al equipo, ya que necesita de tensión para poder ser programado.



Dé OK a la ventana *Hardware Setup*; aparece sólo la ventana de *Programmer*.

Haga clic en el menú *JTAG*; elija la opción *Multi-Device JTAG Chain Setup*.

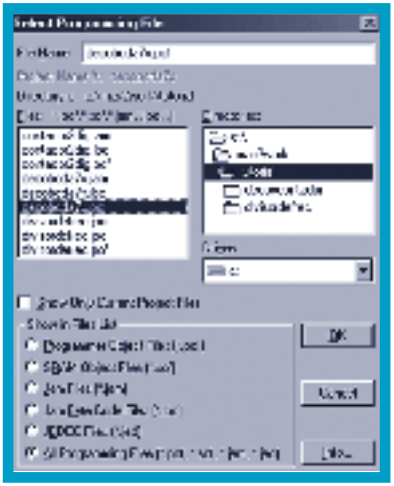
Este paso lo conduce a otra ventana en la que va a especificar el modelo del chip que quiere programar y, además, indicar qué archivo es el que el programador debe cargar en la CPLD. En nuestro caso, el dispositivo es el EPM7128S y el archivo de salida para el programador es el "decobcda7s.pof".

En *Device Name* -nombre del dispositivo-, seleccione la EPM7128S.

Luego, haga clic en *Select Programming File* -selección del archivo de programación-. Aparece una ventana a la derecha con una estructura de subdirectorios. Ubique el que contiene el archivo de programación.

A la izquierda aparece una lista con todos los archivos disponibles.

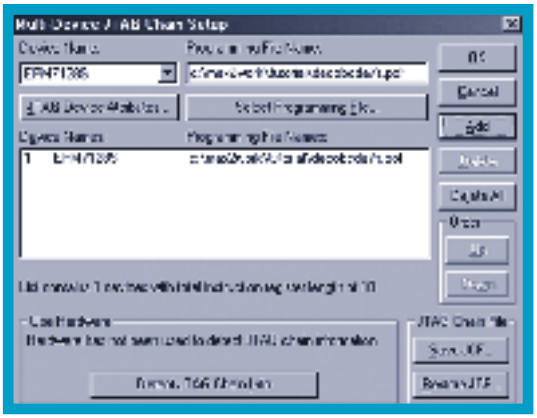
Busque el archivo "decobcda7s en la lista que aparece a la izquierda y que corresponde al subdirectorio C:/max2work/tutorial indicado a la derecha. Marque OK.



Al dar OK, vuelva a la ventana *Programmer*; en ella ha quedado configurado el chip e indicado el archivo a transferir al dispositivo.

Haga clic en *Add* -agregar- a fin de que quede cargado este archivo en el *buffer* de salida del programador.

En la ventana del centro aparece la información del chip y del archivo a utilizar.



Ya conectado el cable de programación al puerto paralelo de la PC y encendido el entrenador, puede usted comprobar que la conexión está bien, si presiona el botón *Detect JTAG Chain Info*. Si todo está en orden, va a recibir el mensaje *JTAG Caín information confirmed by hardware check*.

Como en este caso sólo hay un archivo a utilizar, puede dar OK.

Ahora, está en condiciones de realizar la programación. Para ello, haga clic en el botón *Program*.

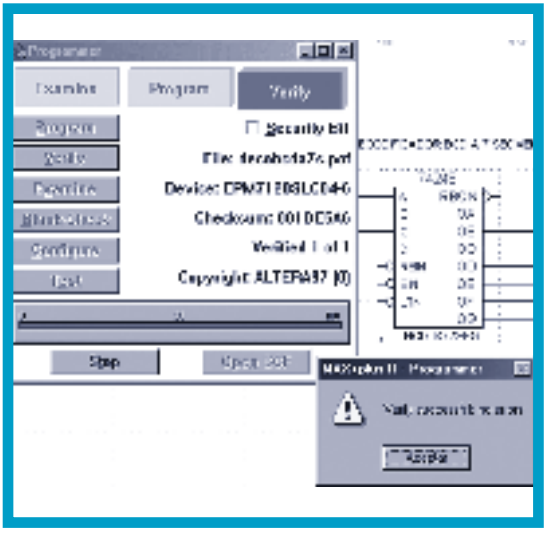
Una barra horizontal indica el progreso de la programación; cuando ésta finalice, llega al 100 %.

Esta operación tiene tres etapas:

- *Examine* -examinar-,
- *Program* -programar- y
- *Verify* -verificar-.

Si todo sale bien, va a ver un mensaje de *Programming Complete* -programación completa-.

Puede hacer clic en el botón Verify para comprobar qué ha quedado programado en el chip. Si todo está correcto, va a leer un mensaje de *Verify successful*: *No Error* -verificación exitosa: no hay errores-.



4. El equipo en el aula

A continuación le presentamos algunos ejemplos de aplicación para este entrenador en lógica programada. Se trata de proyectos que se implementan en forma separada y de uno en el que se combinan los anteriores.

Recuerde que:

- Cada vez que se programa el dispositivo EPM7128, se borra la configuración anteriormente grabada. Por lo tanto, si usted quiere que se sintetice en el chip más de un proyecto simultáneamente, agregue cada uno de esos circuitos en un mismo archivo de trabajo.
- Es necesario tener cuidado en la designación correcta de los pines del chip, ya que éste puede dañarse en caso de que un pin designado como salida esté conectado a una fuente de señal externa entrante. Si el grupo de alumnos utiliza las placas propuestas, éstas ya tienen predefinidos los pines de entrada y salida que van hacia la placa principal donde se conectan con el chip EPM7128; por lo tanto, sólo es necesario respetar las conexiones establecidas[29].

[29] Por ejemplo, las 4 llaves del DIP-Switch SW1 de la placa número 3, ya están conectadas en forma definitiva a los pines 22, 09, 21 y 08 del conector de dicha placa y, a través de él, a los pines 68, 69, 64 y 65 del EPM7128. Asimismo, para dar otro ejemplo, el display de 7 segmentos D1 que se encuentra en la placa de experimentación número 1, recibe señales desde el integrado U1, cuyas 8 entradas provienen -a través del conector asociado a dicha placa- de los pines 16, 17, 18, 10, 11, 15, 12 y 20 del EPM7128.

Vamos a compartir con usted ocho proyectos.

Entrenador en lógica programada

- **Proyecto 1:** Implementación de un decodificador BCD a 7 segmentos
- **Proyecto 2:** Divisor de frecuencias de 4.000.000 Hz a 4 Hz
- **Proyecto 3:** Contador de 2 dígitos BCD con entrada de reloj externa al equipo
- **Proyecto 4:** Contador de 2 dígitos BCD con divisor de frecuencias incorporado
- **Proyecto 5:** Sumador binario sin signo de 4 bits
- **Proyecto 6:** Circuito monoestable disparado por flanco ascendente
- **Proyecto 7:** Combinación de los proyectos 4, 5 y 6 en un mismo diseño
- **Proyecto 8:** Uso del entrenador como herramienta de análisis de circuitos

Cada uno de estos proyectos permite familiarizarse con diferentes formas de síntesis posibles, empleando el ambiente gráfico del cual dispone el software *MAX+PLUS II*.

Los alumnos están desarrollando un circuito que les permite visualizar, en un display de 7 segmentos, la evolución de un contador de décadas de 2 dígitos.

Para tal fin, éste va a estar conectado tanto a una señal de reloj externa como a una interna en la que el reloj es generado a una frecuencia lo suficientemente baja como para poder seguir visualmente los cambios en el número de conteo.

Además, el contador dispone de 3 entradas adicionales a la de reloj, para permitir:

- Modificar el sentido del conteo -ascendente o descendente-.
- Inhibir o habilitar el conteo.
- Borrar -*resetear* o poner a "0"- a los contadores.

En el CD que acompaña a este módulo de capacitación, usted puede encontrar los archivos empleados para realizar la edición, simulación y programación del chip EPM7128:

- Los archivos con extensión "gdf" corresponden a los circuitos esquemáticos del editor gráfico.
- Los archivos con extensión "rpt" son los de reporte de la compilación efectuada.
- Los archivos con extensión "snf" son los que generó el compilador y que se emplean en la etapa de simulación.
- Los archivos con extensión "scf" corresponden al editor del simulador que ya contiene una simulación tipo.
- Los archivos con extensión "pof" son los que hay que cargar en el chip empleando el programa *Programmer*.

Veamos cómo los alumnos fueron dándole solución a su problema

Proyecto 1: Implementación de un decodificador BCD a 7 segmentos

En la sección anterior, la descripción de cada uno de los programas que integran al ambiente de desarrollo *MAX+PLUS II* (editor de texto, compilador, simulador y programador) se corresponde con este diseño, desde la entradas de la información en el editor gráfico hasta la programación del chip.

Para este caso, seleccionamos ciertos pines del EPM7128, a fin de que coincidan con los de las placas de experimentación propuestas.

Aquí, las entradas del decodificador provienen del *DIP-Switch* número 1 de la placa número. Como salidas al display de 7 segmentos se utilizan los pines que excitan al circuito integrado U1 (ULN2803) que corresponde al control del display D1 de la placa número 1.

Para poder generar este proyecto los alumnos necesitan, entonces, conectar las dos placas de experimentación.

De esta manera, pueden cambiar el dígito BCD que se visualiza en D1, modificando las

4 llaves de SW1, y generando números desde 0000 (que equivale al dígito "0") hasta el 1001 (que equivale al dígito "9").

Este proyecto es denominado "decobcda7s.gdf".

Para complementar esta tarea, sus alumnos pueden generar un nuevo proyecto que permita entrar el dato en BCD a través del DIP-Switch SW2 y visualizar la información de salida del decodificador en el display D2.

Proyecto 2: Divisor de frecuencias de 4.000.000 Hz a 4 Hz

Este proyecto tiene como finalidad servir como generador de reloj del proyecto "Contador de 2 dígitos BCD".

La función del circuito que implementan los estudiantes permite obtener una frecuencia de reloj muy estable de baja frecuencia (4 ciclos por segundo); la frecuencia inicial es la que proviene de un oscilador de muy alta precisión cuya frecuencia es de 4 megahertz.

La entrada de excitación al circuito proviene de una señal de reloj de 4.000.000 Hz que ingresa por el pin de GCLK (pin 83) y divide dicha señal en frecuencia por 1.000.000; por esto, a la salida, tenemos una onda cuadrada de 4 Hz (con período de 250 milisegundos).

> No deje de verificar que el jumper J7 de la placa principal esté colocado correctamente, para que la salida del oscilador a cristal vaya al pin 83 del EPM7128.

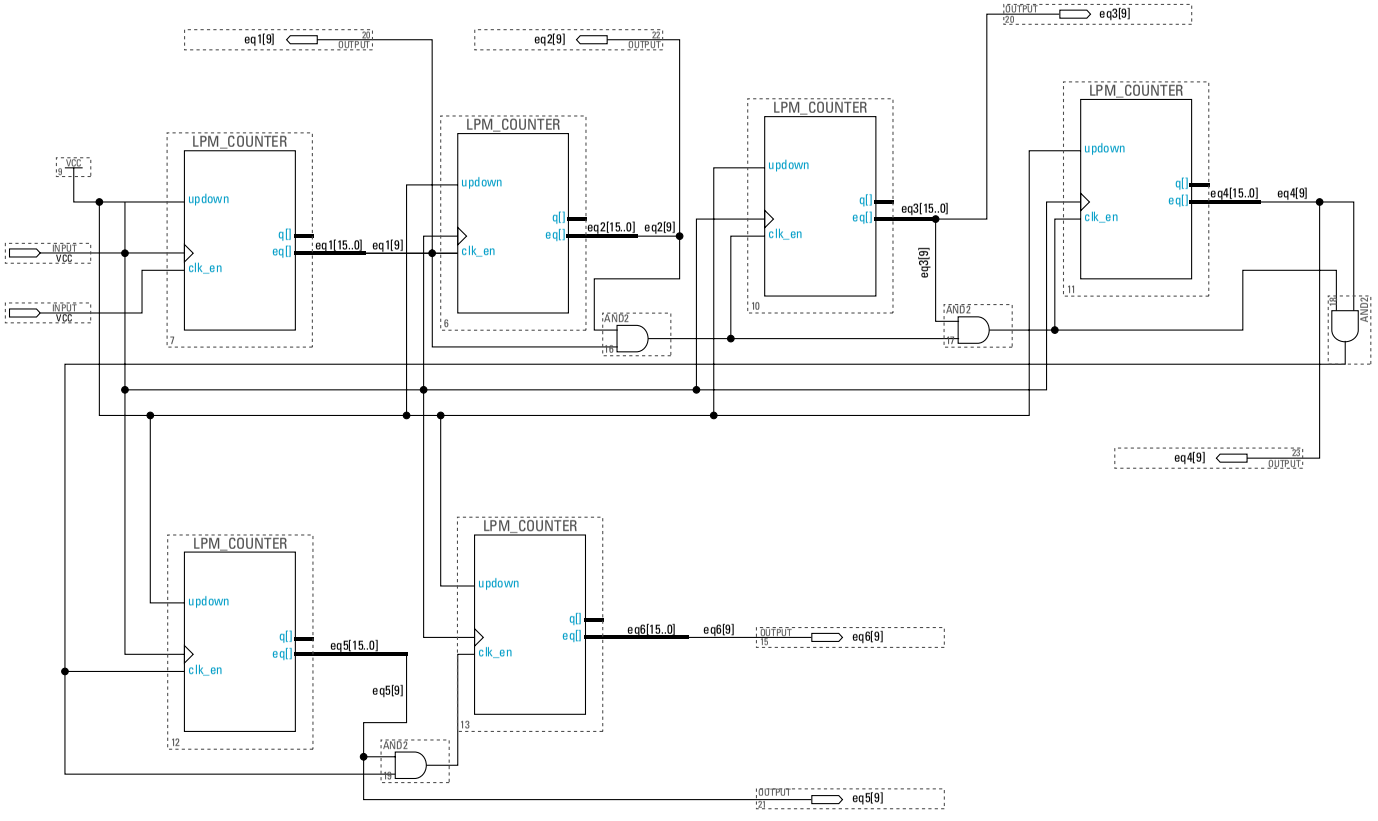
Esta señal de salida ya dividida es sacada por uno de los conectores BNC de la placa de experimentación número 3, que puede ser visualizada con un osciloscopio.

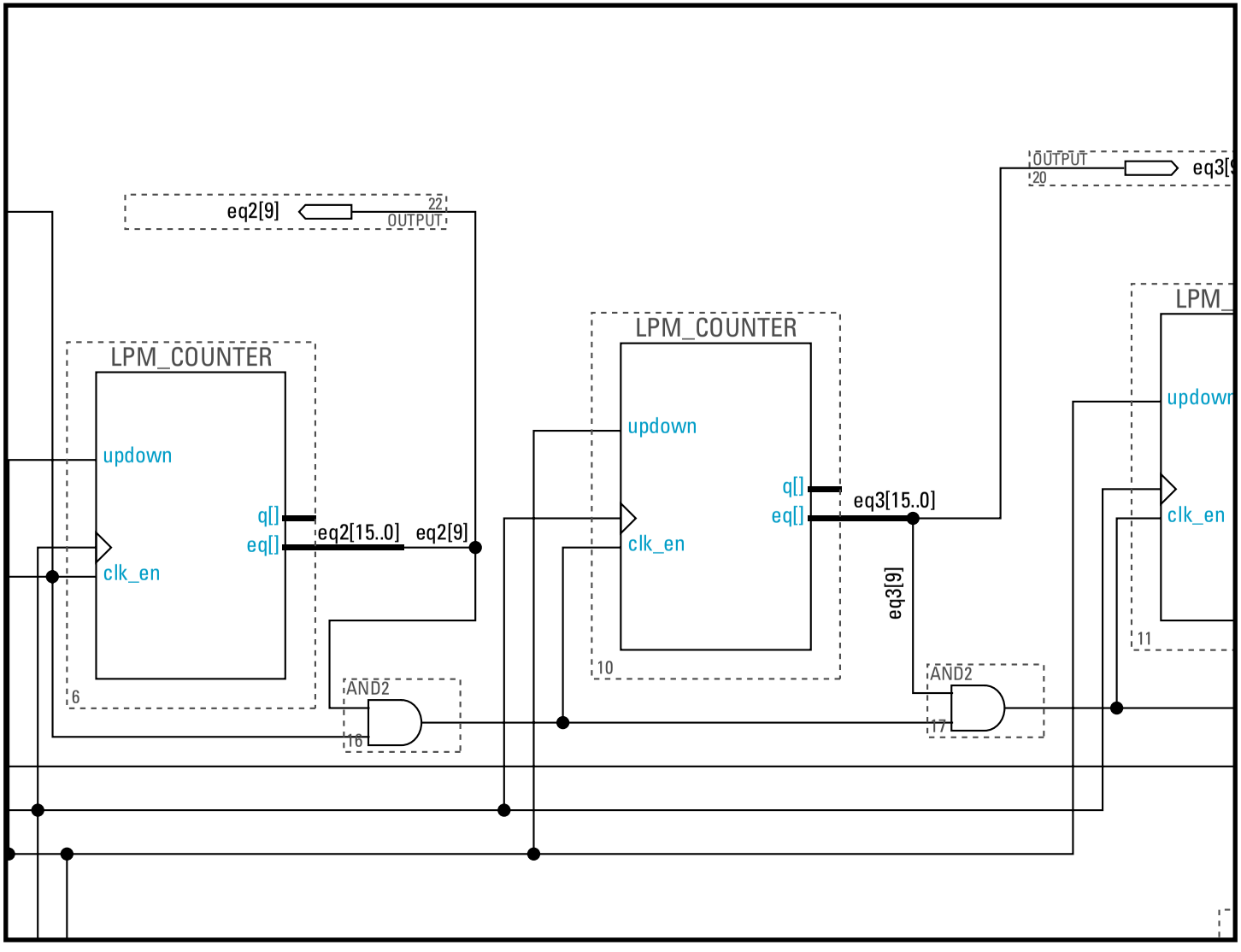
En este proyecto, los alumnos implementan una cascada sincrónica de contadores de década (BCD); cada uno de ellos divide 10 veces la frecuencia de la señal de reloj.

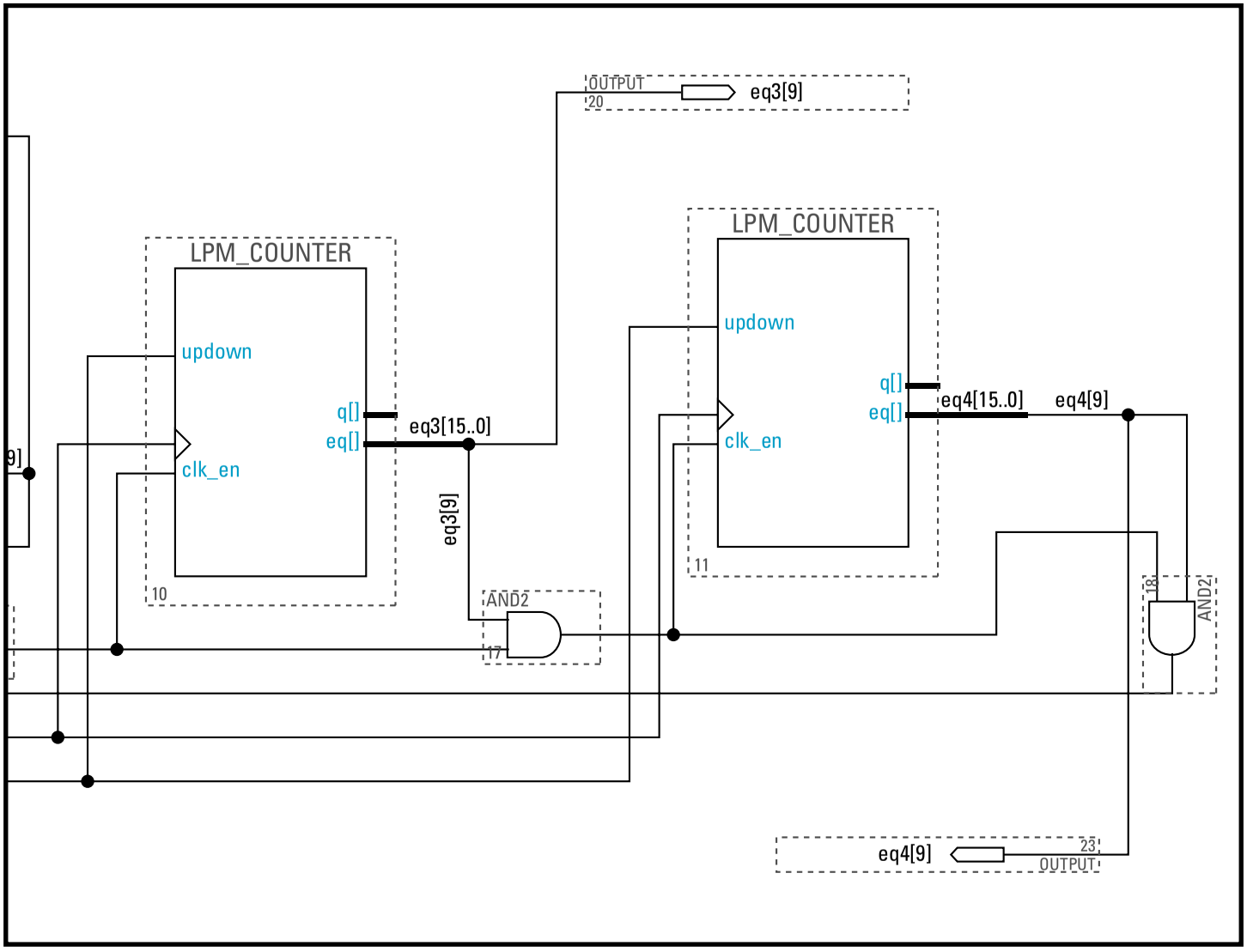
Para lograr un total de 1.000.000 de veces, deben utilizar 6 de ellos.

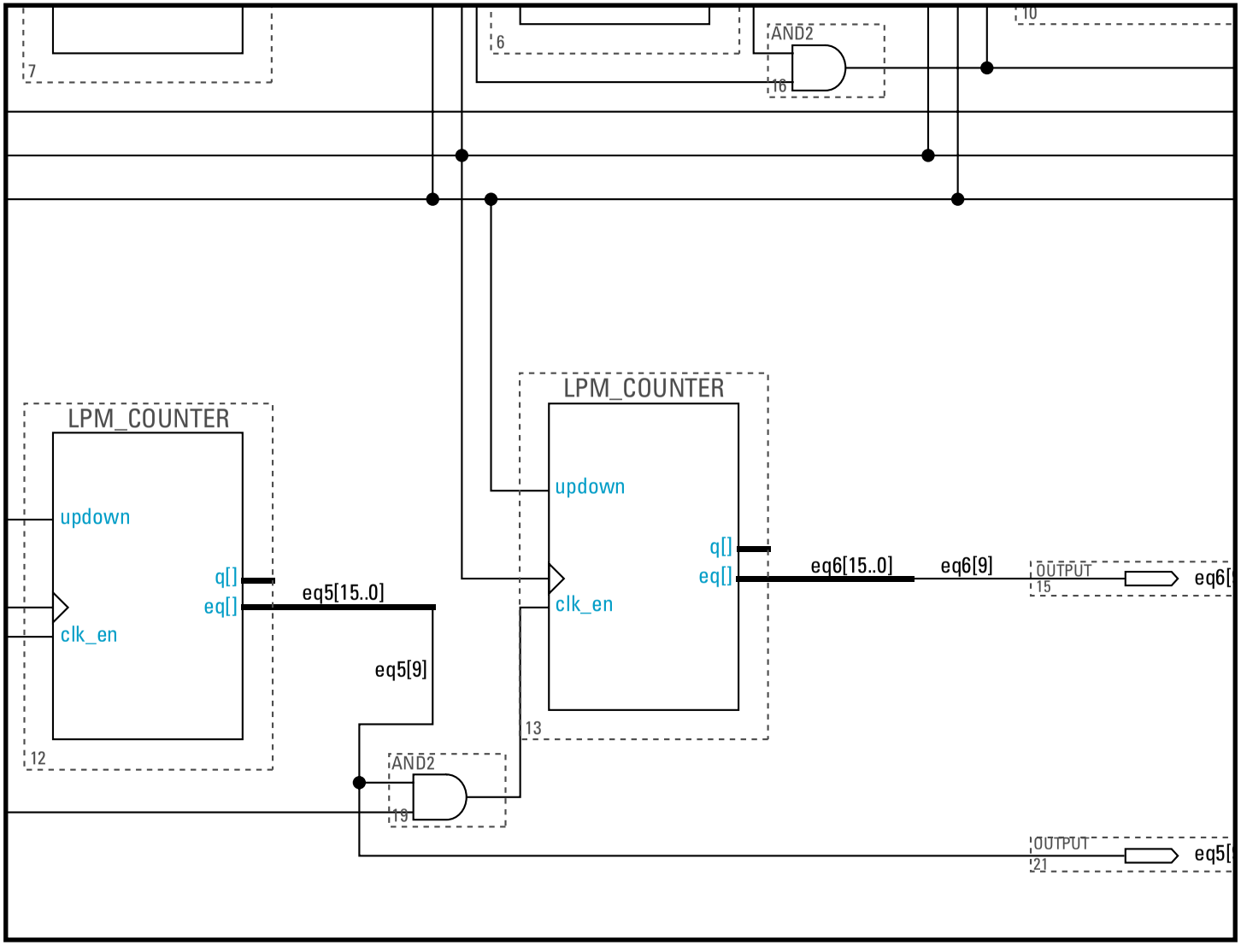
Llaman a este proyecto "divisorfrec.gdf"

En las figuras de la proxima página mostramos el circuito total y ampliaciones de sus partes.


eq1[9]
OUTPUT
eq2[9]
OUTPUT
OUTPUT
eq3[9]
LPM_COUNTER
updown
clk_en
q[]
eq[]
eq1[15..0]
eq1[9]
eq2[15..0]
eq2[9]
eq3[15..0]
eq3[9]
eq4[15..0]
eq4[9]
VCC
INPUT
AND2
eq4[9]
OUTPUT
eq5[15..0]
eq5[9]
eq6[15..0]
eq6[9]
OUTPUT
eq6[9]
OUTPUT
eq5[9]


eq2[9]
22
OUTPUT
LPM_COUNTER
updown
clk_en
q[]
eq[]
eq2[15..0]
eq2[9]
6
AND2
16
LPM_COUNTER
updown
clk_en
q[]
eq[]
eq3[15..0]
10
eq3[9]
AND2
17
OUTPUT
20
LPM_
updown
clk_en
11


OUTPUT
20
eq3[9]
LPM_COUNTER
updown
q[]
eq[]
clk_en
eq3[15..0]
10
eq3[9]
AND2
17
LPM_COUNTER
updown
q[]
eq[]
clk_en
eq4[15..0]
eq4[9]
11
18
AND2
9]
eq4[9]
23
OUTPUT

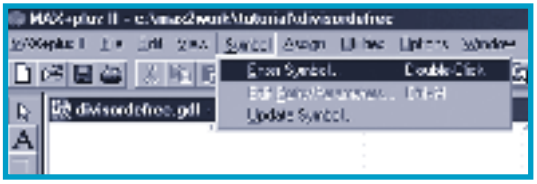

7
6
AND2
16
10
LPM_COUNTER
updown
q[]
eq[]
clk_en
eq5[15..0]
eq5[9]
12
LPM_COUNTER
updown
q[]
eq[]
clk_en
eq6[15..0]
eq6[9]
13
OUTPUT
15
eq6[
AND2
19
OUTPUT
21
eq5[

Este divisor está compuesto por 6 contadores tipo BCD, en conexión sincrónica; en él, todas las entradas de reloj están conectadas a una única entrada denominada "reloj".

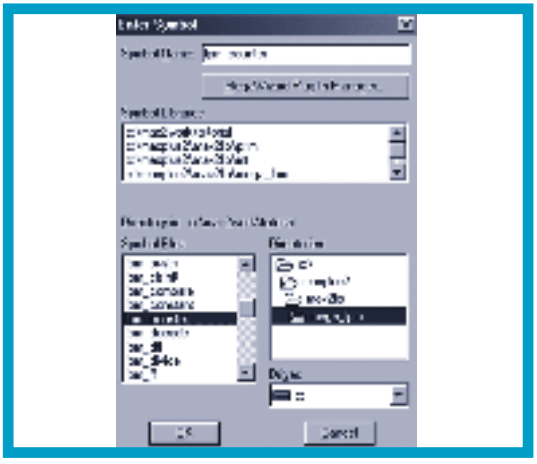
Se emplean, además, compuertas *and* de 2 entradas que se obtienen de la librería *prim*, accediendo a la opción *and2*.

Para dividir por 1.000.000, los estudiantes usan cada contador dividiendo por 10 cada vez. Para ello, necesitan el total de 6 contadores que obtienen de la librería de componentes parametrizados "mega_lpm".

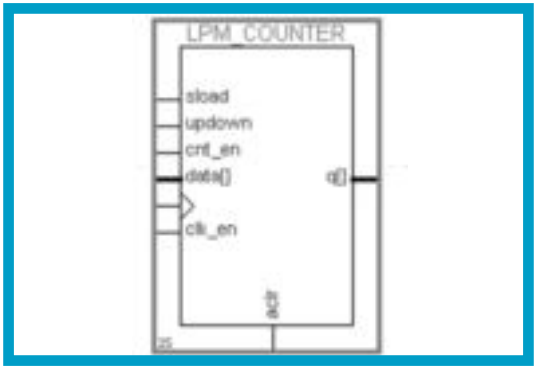
Volvamos al contador. Los alumnos entran a *Symbol* del menú principal y eligen la opción *Enter symbol*.

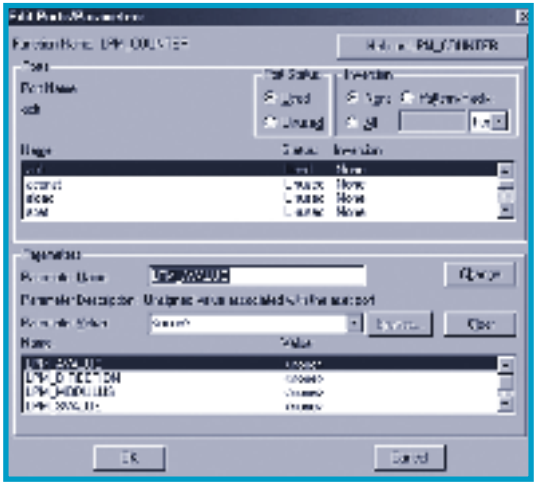


Aparece una ventana en la que seleccionan la opción *mega_lpm*. Al hacer doble clic sobre ella, se ve una lista de todos los componentes parametrizados. Eligen la opción *lpm_counter.*



Además del símbolo del contador, se ve una ventana que permite asignar las señales a utilizar en el contador generalizado.





Los estudiantes lo configuran para que sea un contador BCD con entrada de reloj y habilitación de reloj, y para que tenga una salida que indica que el conteo llega a "9".

Cada contador tiene salidas de conteo denominadas q[ ]. Su cantidad de bits depende de cómo se configura el contador. En nuestro caso, como el grupo de alumnos decide que sean del tipo BCD, cada uno de ellos va a tener 4 bits para poder contar desde "0000" ("0") hasta "1001" ("9").

Además de estas salidas, existen otras llamadas salidas eq[ ]. Cada una de estas salidas se pone a "1" cuando el valor del conteo presente en las salidas de datos q[ ] coincide con el número que las designa. Por ejemplo: la salida eq[9], se pone a "1" cada vez que las salidas de datos llegan a "1001". Cuando, luego del siguiente ciclo de reloj, pasan a "0000", la salida vuelve a "0".

Para el primer contador, entonces, cada 10 ciclos del reloj de entrada, en eq1[9] se cuenta con una onda cuadrada de período 10 veces menor que la señal de entrada "reloj" o -lo que es lo mismo- dividido en frecuencia por 10.

La idea, entonces, es la siguiente:

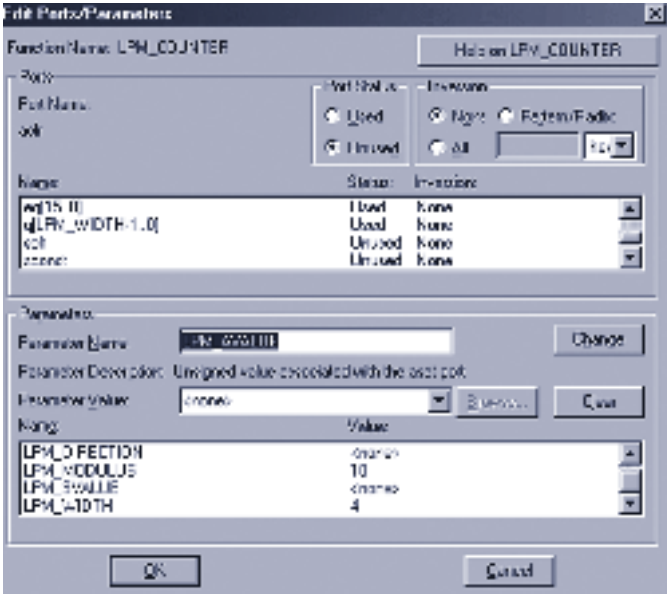
- El primer contador cuenta de "0000" a "1001" y repite dicho ciclo, indefinidamente.
- Como el segundo contador tiene el reloj conectado a la misma entrada que el primero, éste debe cambiar su cuenta cada vez que el primer contador pasa de "9" a "0".
- Para ello, emplea la entrada de habilitación de reloj"clk_en". Cuando hay un "1" en "clk_en," se permite que el contador cuente ciclos de reloj.
- Uniendo la salida eq1[9] a la entrada de "clk_en" del contador 2, logra que cada vez que el contador 1 pasa de "9" a "0", el contador 2 cuente un evento.
- Si se saca una señal de eq2[9], se obtiene una onda cuadrada 10 veces más lenta que la salida eq1[9] y, como ésta ya era 10 veces más lenta que "reloj", hasta aquí se ha dividido por 100.
- Para dividir ahora por 1000, los alumnos conectan el tercer contador. Se aseguran que éste cuente un evento sólo cuando se verifica que los contadores 1 y 2 llegan simultáneamente a "9", es decir a "99". Por eso es recomendable usar una compuerta and de 2 entradas.
- Cada entrada and está conectada a una salida eq[9], a fin de garantizar que el contador 3 está sólo habilitado a contar cuando los anteriores llegan -ambos- al número "9".

En la siguiente figura vemos una simulación de las salidas eq[9] de los tres primeros contadores BCD.

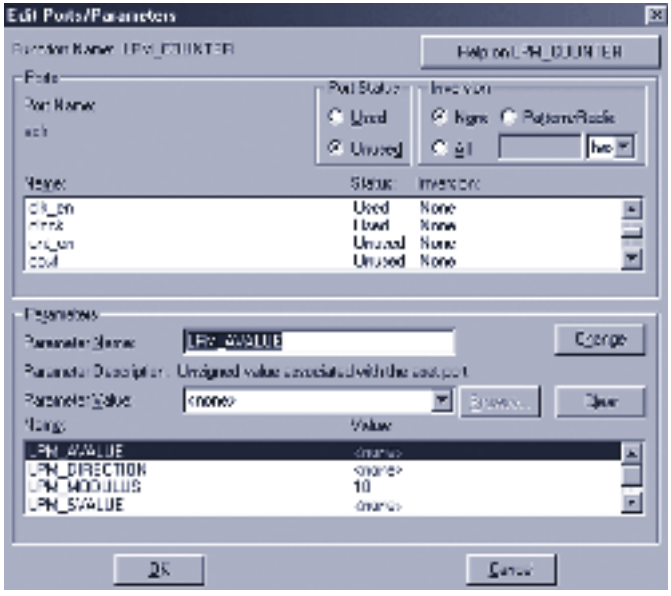
Los alumnos repiten este proceso para el cuarto contador, que debe quedar habilitado para contar sólo cuando los tres primeros llegan, simultáneamente, a "9" -es decir, para el conteo "999"-. Para esto, usan otra *and*; ésta permite que se haga la *and* total entre las señales eq1[9], eq2[9] y eq3[9].

Repiten para el resto de los contadores, empleando otras compuertas *and* adicionales.

Para editar los parámetros de los contadores, hacen clic en cada uno de ellos y presionan el botón derecho del mouse. Aparece la ventana de edición *Edit Ports/Parameters*.



En las siguientes figuras pueden verse los parámetros que los alumnos han considerado:



Las entradas a utilizar son la señal de reloj clock, la señal de habilitación de conteo clock_en y una entrada denominada *updown* que permite definir si se va a contar en forma ascendente o descendente (Esta señal puede anularse en la ventana de Edit Ports/Parameters...; pero, el profesor les propone mantenerla, porque va a ser usada en el próximo proyecto).

Conectan *updown* a Vcc ("1" lógico) para que cuente siempre en forma progresiva o ascendente.

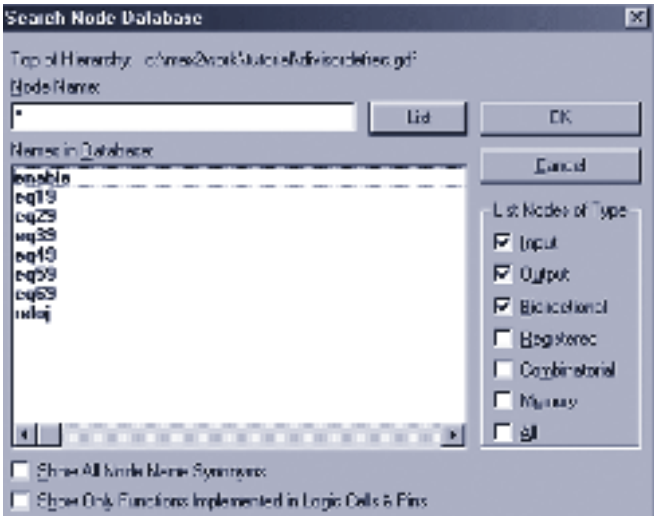
Los alumnos no utilizan las salidas q[ ] de los contadores; como sólo les interesa implementar un divisor de frecuencias, emplean las salidas para la detección de un conteo específico que es el número "9".

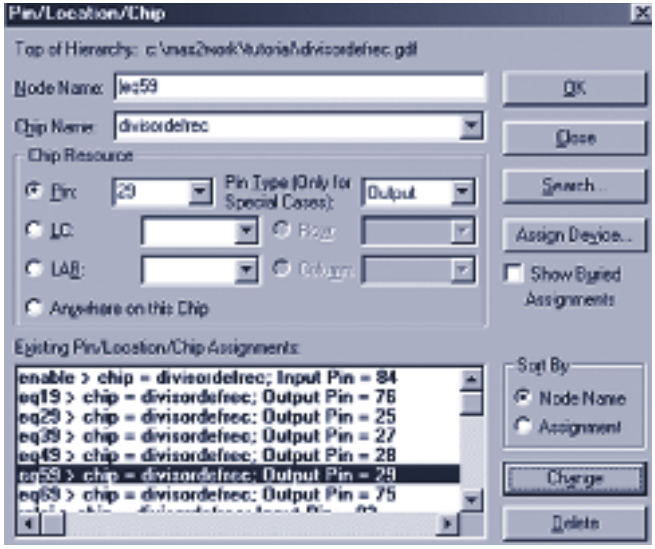
La salida final del divisor es la eq6[9] (último contador).

A continuación, realizan las uniones entre componentes.

> Explicamos en detalle este proceso, en el ejemplo de diseño de un decodificador BCD a 7 segmentos.

Una vez que el circuito está dibujado, realizan la asignación de dispositivo (EPM7128SCL84) dentro de la familia MAX7000S y de los pines:





A continuación, los alumnos compilan. Como no han tenido errores, inician la simulación para comprobar que, funcionalmente, su diseño responde a lo esperado.

Las asignaciones que se muestran en las figuras son tales que:

- El reloj está conectado al pin 83 del chip; éste va al oscilador de cristal vía el Jumper J7.
- El *enable* está conectado al interruptor de la placa de experimentación número 1 denominada "ENABLE".
- La salida eq6[9] -que es la que se ha dividido por 1.000.000- está conectada al pin 75 del chip para que termine en el conector BNC 1 de la placa de experimentación número 3.
- La salida eq1[9] que divide por 10, está conectada al BNC 2 de la placa.
- Las otras salidas, eq2[9] , eq3[9] , eq4[9] y eq5[9], se envían al conector número 2 a los pines 2, 3, 4 y 5, respectivamente.

En las siguientes figuras se presentan extractos del archivo de reporte que muestran cómo ha quedado configurado el chip y la cantidad de lógica empleada.

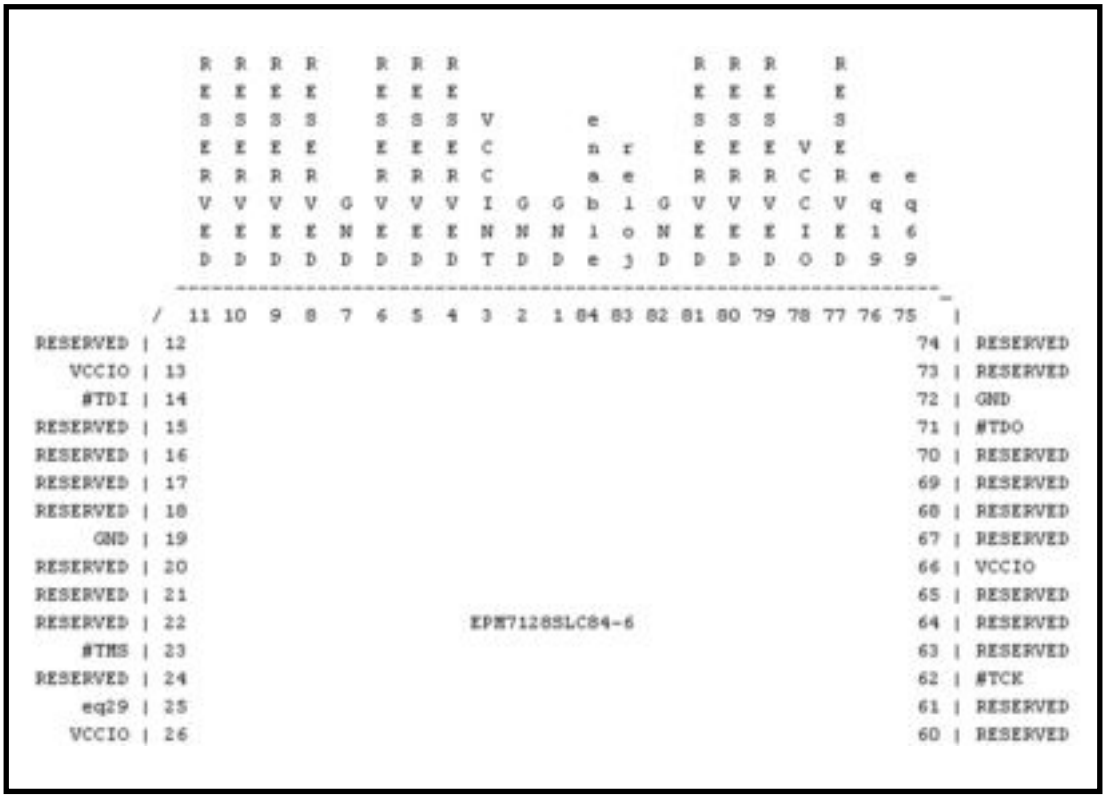
** INPUTS **

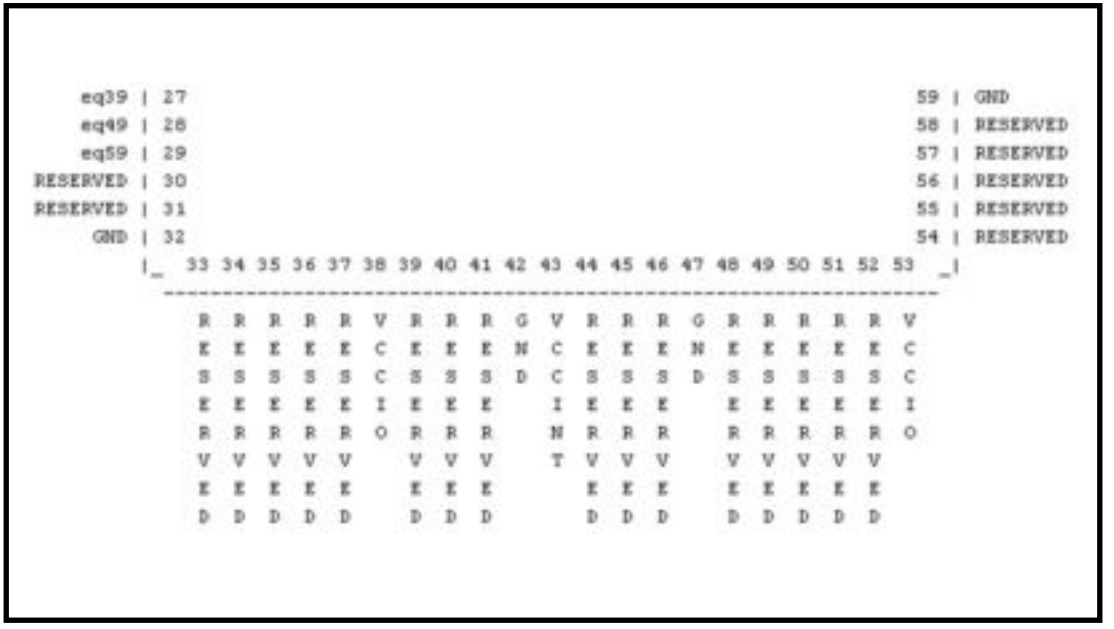
Pin	LC	LAB	Primitive	Code	Shareable Expanders Total	Shareable Expanders Shared	Shareable Expanders n/a	Fan-In INP	Fan-In FBK	Fan-Out OUT	Fan-Out FBK	Name
84	-	-	INPUT		0	0	0	0	0	0	4	enable
83	-	-	INPUT	G	0	0	0	0	0	0	0	reloj

** OUTPUTS **

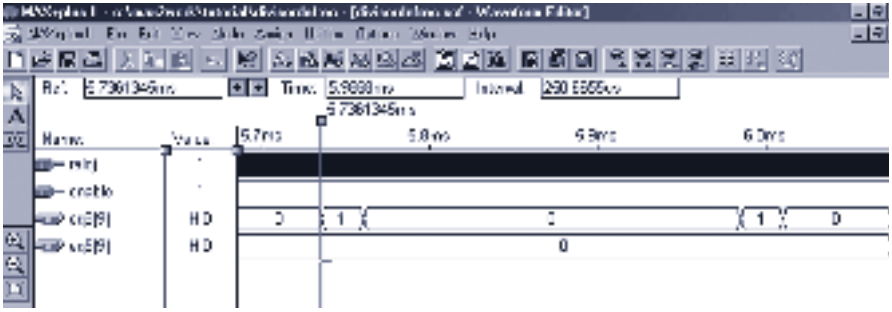
Pin	LC	LAB	Primitive	Code	Shareable Expanders Total	Shareable Expanders Shared	Shareable Expanders n/a	Fan-In INP	Fan-In FBK	Fan-Out OUT	Fan-Out FBK	Name
76	120	H	OUTPUT	t	0	0	0	0	4	0	0	eq19
25	45	C	OUTPUT	t	0	0	0	0	4	0	0	eq29
27	43	C	OUTPUT	t	0	0	0	0	4	0	0	eq39
28	40	C	OUTPUT	t	0	0	0	0	4	0	0	eq49
29	38	C	OUTPUT	t	0	0	0	0	4	0	0	eq59
75	118	H	OUTPUT	t	0	0	0	0	4	0	0	eq69

Total dedicated input pins used:	2/4	(50%)
Total I/O pins used:	10/64	(15%)
Total logic cells used:	48/128	(37%)
Total shareable expanders used:	0/128	(0%)
Total Turbo logic cells used:	48/128	(37%)
Total shareable expanders not available (n/a):	0/128	(0%)
Average fan-in:	9.20	
Total fan-in:	442	
Total input pins required:	2	
Total fast input logic cells required:	0	
Total output pins required:	6	
Total bidirectional pins required:	0	
Total reserved pins required	4	
Total logic cells required:	48	
Total flipflops required:	24	
Total product terms required:	108	
Total logic cells lending parallel expanders:	0	
Total shareable expanders in database:	0	
Synthesized logic cells:	0/ 128	(0%)


RESERVED | 12
VCCIO | 13
#TDI | 14
RESERVED | 15
RESERVED | 16
RESERVED | 17
RESERVED | 18
GND | 19
RESERVED | 20
RESERVED | 21
RESERVED | 22
#TMS | 23
RESERVED | 24
eq29 | 25
VCCIO | 26
74 | RESERVED
73 | RESERVED
72 | GND
71 | #TDO
70 | RESERVED
69 | RESERVED
68 | RESERVED
67 | RESERVED
66 | VCCIO
65 | RESERVED
64 | RESERVED
63 | RESERVED
62 | #TCK
61 | RESERVED
60 | RESERVED
EPM7128SLC84-6


eq39 | 27
eq49 | 28
eq59 | 29
RESERVED | 30
RESERVED | 31
GND | 32
59 | GND
58 | RESERVED
57 | RESERVED
56 | RESERVED
55 | RESERVED
54 | RESERVED

En las siguientes figuras podemos ver algunos gráficos de simulaciones hechas de este divisor. Se han simulado sólo las salidas correspondientes al tercer y quinto contador:



Por su parte, el contador de 2 dígitos puede ser replicado del divisor de frecuencias (segundo proyecto), ya que está formado, básicamente, por los dos primeros contadores BCD de dicho divisor. Cada una de las salidas de estos contadores va a la entrada de uno de los decodificadores BCD a 7 segmentos. Sus salidas van a las entradas de los drivers - manejadores de corriente- ULN2803 (U1 y U2) de los displays D1 y D2 ubicados en la placa de experimentación número 1.

Proyecto 3: Contador de 2 dígitos BCD con entrada de reloj externa al equipo

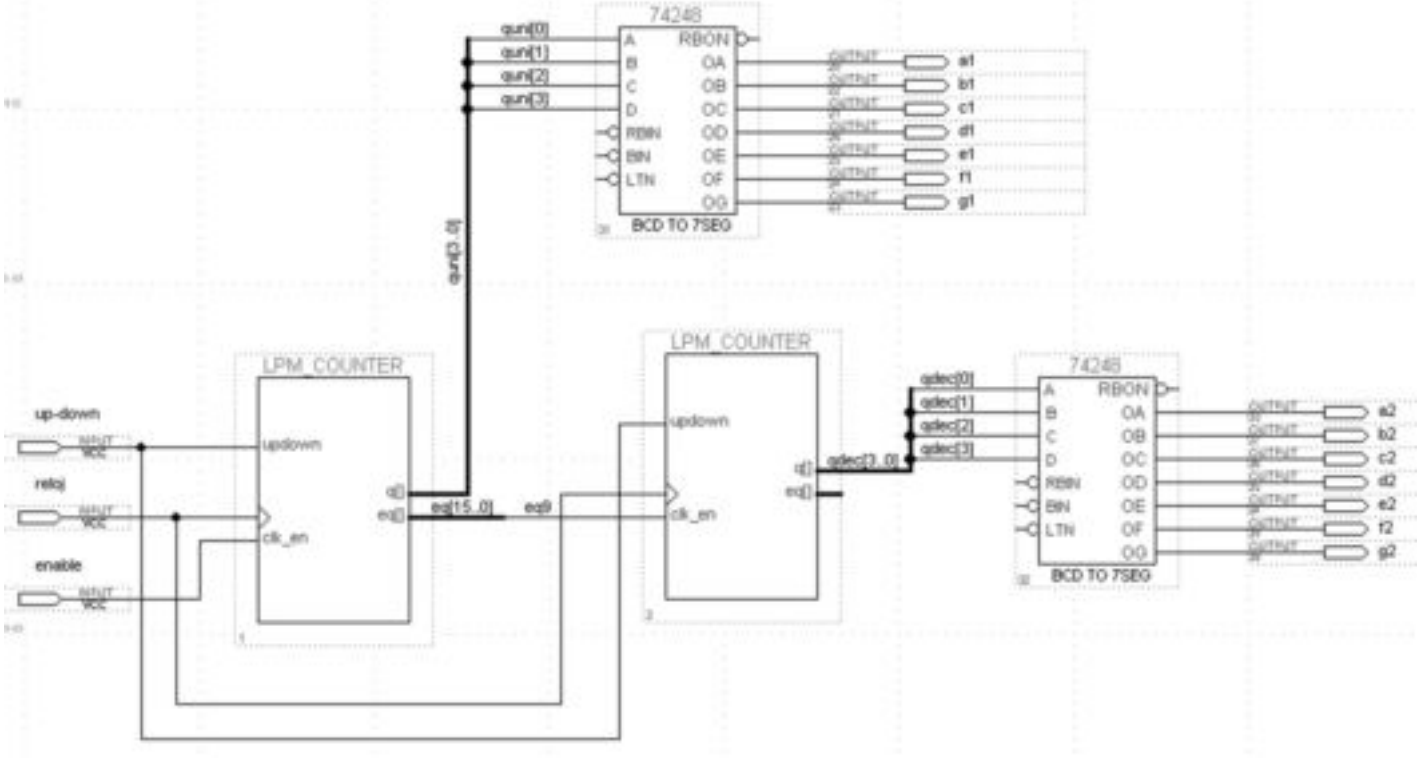
Este proyecto consta de dos contadores BCD y dos decodificadores BCD a 7 segmentos, a fin de poder implementar un contador de 2 dígitos BCD con visualización en 2 displays tipo LED de 7 segmentos.

Como los estudiantes ya han diseñado el decodificador BCD a 7 segmentos en el primer proyecto, lo único que necesitan es duplicarlo, haciendo *Copy* y *Paste*.

Sin embargo, existe una diferencia de conexión: ahora, el reloj es externo al entrenador, por lo que dicha señal debe ser inyectada desde el conector número 2 de la placa principal (pin 17 del DB25), donde se coloca el jumper J7 de tal forma que la conexión quede fijada con el pin 83 del EPM7128 (GCLK1).

El grupo de alumnos llama "contador2-dig1.gdf" a este proyecto.

El esquema del circuito es:


7424S
A
B
C
D
RBIN
BIN
LTN
RBON
OA
OB
OC
OD
OE
OF
OG
BCD TO 7SEG
qun[0]
qun[1]
qun[2]
qun[3]
qun[3..0]
a1
b1
c1
d1
e1
f1
g1
LPM_COUNTER
updown
clk_en
q[]
eq[]
up-down
reloj
enable
eq[15..0]
eq9
qdec[0]
qdec[1]
qdec[2]
qdec[3]
qdec[3..0]
a2
b2
c2
d2
e2
f2
g2

Para desarrollar este proyecto, apelan a los archivos "divisorfrec.gdf" y "decobcda7s.gdf", y hacen *Copy* y *Paste*, a fin de copiar el decodificador y parte del circuito del divisor de frecuencias.

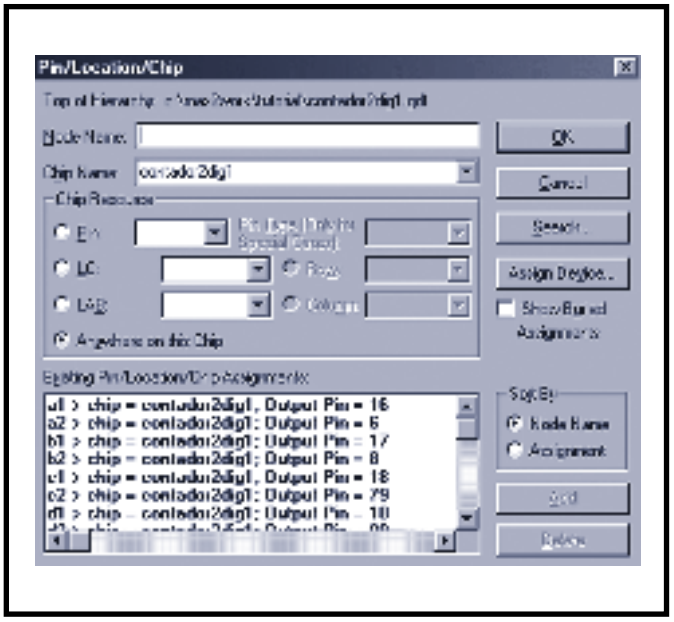
Como necesitan dos decodificadores, una vez que copian "contador2dig1.gdf" a la hoja de trabajo, vuelven a hacer un *Copy* y *Paste* para obtener el segundo.

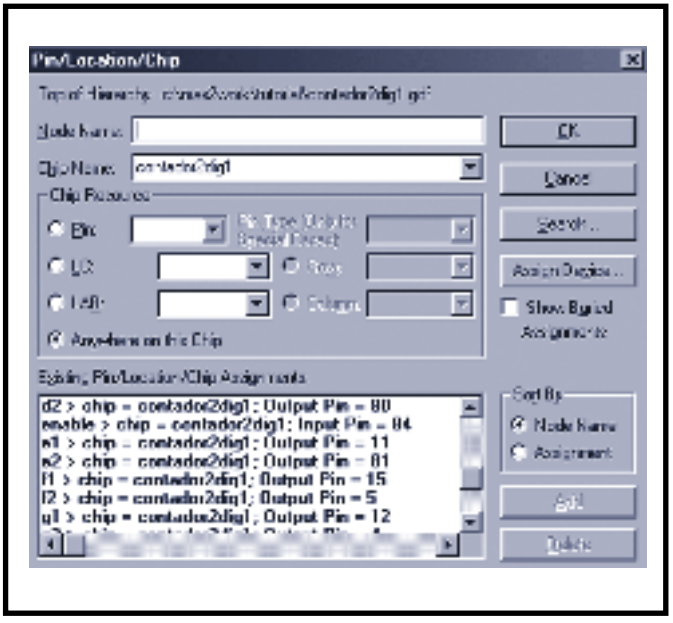
Por otro lado, sacan el circuito de los dos contadores BCD del circuito del divisor de frecuencias, borrando lo que no corresponde, es decir, los 4 últimos flip-flops.

Las entradas a este proyecto son: *enable*, reloj y *up-down* -habilitación de reloj, reloj y control de conteo progresivo-regresivo, respectivamente):

- Asocian *enable* al interruptor de enable de la placa de experimentación número 1.
- Hacen lo mismo con la entrada *up-down* y con el interruptor *up-down*.
- Toman el reloj del conector número 2, pin 17, de la placa principal, donde ajustan convenientemente J7.
- Envian las salidas de los decodificadores (14 en total) a los displays de 7 segmentos de la placa de experimentación número 1.

Paso seguido, como antes, los alumnos asignan al proyecto el dispositivo EPM7128SLC84 y, luego, los pines a emplear:





Una vez hecha las asignaciones, el grupo de alumnos compila y simula.

Las siguientes figuras muestran extractos del archivo de reporte de la compilación realizada:

```
** INPUTS **

                                          Shareable
                                          Expanders      Fan-In     Fan-Out
Pin     LC   LAB  Primitive     Code    Total Shared n/a INP  FBK  OUT  FBK  Name
84      -    -       INPUT              0      0     0   0    0    0    4   enable
83      -    -       INPUT   G          0      0     0   0    0    0    0   reloj
2       -    -       INPUT              0      0     0   0    0    0   12   up-down
```

```
** OUTPUTS **

                                          Shareable
                                          Expanders      Fan-In     Fan-Out
Pin     LC   LAB  Primitive     Code    Total Shared n/a INP  FBK  OUT  FBK  Name
16      27   B       OUTPUT      t          0      0     0   0    4    0    0   a1
6       13   A       OUTPUT      t          0      0     0   0    4    0    0   a2
17      25   B       OUTPUT      t          0      0     0   0    4    0    0   b1
8       11   A       OUTPUT      t          0      0     0   0    4    0    0   b2
18      24   B       OUTPUT      t          0      0     0   0    4    0    0   c1
79      125  H       OUTPUT      t          0      0     0   0    4    0    0   c2
10      6    A       OUTPUT      t          0      0     0   0    4    0    0   d1
80      126  H       OUTPUT      t          0      0     0   0    4    0    0   d2
11      5    A       OUTPUT      t          0      0     0   0    3    0    0   e1
81      128  H       OUTPUT      t          0      0     0   0    3    0    0   e2
15      29   B       OUTPUT      t          0      0     0   0    4    0    0   f1
5       14   A       OUTPUT      t          0      0     0   0    4    0    0   f2
12      3    A       OUTPUT      t          0      0     0   0    4    0    0   g1
4       16   A       OUTPUT      t          0      0     0   0    4    0    0   g2
```

```
Total dedicated input pins used:                3/4      ( 75%)
Total I/O pins used:                           18/64     ( 28%)
Total logic cells used:                        28/128    ( 21%)
Total shareable expanders used:                 4/128    (  3%)
Total Turbo logic cells used:                  28/128    ( 21%)
Total shareable expanders not available (n/a):  0/128    (  0%)
Average fan-in:                                 5.14
Total fan-in:                                  144

Total input pins required:                      3
Total fast input logic cells required:          0
Total output pins required:                    14
Total bidirectional pins required:              0
Total reserved pins required                    4
Total logic cells required:                    28
Total flipflops required:                       8
Total product terms required:                  92
Total logic cells lending parallel expanders:   0
Total shareable expanders in database:          4

Synthesized logic cells:                        0/ 128   (  0%)
```

```
            R                                               R  R  R
            E                    u                          E  E  E
            S                 V  p     e                    S  S  S
            E                 C  -     n  r              V  E  E  E
            R                 C  d     a  e              C  R  R  R
            V     G           I  o  G  b  l  G           C  V  V  V
      e  d  E  b  N  a  f  g  N  w  N  l  o  N  e  d  c  I  E  E  E
      1  1  D  2  D  2  2  2  T  n  D  e  j  D  2  2  2  O  D  D  D
    ---------------------------------------------------------------_
   / 11 10  9  8  7  6  5  4  3  2  1 84 83 82 81 80 79 78 77 76 75   |
      g1 | 12                                                   74 | RESERVED
   VCCIO | 13                                                   73 | RESERVED
    #TDI | 14                                                   72 | GND
      f1 | 15                                                   71 | #TDO
      a1 | 16                                                   70 | RESERVED
      b1 | 17                                                   69 | RESERVED
      c1 | 18                                                   68 | RESERVED
     GND | 19                                                   67 | RESERVED
RESERVED | 20                                                   66 | VCCIO
RESERVED | 21                                                   65 | RESERVED
RESERVED | 22                 EPM7128SLC84-6                    64 | RESERVED
    #TMS | 23                                                   63 | RESERVED
RESERVED | 24                                                   62 | #TCK
RESERVED | 25                                                   61 | RESERVED
   VCCIO | 26                                                   60 | RESERVED
```

```
RESERVED | 27                                                   59 | GND
RESERVED | 28                                                   58 | RESERVED
RESERVED | 29                                                   57 | RESERVED
RESERVED | 30                                                   56 | RESERVED
RESERVED | 31                                                   55 | RESERVED
     GND | 32                                                   54 | RESERVED
  |_ 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53  _|
    ------------------------------------------------------------------
      R  R  R  R  R  V  R  R  R  G  V  R  R  R  G  R  R  R  R  R  V
      E  E  E  E  E  C  E  E  E  N  C  E  E  E  N  E  E  E  E  E  C
      S  S  S  S  S  C  S  S  S  D  C  S  S  S  D  S  S  S  S  S  C
      E  E  E  E  E  I  E  E  E     I  E  E  E     E  E  E  E  E  I
      R  R  R  R  R  O  R  R  R     N  R  R  R     R  R  R  R  R  O
      V  V  V  V  V     V  V  V     T  V  V  V     V  V  V  V  V
      E  E  E  E  E     E  E  E        E  E  E     E  E  E  E  E
      D  D  D  D  D     D  D  D        D  D  D     D  D  D  D  D
```

Para la simulación, abren un nuevo archivo denominado "contador2dig1.scf". En él cargan:

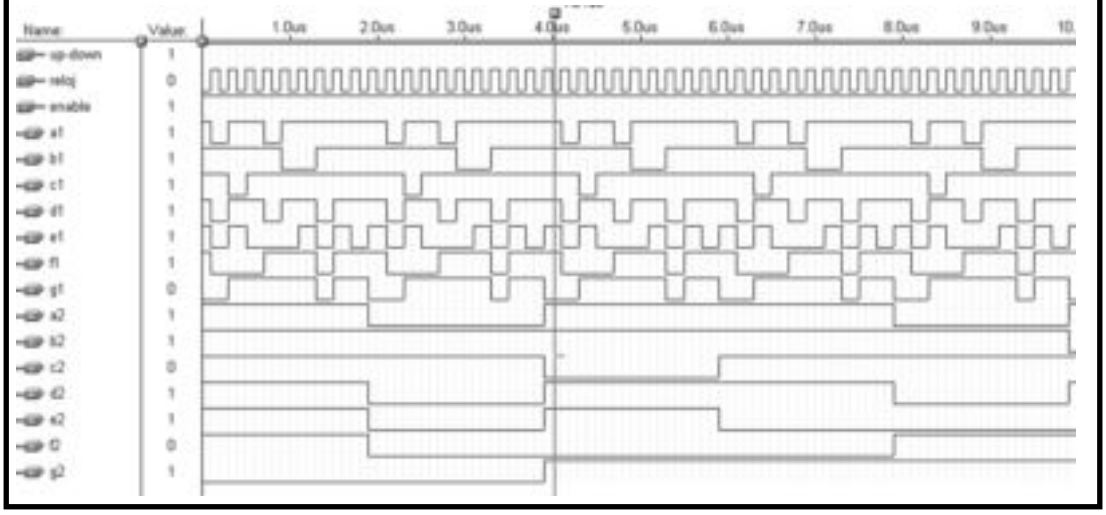
- las señales de entrada *enable*, reloj y *up-down;*
- las 14 salidas a1 hasta g1 y a2 hasta g2.

Vemos una simulación en la que se marca un instante de tiempos que corresponde al número de cuenta "20".

Esto se corresponde con los niveles lógicos que adquieren las salidas de 7 segmentos. Para el número "2" (decenas) los segmentos a2, b2, d2, e2 y g2, están en nivel alto y los demás en bajo.

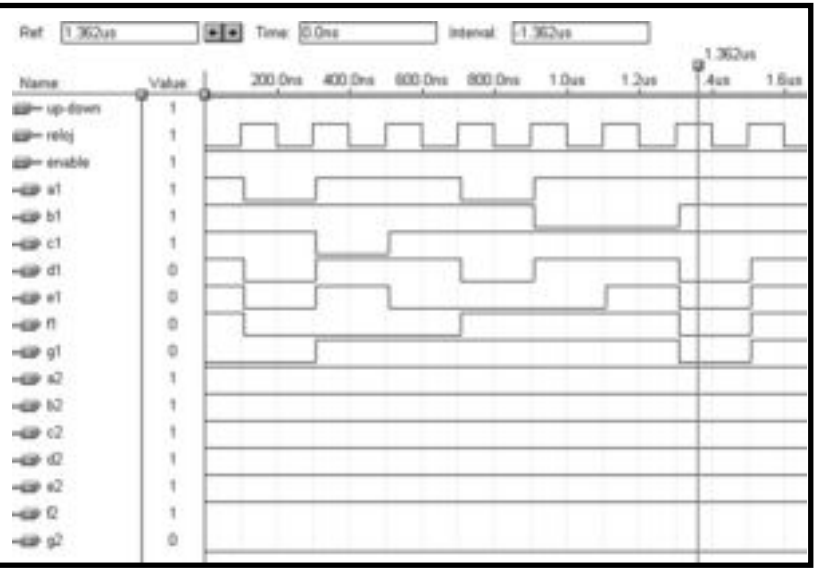
Para el "0" (unidades), todos los segmentos menos g1 están en alto.

Estos valores se pueden ver en la columna "Value".



La siguiente figura muestra otro momento del conteo. Aquí se marca el que corresponde al número "07":

Para la programación los alumnos emplean, como antes, el archivo "contador2dig1.pof", habiendo seleccionado como dispositivo el EPM7128, previamente.



Proyecto 4: Contador de 2 dígitos BCD con divisor de frecuencias incorporado

Este proyecto es similar al anterior, con la diferencia de que en el chip se implementa tanto el contador de 2 dígitos como el divisor de frecuencias.

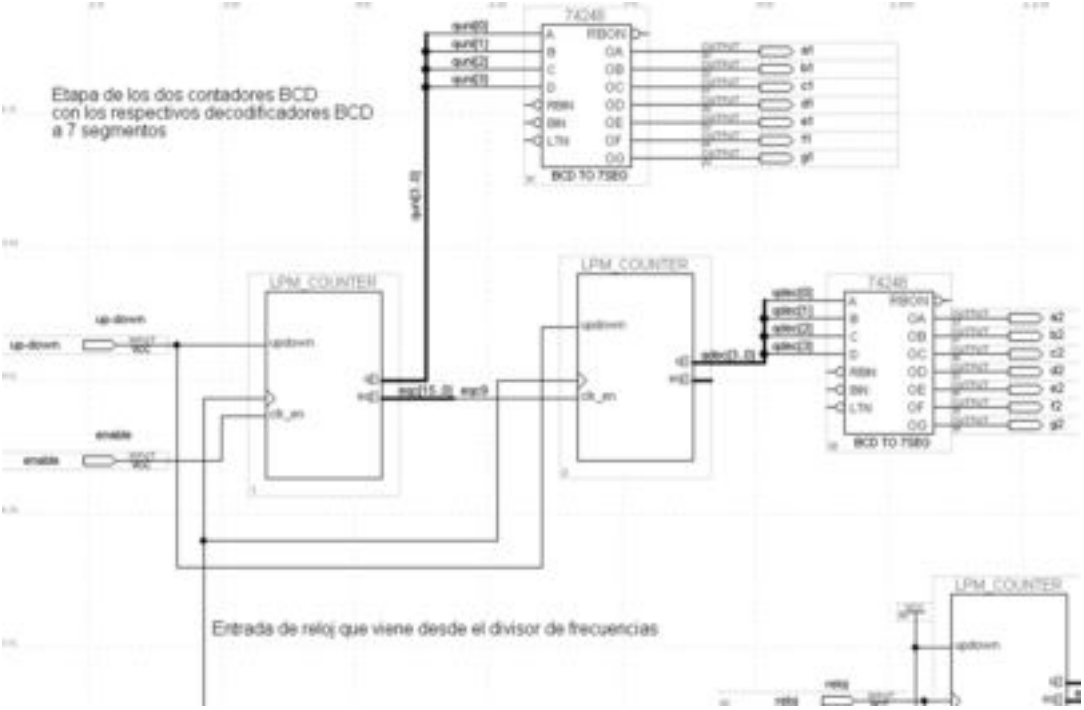
De esta manera, la señal de reloj que recibe el contador proviene del oscilador a cristal, dividiendo dicha frecuencia de 4 MHz por 1.000.000.

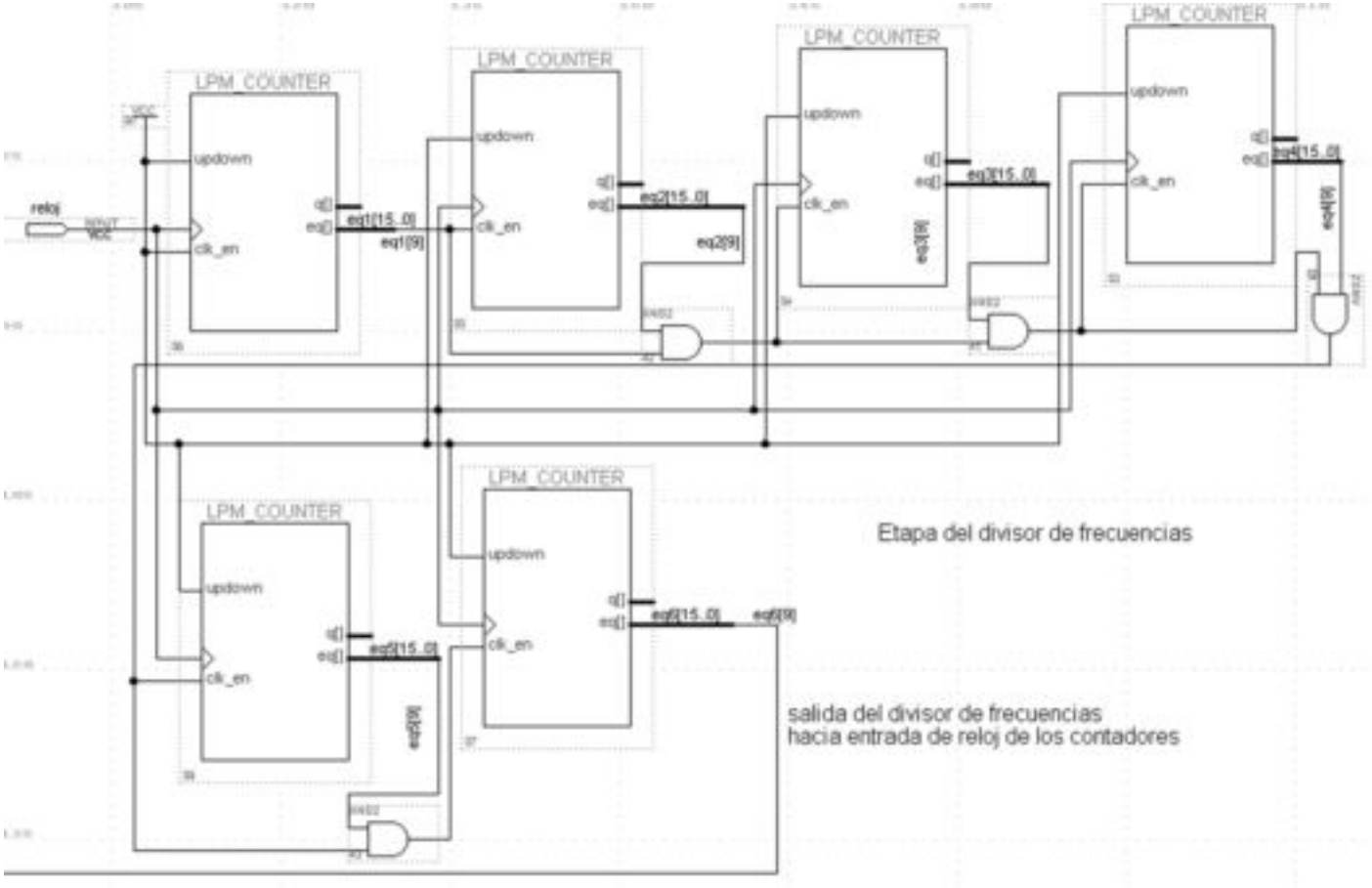
Con esto no es necesario disponer de una señal externa al entrenador para manejar el conteo.

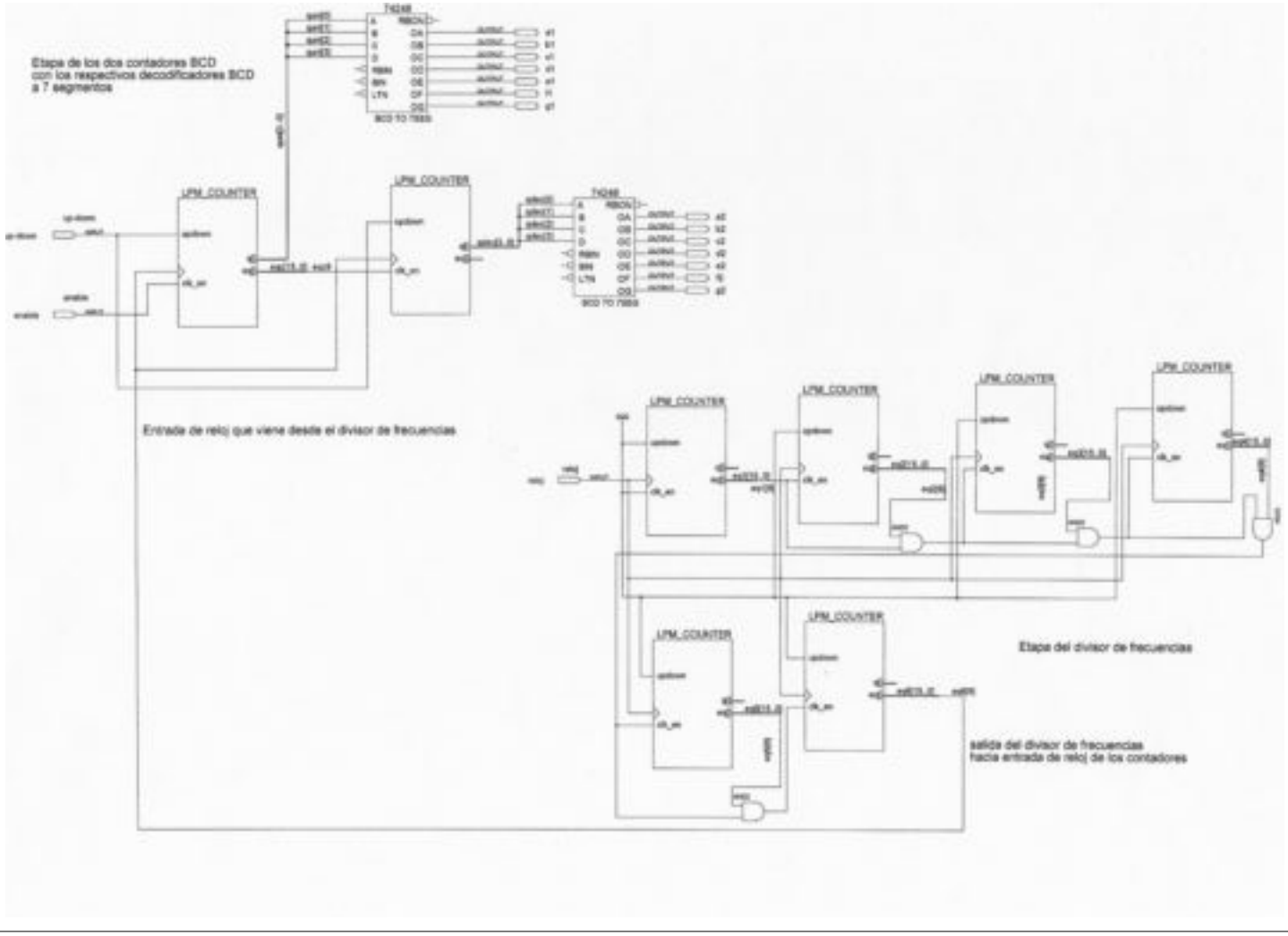
La diferencia en las asignaciones de pines respecto a los otros proyectos radica en que, ahora, tanto la entrada al contador como la salida del bloque divisor de frecuencias son señales internas al chip EPM7128.

Este proyecto se llama "contador2dig2.gdf".

En las siguientes figuras vemos el esquema propuesto por los alumnos y algunas ampliaciones.




LPM_COUNTER
updown
clk_en
reloj
eq1[15..0]
eq1[9]
eq2[15..0]
eq2[9]
eq3[15..0]
eq3[9]
eq4[15..0]
eq4[9]
eq5[15..0]
eq5[9]
Etapa del divisor de frecuencias
salida del divisor de frecuencias
hacia entrada de reloj de los contadores


Etapa de los dos contadores BCD
con los respectivos decodificadores BCD
a 7 segmentos
LPM_COUNTER
LPM_COUNTER
74248
BCD TO 7SEG
Entrada de reloj que viene desde el divisor de frecuencias
Etapa del divisor de frecuencias
salida del divisor de frecuencias
hacia entrada de reloj de los contadores

En este caso, los estudiantes han sacado las señales internas del divisor de frecuencias que les permitían ver cómo se iba dividiendo la señal de reloj de entrada. Sólo dejaron la salida del sexto flip-flop que, internamente, se conecta a la entrada de reloj de los dos contadores BCD; es decir, la entrada de reloj proveniente del oscilador de cristal a través del jumper J7 va directo a la entrada de reloj del divisor.

El grupo de los dos contadores BCD recibe esa señal dividida por 1.000.000.

Las asignaciones para las salidas de 7 segmentos siguen siendo las mismas que para el proyecto anterior, al igual que las entradas *enable*, reloj y *up-down*.

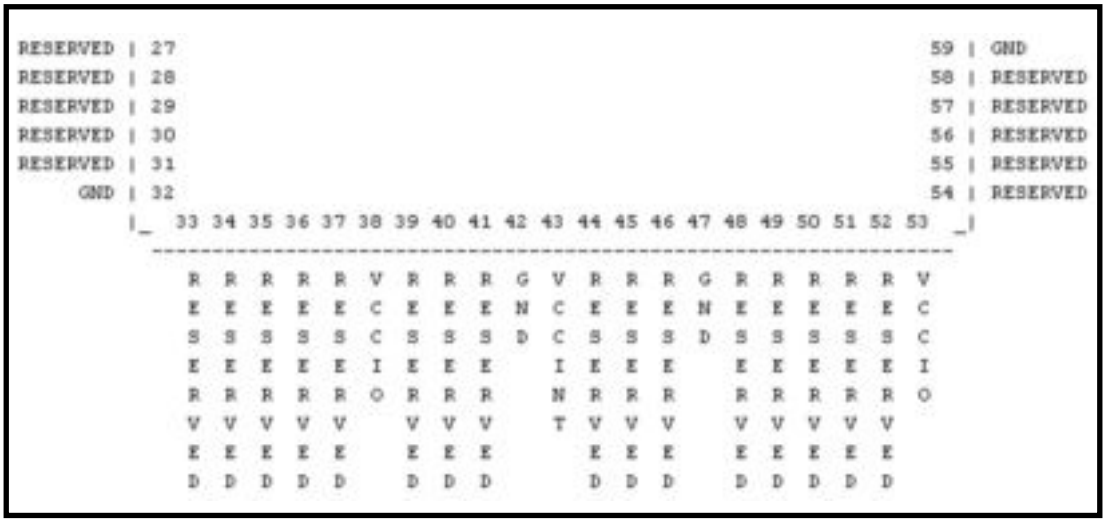
En las siguientes figuras vemos extractos del reporte, una vez hecha la compilación.

```
** INPUTS **

                                        Shareable
                                        Expanders      Fan-In    Fan-Out
 Pin     LC   LAB  Primitive    Code   Total Shared n/a INP  FBK  OUT  FBK  Name
  84      -    -       INPUT           0      0   0    0    0    0    4  enable
  83      -    -       INPUT  G        0      0   0    0    0    0    0  reloj
   2      -    -       INPUT           0      0   0    0    0    0   12  up-down
```

```
** OUTPUTS **

                                        Shareable
                                        Expanders      Fan-In    Fan-Out
 Pin     LC   LAB  Primitive    Code   Total Shared n/a INP  FBK  OUT  FBK  Name
  16     27    B       OUTPUT     t       0      0   0    0    4    0    0  a1
   6     13    A       OUTPUT     t       0      0   0    0    4    0    0  a2
  17     25    B       OUTPUT     t       0      0   0    0    4    0    0  b1
   8     11    A       OUTPUT     t       0      0   0    0    4    0    0  b2
  18     24    B       OUTPUT     t       0      0   0    0    4    0    0  c1
  79    125    H       OUTPUT     t       0      0   0    0    4    0    0  c2
  10      6    A       OUTPUT     t       0      0   0    0    4    0    0  d1
  80    126    H       OUTPUT     t       0      0   0    0    4    0    0  d2
  11      5    A       OUTPUT     t       0      0   0    0    3    0    0  e1
  81    128    H       OUTPUT     t       0      0   0    0    3    0    0  e2
  15     29    B       OUTPUT     t       0      0   0    0    4    0    0  f1
   5     14    A       OUTPUT     t       0      0   0    0    4    0    0  f2
  12      3    A       OUTPUT     t       0      0   0    0    4    0    0  g1
   4     16    A       OUTPUT     t       0      0   0    0    4    0    0  g2
```

```
Total dedicated input pins used:                 3/4      ( 75%)
Total I/O pins used:                            18/64     ( 28%)
Total logic cells used:                         70/128    ( 54%)
Total shareable expanders used:                  5/128    (  3%)
Total Turbo logic cells used:                   70/128    ( 54%)
Total shareable expanders not available (n/a):   6/128    (  4%)
Average fan-in:                                  8.28
Total fan-in:                                   580

Total input pins required:                       3
Total fast input logic cells required:           0
Total output pins required:                     14
Total bidirectional pins required:               0
Total reserved pins required                     4
Total logic cells required:                     70
Total flipflops required:                       32
Total product terms required:                  197
Total logic cells lending parallel expanders:    0
Total shareable expanders in database:           5

Synthesized logic cells:                         0/ 128   (  0%)
```

```
                     R                                                R  R  R
                     E                    u                           E  E  E
                     S                 V  p     e                     S  S  S
                     E                 C  -     n  r               V  E  E  E
                     R                 C  d     a  e               C  R  R  R
                     V     G           I  o  G  b  l  G            C  V  V  V
               e  d  E  b  N  a  f  g  N  w  N  l  o  N  e  d  c  I  E  E  E
               1  1  D  2  D  2  2  2  T  n  D  e  j  D  2  2  2  O  D  D  D
            -----------------------------------------------------------------_
          / 11 10  9  8  7  6  5  4  3  2  1 84 83 82 81 80 79 78 77 76 75   |
      g1 | 12                                                              74 | RESERVED
   VCCIO | 13                                                              73 | RESERVED
    #TDI | 14                                                              72 | GND
      f1 | 15                                                              71 | #TDO
      a1 | 16                                                              70 | RESERVED
      b1 | 17                                                              69 | RESERVED
      c1 | 18                                                              68 | RESERVED
     GND | 19                                                              67 | RESERVED
RESERVED | 20                                                              66 | VCCIO
RESERVED | 21                                                              65 | RESERVED
RESERVED | 22                       EPM7128SLC84-6                         64 | RESERVED
    #TMS | 23                                                              63 | RESERVED
RESERVED | 24                                                              62 | #TCK
RESERVED | 25                                                              61 | RESERVED
   VCCIO | 26                                                              60 | RESERVED
```



En este caso, las simulaciones son similares a los dos casos anteriores, por lo que no las repetimos.

Proyecto 5: Sumador binario sin signo de 4 bits

Los alumnos quieren saber cómo funciona un sumador de números binarios sin signo, de 4 bits.

Para ello, lo implementan en el entrenador, empleando 8 llaves que permiten entrar los datos de los números A y B (4 llaves para cada número) y 4 diodos emisores de luz para visualizar el resultado de la suma.

Adicionalmente, agregan otro led que indica que existe *overflow* -sobrecarga-; es decir, cuándo se han excedido en la capacidad de representar un número de 4 bits.

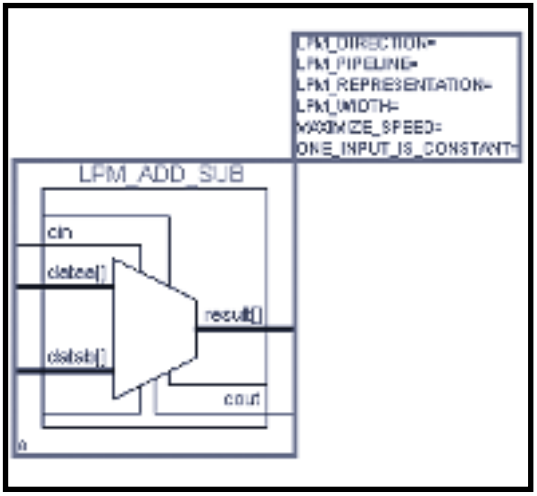
Diseñan, entonces, un sumador de números binarios sin signo de dos operandos *dataa* y *datab* con salidas *result* y *cout*.

- *dataa* y *datab* son dos números binarios sin signo, de 4 bits, que pueden ir desde "0000" (0 en notación decimal) hasta "1111" (15 en notación decimal). Estos números son ajustados desde la placa de experimentación número 3, empleando los DIP-Switches SW1 (para *dataa*) y SW2 (para *datab*).
- La salida de datos de 4 bits *result* se envía -a través del integrado U1 de la misma placa- a los 4 diodos led de color rojo (D1 a D4). D1 es el bit menos significativo (LSB) y D4 el bit más significativo (MSB).
- La salida *cout* se manda al diodo led D8 (color verde). Es una indicación de *overflow* -desborde- que señala una suma que supera los 4 bits (números mayores a 15 en decimal).

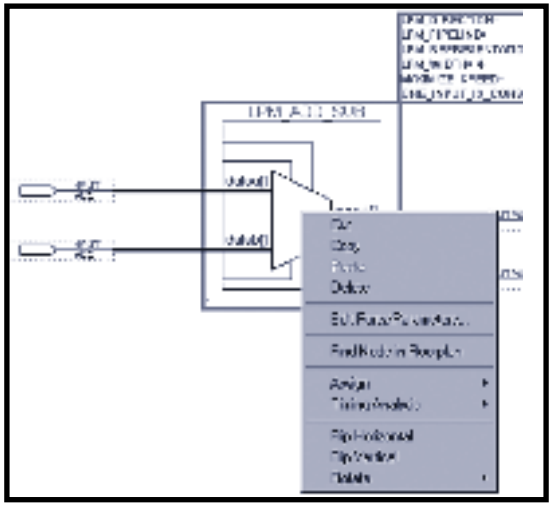
Para realizar el diseño, los estudiantes crean un proyecto denominado "sumador4bits.gdf":

- Entran, sucesivamente, a File → Project → *Name*.
- En File → *New*, crean un nuevo archivo, en modo gráfico, de extensión "gdf". Lo salvan como "sumador4bits.gdf".
- En *Symbol* → *Enter symbol* eligen la opción "mega_lpm" donde seleccionan "lpm_add_sub", que es un modelo parametrizado de un circuito sumador-restador.

Como se trata de un circuito genérico, tienen que especificarlo.



La primera vez[30] que los estudiantes acceden a un símbolo de una librería parametrizada, se abre automáticamente una ventana de diálogo, mostrando los parámetros que se pueden modificar.



Este componente *add_sub* es un sumador-restador de "n" bits, que puede sumar dos números binarios con o sin signo, según cómo se lo programe.

Por defecto, viene armado para realizar operaciones con signo; pero, para desarrollar su proyecto, los estudiantes lo cambian a "sin signo".

En la parte superior de la ventana de *Edit Ports/Parameters...* se puede ver una lista.

[30] En cualquier otro momento, marcando el símbolo parametrizado, aparece señalado con rojo. Haciendo clic con el botón derecho se despliega un menú en el que es posible elegir la opción Edit Ports/Parameters...

Con el título *Name* se encuentran todas las líneas de entrada y salida del componente.

En Status se indica si se usa *-Used-* o no *-Unused-* esa señal. Esto se puede modificar, haciendo clic en una señal dada y seleccionando en *Port status*: *Used* o *Unused*.

Los alumnos indican *Used* para las señales:

- *cout* -salida de indicación de *overflow*; desborde-,
- *dataa*[.....] -dato de entrada del operando "a"-,
- *datab*[.....] -dato de entrada del operando "b"-,
- *result*[.....] -dato de salida-.

En la ventana inferior, cambian:

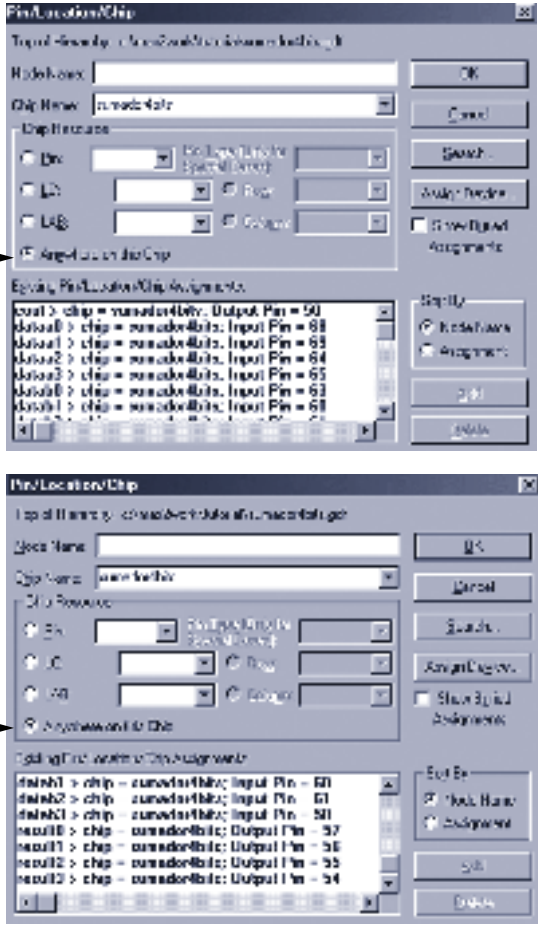
- "LPM_REPRESENTATION" a "UNSIGNED"
- LPM_WIDTH a "4".

Dejan los demás datos *een none* -ninguno-.

Como en los ejemplos anteriores, agregan los pines de entrada de datos de los operandos "a" y "b", que llaman *dataa*[3..0] y *datab*[3..0], respectivamente; también, la salidas de desborde *cout* y la de datos *result*[3..0].

En *Assign* → *Device*, seleccionan la opción del EPM7128SLC84-6

Luego, en *Pin/Location/*Chip asignan los pines:



Finalmente, realizan la compilación y, por último, la programación empleando el archivo "sumador4bits.pof".

De esta manera, queda configurado el chip para que la entrada "a" provenga del DIP-Switch SW1 y la entrada "b" del SW2 de la placa de experimentación número 3.

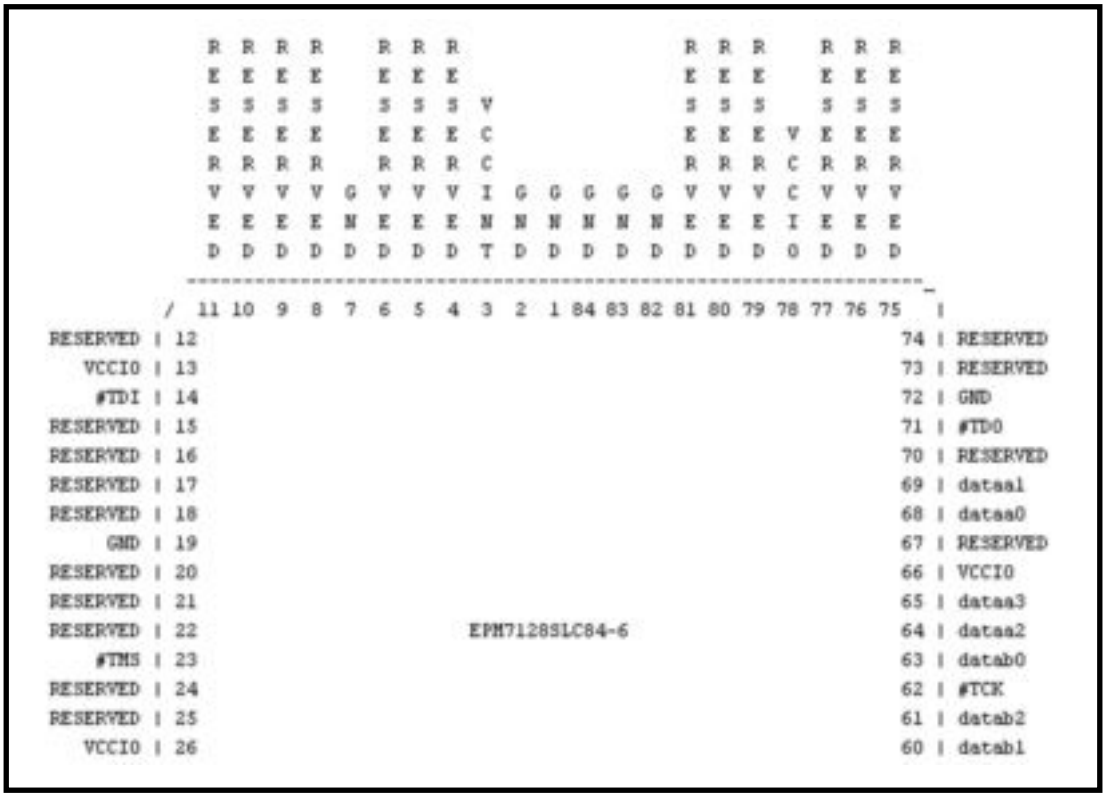
La salida de datos de 4 bits va a través del integrado U1 de la misma placa a los 4 *led* de color rojo "D1" a "D4", mientras que la salida de *overflow* denominada *cout* comanda al *led* verde "D8".

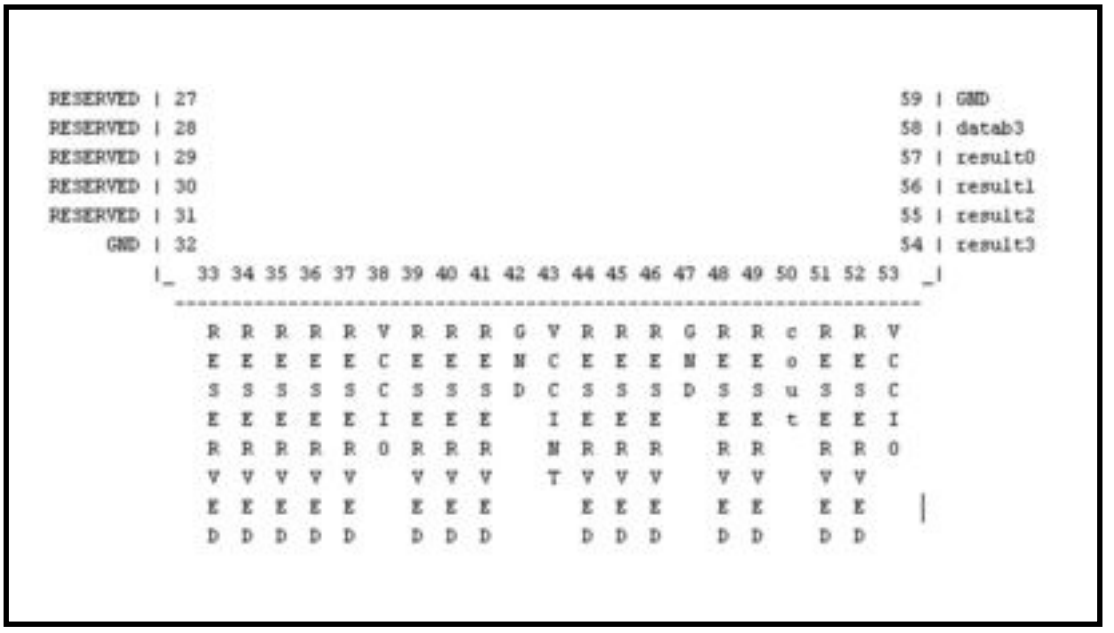
En las siguientes figuras vemos extractos del archivo de reporte "sumador4bits.rpt" desde los que es posible verificar que las asignaciones están correctas.

```
** INPUTS **

                                                Shareable
                                                Expanders      Fan-In    Fan-Out
 Pin     LC   LAB  Primitive     Code    Total Shared n/a  INP   FBK   OUT   FBK   Name
  68  (105)  (G)       INPUT                0      0   0    0     0     4     1   dataa0
  69  (107)  (G)       INPUT                0      0   0    0     0     3     1   dataa1
  64   (99)  (G)       INPUT                0      0   0    0     0     2     1   dataa2
  65  (101)  (G)       INPUT                0      0   0    0     0     2     0   dataa3
  63   (97)  (G)       INPUT                0      0   0    0     0     4     1   datab0
  60   (93)  (F)       INPUT                0      0   0    0     0     3     1   datab1
  61   (94)  (F)       INPUT                0      0   0    0     0     2     1   datab2
  58   (91)  (F)       INPUT                0      0   0    0     0     2     0   datab3
```

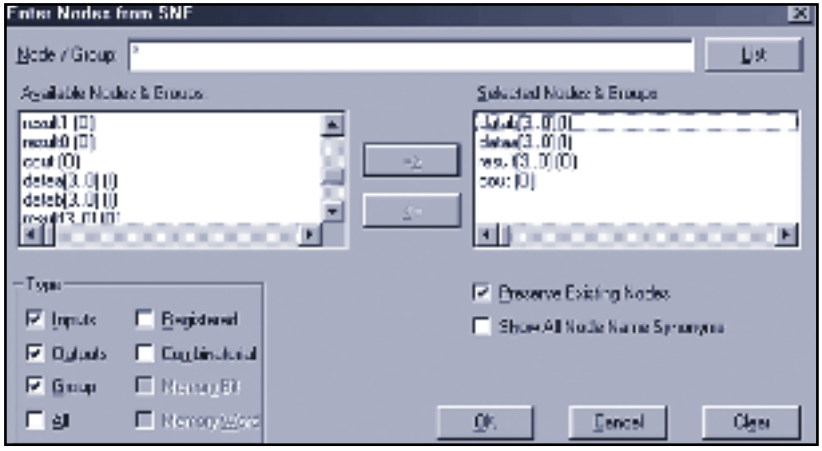
```
** OUTPUTS **

                                                Shareable
                                                Expanders      Fan-In    Fan-Out
 Pin     LC   LAB  Primitive     Code    Total Shared n/a  INP   FBK   OUT   FBK   Name
  50     75    E       OUTPUT      t        4      4   0    8     0     0     0   cout
  57     88    F       OUTPUT      t        0      0   0    2     0     0     0   result0
  56     86    F       OUTPUT      t        2      1   0    4     0     0     0   result1
  55     85    F       OUTPUT      t        3      2   0    6     0     0     0   result2
  54     83    F       OUTPUT      t        2      2   0    2     1     0     0   result3
```

```
Total dedicated input pins used:                0/4      (  0%)
Total I/O pins used:                           17/64     ( 26%)
Total logic cells used:                         6/128    (  4%)
Total shareable expanders used:                10/128    (  7%)
Total Turbo logic cells used:                   6/128    (  4%)
Total shareable expanders not available (n/a):  0/128    (  0%)
Average fan-in:                                 4.83
Total fan-in:                                   29

Total input pins required:                      8
Total fast input logic cells required:          0
Total output pins required:                     5
Total bidirectional pins required:              0
Total reserved pins required                    4
Total logic cells required:                     6
Total flipflops required:                       0
Total product terms required:                   28
Total logic cells lending parallel expanders:   0
Total shareable expanders in database:          6

Synthesized logic cells:                        0/ 128   (  0%)
```

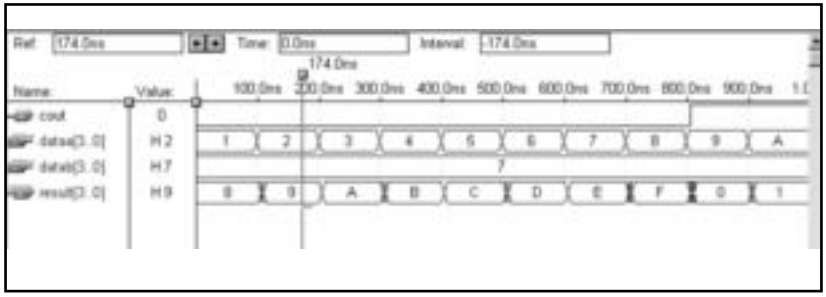

RESERVED
VCCIO
#TDI
RESERVED
RESERVED
RESERVED
RESERVED
GND
RESERVED
RESERVED
RESERVED
#TMS
RESERVED
RESERVED
VCCIO
EPM7128SLC84-6
RESERVED
RESERVED
#TDO
RESERVED
dataa1
dataa0
RESERVED
VCCIO
dataa3
dataa2
datab0
#TCK
datab2
datab1


RESERVED
RESERVED
RESERVED
RESERVED
RESERVED
GND
GND
datab3
result0
result1
result2
result3

Para realizar la simulación, crean un archivo "sumador4bits.scf" al que cargan las señales a emplear a través del menú *Node* en la opción *Enter nodes from* SNF (Recuerde los pasos que puntualizamos al describir el software).

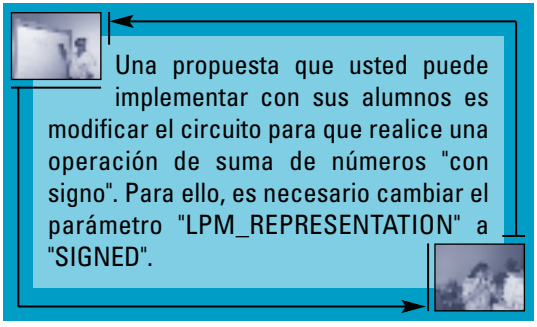


En la siguiente figura se muestra una corrida de la simulación del sumador



Aquí, agrupan los bits de datos de las entradas *dataa* y *datab*, y de la salida *result*.

Para este caso, dejan el dato de *datab* constante e igual a "7"; a la entrada *dataa* asignan datos variables, cambiando cada 100 ms, desde el número "1", con incrementos de a uno por vez.



Proyecto 6: Circuito monoestable disparado por flanco ascendente

Éste es un ejemplo de cómo se puede construir un circuito monoestable totalmente digital sin resistencia ni condensadores, como los que tienen que usarse en los chip de lógica estándar 74LS221, MC14528 o similares.

La ventaja principal es la de implementarlo dentro del chip sin requerir de componentes externos adicionales. Los pulsos así generados son de gran estabilidad en el tiempo, si se emplea un reloj a base de cristal de cuarzo; esto no ocurre al emplear componentes como resistencias y capacitores, cuyos coeficientes de temperatura son de varios cientos de partes por millón por grado centígrado.

El circuito está formado por dos fli-flop tipo D y compuertas.

Tiene una entrada de datos "X" y una salida de datos 2salida_m".

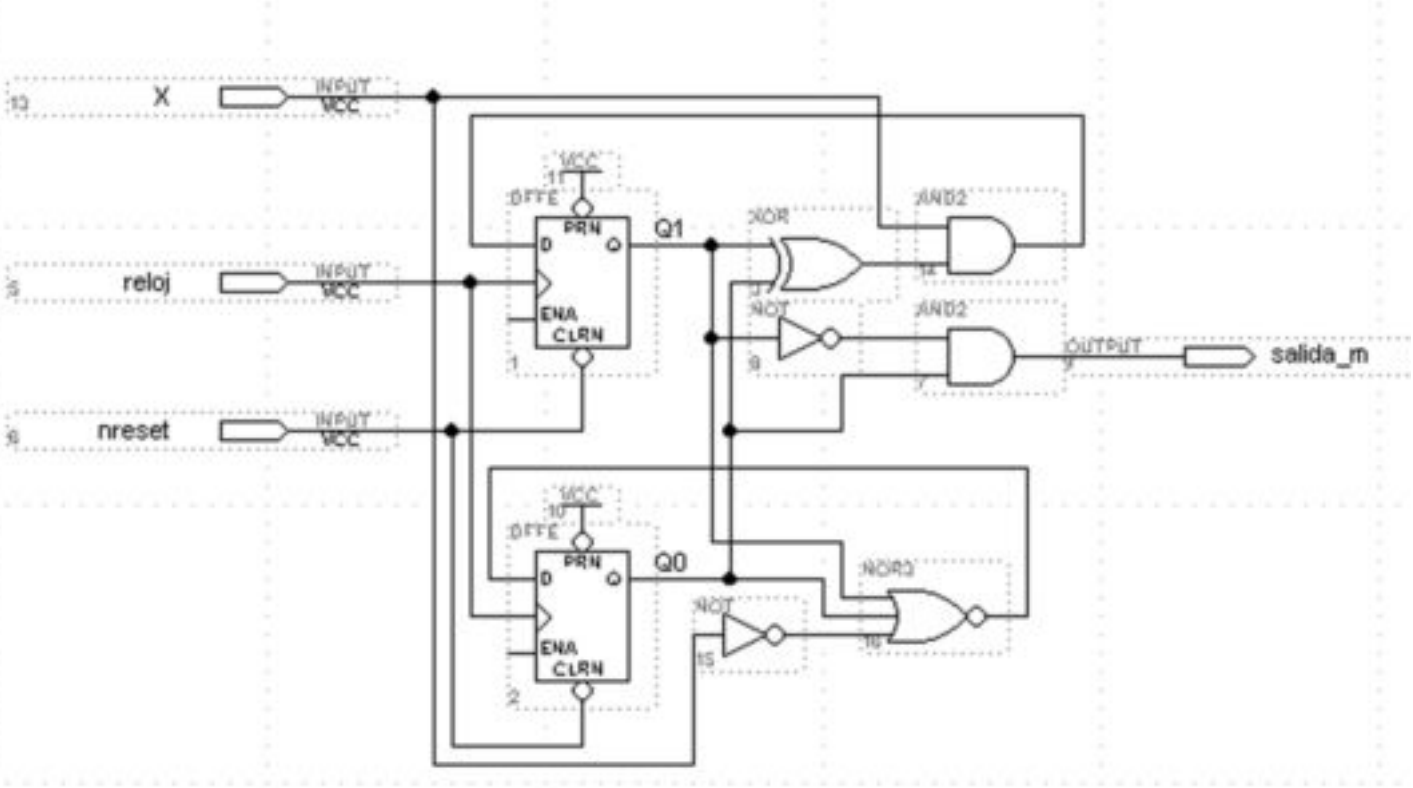
Es un circuito secuencial, ya que consta de dos elementos de memoria (los flip-flops mencionados) y, por lo tanto, trabaja con una señal de reloj que, en este caso, proviene del pin 83 del EPM7128, el que se conecta al oscilador a cristal de la placa principal vía el jumper J7.

De esta manera, en cada período del reloj de 4 MHz (cada 250 nanosegundos), el circuito va a muestrear la entrada "X". Cuando detecta que esta señal cambia de "0" lógico a "1" lógico, pone la salida "salida_m" (hasta el momento, en "0" lógico) a "1" lógico durante un ciclo de reloj (durante 250 ns) y, luego, dicha salida va nuevamente a "0" lógico.

De esta manera, lo que los alumnos logran es un detector de flancos de señal "positivos" o "ascendentes" y lo registran con un pulso "positivo" (de 0" va a "1" y vuelve a "0", luego de un período de reloj). Esto se denomina monoestable disparado por flanco positivo.

Para la implementación, el grupo de alumnos crea un proyecto denominado "monoestable.gdf": y abre un nuevo archivo de trabajo que salvan con el mismo nombre.

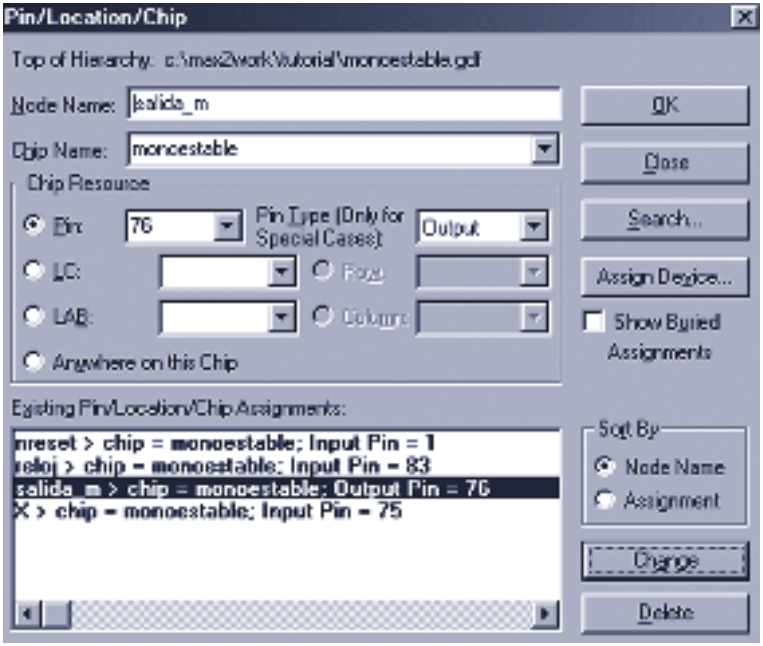
El circuito que van a implementar es el siguiente: 


X
INPUT
VCC
reloj
INPUT
VCC
nreset
INPUT
VCC
DFFE
PRN
D
Q
ENA
CLRN
Q1
Q0
XOR
NOT
AND2
NOR2
OUTPUT
salida_m

Está formado por dos flip-flops tipo D que seleccionan de la librería *prim* dentro del menú *Symbol* en la opción *Enter symbol* con el nombre de *dffe*.

El resto de los componentes son compuertas básicas, que los alumnos también obtienen en esa misma librería:

- compuertas *and* de 2 entradas con el nombre de "and2",
- inversores con el nombre de "*not*",
- una *nor* de 3 entradas como "nor3" y
- una compuerta *or*-exclusiva como "*xor*".



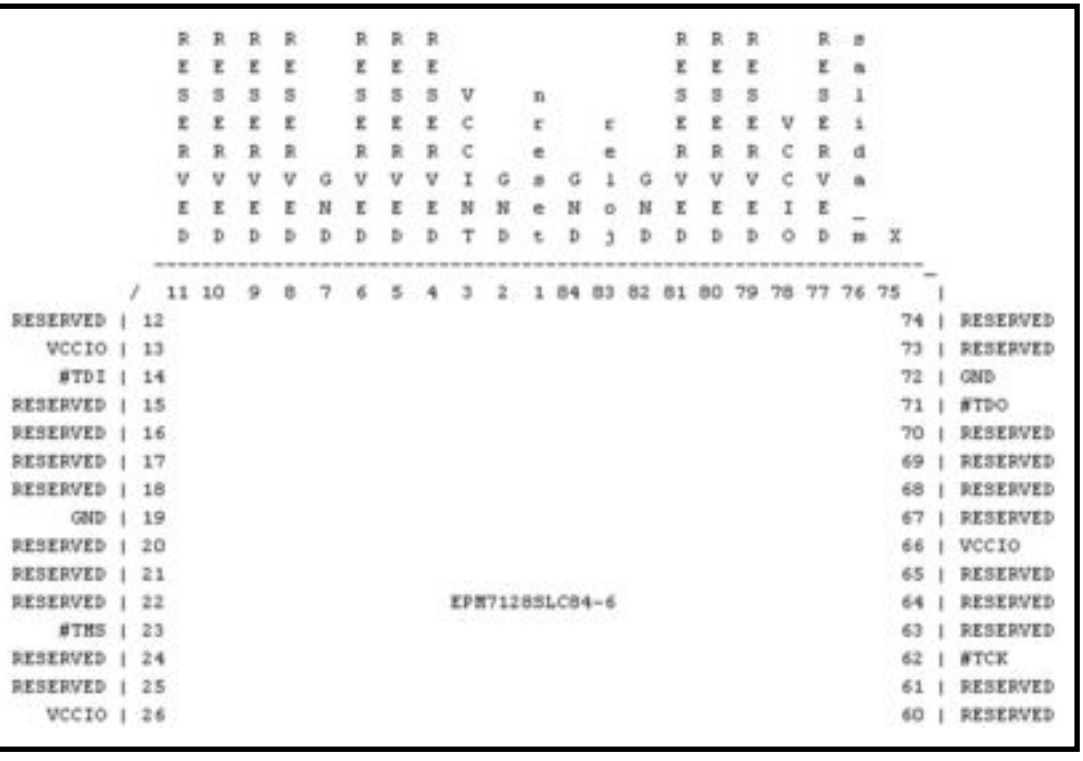
Compilan y, luego, realizan la programación empleando el archivo "monoestable.pof".

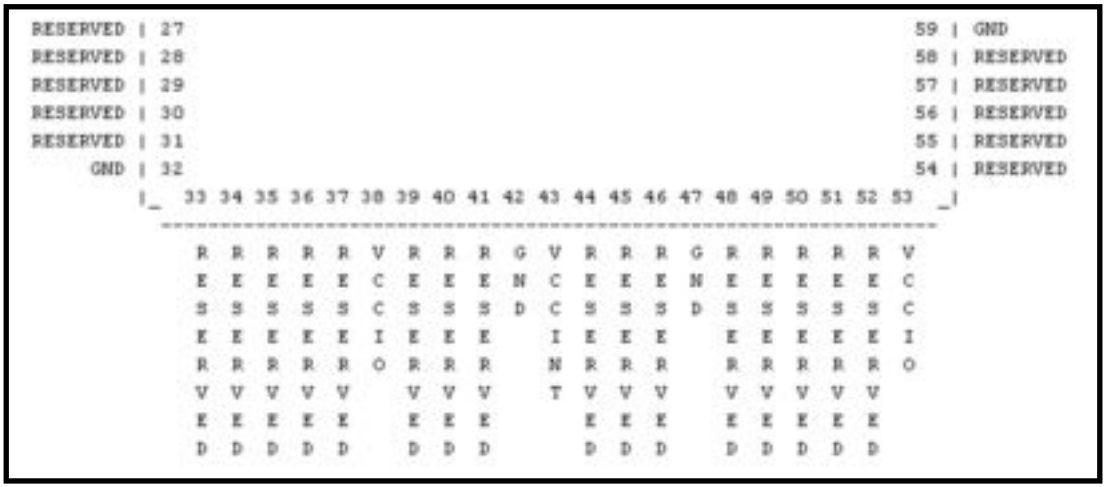
De forma análoga a como lo hicieron en los proyectos anteriores, seleccionan el chip EPM7128 y le asignan los pines:

```
** INPUTS **

                                    Shareable
                                    Expanders      Fan-In    Fan-Out
 Pin      LC   LAB  Primitive    Code  Total Shared n/a INP  FBK  OUT  FBK  Name
   1       -    -      INPUT   G        0      0   0    0    0    0    0  nreset
  83       -    -      INPUT   G        0      0   0    0    0    0    0  reloj
  75   (118)   (H)    INPUT            0      0   0    0    0    0    2  X
```

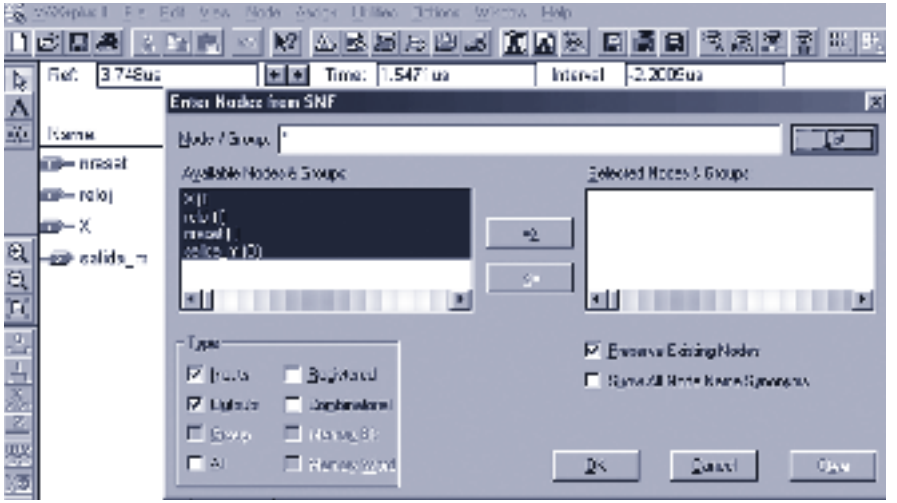
```
** OUTPUTS **

                                    Shareable
                                    Expanders      Fan-In    Fan-Out
 Pin      LC   LAB  Primitive    Code  Total Shared n/a INP  FBK  OUT  FBK  Name
  76     120    H     OUTPUT      t        0      0   0    0    2    0    0  salida_m
```

```
Total dedicated input pins used:                 2/4      ( 50%)
Total I/O pins used:                             6/64     (  9%)
Total logic cells used:                          3/128    (  2%)
Total shareable expanders used:                  0/128    (  0%)
Total Turbo logic cells used:                    3/128    (  2%)
Total shareable expanders not available (n/a):   0/128    (  0%)
Average fan-in:                                  4.00
Total fan-in:                                    12

Total input pins required:                       3
Total fast input logic cells required:           0
Total output pins required:                      1
Total bidirectional pins required:               0
Total reserved pins required                     4
Total logic cells required:                      3
Total flipflops required:                        2
Total product terms required:                    4
Total logic cells lending parallel expanders:    0
Total shareable expanders in database:           0

Synthesized logic cells:                         0/ 128   (  0%)
```

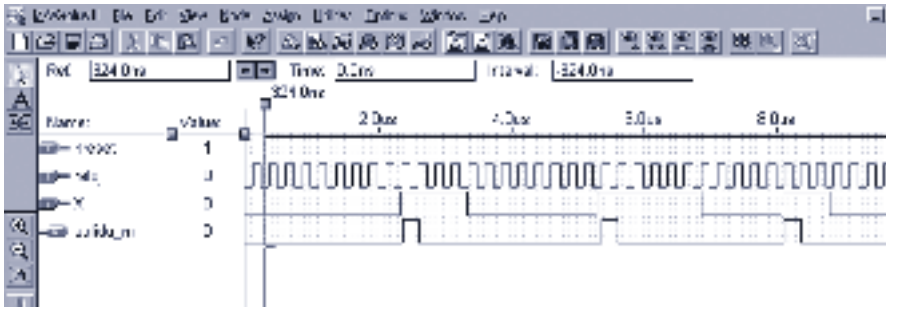


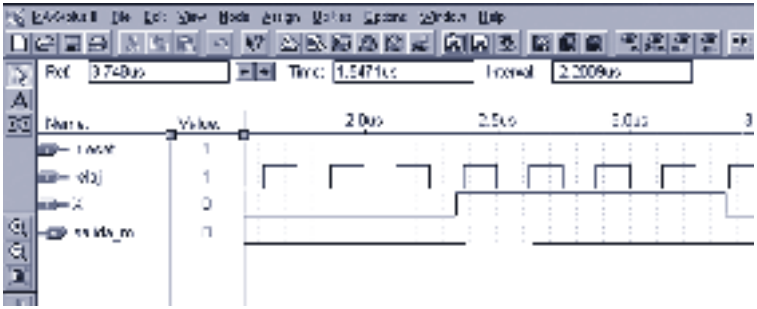


En este caso, realizan las simulaciones generando un archivo "monoestable.scf" y seleccionando -desde el menú *Node*, en la opción *Enter Nodes from* SNF- las señales X, reloj, *nreset* y *salida_m*.



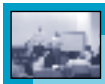
Veamos una simulación en la que se hace un zoom para analizar con más detenimiento los pulsos generados a la salida, cuando aparece un flanco positivo en la entrada:





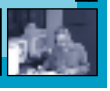
Se puede observar claramente cómo el pulso generado dura un solo ciclo de reloj, al ser detectado el flanco ascendente en "X".

Para este caso -como se trata de una simulación-, los alumnos cambian el reloj a 200 ns de período, en lugar de los 250 ns que corresponderían al reloj de 4 MHz. Para esto, se posicionan sobre el símbolo de la señal reloj y presionan el botón derecho del mouse para seleccionar las opciones overwrite → clock.



Sus alumnos pueden modificar el circuito para que se dispare con el flanco negativo (lo que se logra, simplemente, negando la entrada "X".

Una segunda propuesta es que modifiquen el circuito para que, además, el pulso de salida al detectarse dicho flanco, sea "negativo", es decir, esté normalmente en "1" y cambie a "0", sólo durante un ciclo de reloj (Para esto, necesitarían incluir otro negador más a la salida del circuito anterior).



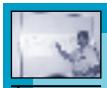
Proyecto 7: Combinación de los proyectos 4, 5 y 6 en un mismo diseño

Este proyecto tiene como finalidad plantear a los alumnos el problema de cómo es posible integrar en un solo chip varios proyectos realizados en forma separada, comprobando hasta dónde llega la flexibilidad del diseño lógico cuando se emplea este tipo de tecnología, cómo se reduce drásticamente el tiempo de diseño y cómo se supera la dificultad de tener que replantear todo de nuevo cada vez que es necesario realizar un proyecto diferente.

En este caso, los alumnos crean un nuevo proyecto "tresproyectos.gdf", en el que copian los esquemáticos de los tres proyectos anteriores tal como están definidos -es decir, manteniendo sus asignaciones-.

La manera más fácil de realizarlo es partir del proyecto del contador con divisor incorporado, al que agregan los circuitos del sumador y del monoestable mediante *Copy* y *Paste*.

Luego, realizan las asignaciones a los pines faltantes, compilan el proyecto y programan el chip.



Si prevemos que no haya superposición de uso en los pines de entrada y salida, podemos mantener las mismas asignaciones.

Para otros casos que implementemos, verificamos que no haya dobles asignaciones. De todas maneras, si existe alguna incompatibilidad, el compilador va a darnos un mensaje de error.

Pero, lo que el compilador no puede saber es qué es lo que le conectamos a cada uno de los pines del chip.

Proyecto 8: Uso del entrenador como herramienta de análisis de circuitos

Hasta aquí hemos planteado el empleo del equipo para realizar diseños de circuitos digitales.

Existe, además, una variante que es la de usar el software de simulación temporal del MAX+PLUS II como una herramienta de entrenamiento para el análisis de circuitos; es decir, para conocer el funcionamiento de un circuito mediante su simulación temporal a través de la excitación de las entradas y del posterior análisis de la evolución de las salidas.

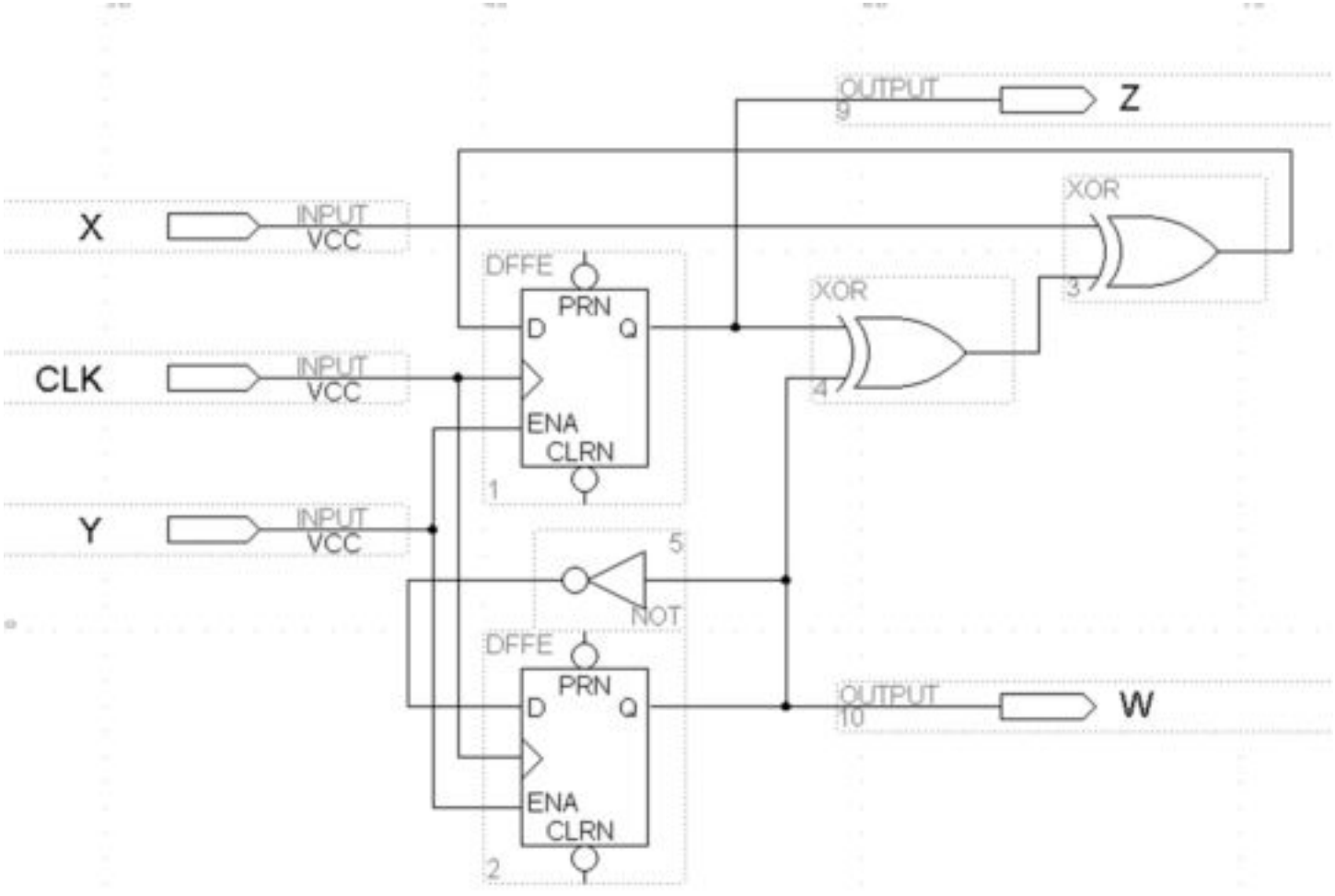
Esta posibilidad es válida tanto para analizar cómo funciona algo tan elemental como una compuerta and o algo un poco más complejo como un flip-flop tipo "D", como para indagar en circuitos más complejos como contadores, registro de desplazamiento e, inclusive, microprocesadores.

Veamos cómo hacerlo, por ejemplo, con un circuito secuencial formado por dos flip-flop tipo "D" y 3 compuertas; las entradas son "X", "Y", "CLK" y las salidas "W" y "Z".

Mediante la simulación vamos a determinar su funcionamiento y, por lo tanto, a saber qué es o, lo que es lo mismo, qué hace.

El proyecto se llama "análisis.gdf".

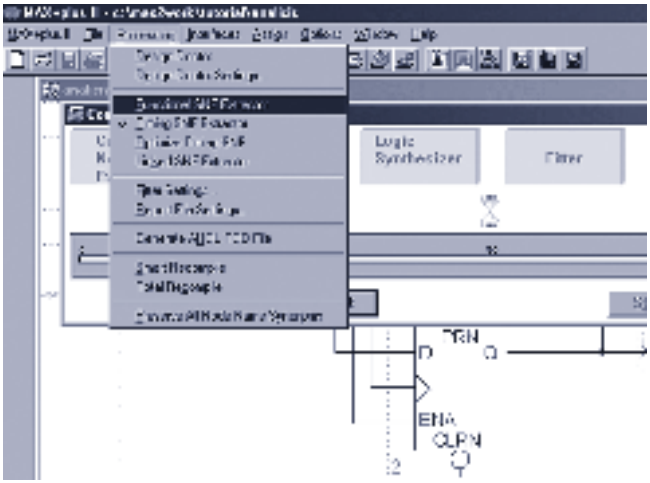
El esquema del circuito: ⟶


X
INPUT
VCC
CLK
INPUT
VCC
Y
INPUT
VCC
DFFE
PRN
D
Q
ENA
CLRN
1
DFFE
PRN
D
Q
ENA
CLRN
2
NOT
5
XOR
4
XOR
3
OUTPUT
9
Z
OUTPUT
10
W

Dibujamos el circuito empleando la librería de símbolos *prim*; en ella:

- el flip-flop corresponde al símbolo *dffe*,
- el inversor a *not*,
- las compuertas *or*-exclusiva a *xor*.

Para realizar el análisis, corresponde hacer una compilación funcional; en ella no es importante asignar un dispositivo o pines al proyecto, ya que no lo vamos a sintetizar sino sólo a analizar para saber cómo funciona.



Luego de esto, presionamos *start* en la ventana de compilación. Al terminar, aparece un cartel que informa el proceso desarrollado; aquí, los mensajes de *warning* sólo indican que el usuario no especificó el dispositivo a utilizar y, por lo tanto, lo hizo el compilador eligiendo el EPM7032 -que es el más chico de la serie MAX7000S-.

Ya estamos listos para realizar la simulación. Creamos un nuevo archivo con extensión "scf" y lo salvamos como "análisis.scf".

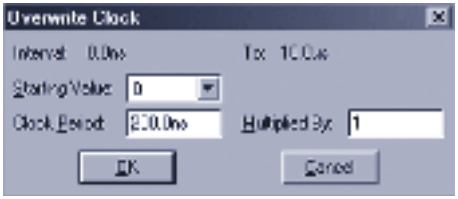
Antes de simular, consideramos las variables en juego. En este caso, son "X", "Y", "CLK" como entradas, y "W" y "Z" como salidas.

Por comodidad, cambiamos el tiempo final de simulación de 1.0 µs -que es en defecto- a 10.0 µs.

A partir del esquema, inferimos que se trata de un circuito secuencial sincrónico, ya que las entradas de reloj de los flip-flop están unidas entre sí y provienen de una entrada común. Por otra parte, la entrada "Y" está conectada a las entradas de habilitación de reloj de cada flip-flop.

Desde el menú que se despliega al presionar el botón derecho del mouse sobre el nombre CLK, generamos el reloj con 200 ns de período.

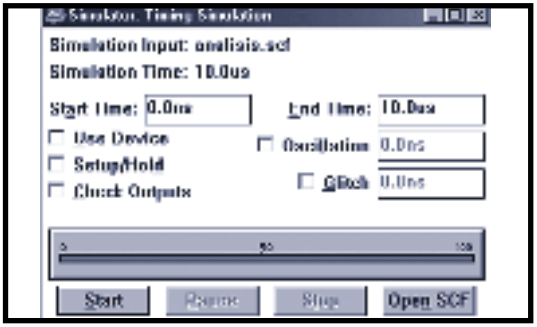
Elegimos *Overwrite clock*. Al hacer clic, aparece una ventana que nos permite concretar el cambio de período.



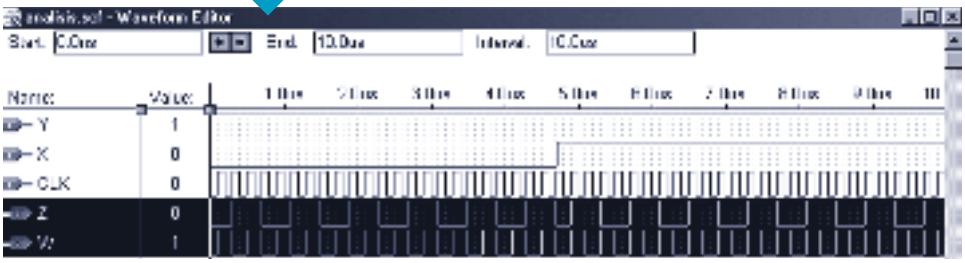
Ponemos a "Y" siempre en "1", para habilitar a los flip-flop a que cuenten.

La entrada que queda es "X", una incógnita. Podemos empezar a probar, dejándola la mitad del tiempo en"0" y el resto en "1".

Con todos los elementos definidos, corremos la simulación.



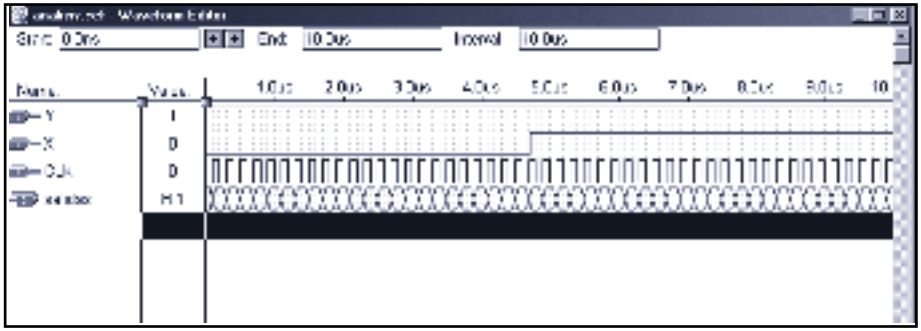
Y obtenemos un gráfico:



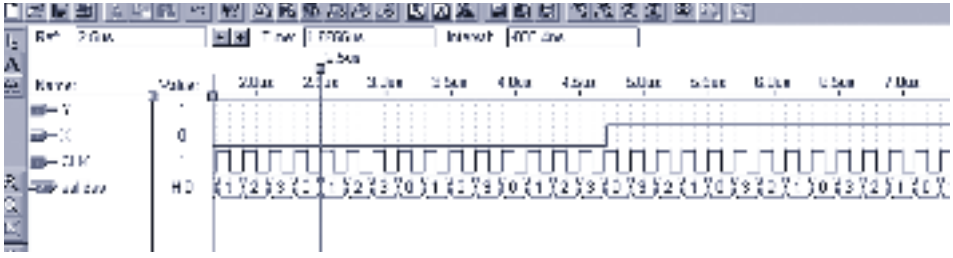
Observamos que la salida "W" es una onda cuadrada de mitad de frecuencia que el reloj de entrada y que "Z" es, también, una onda cuadrada pero de mitad de frecuencia que "W".

Podemos agrupar las dos salidas en una sola gráfica. Haciendo clic sobre "Z" y arrastrando hasta "W", quedan ambos gráficos en "negro". Haciendo clic con el botón derecho del mouse, se despliega un menú donde aparece la opción *Enter group*. En la ventana que aparece, damos el nombre de salidas al nuevo gráfico y seleccionamos la opción *Hex* -formato hexadecimal-.

Aparece, entonces, el gráfico de "salidas" en lugar de "W" y "Z".



Como no se ve ninguna numeración, hacemos un zoom y nos centramos alrededor del cambio de "X" cuando pasa de "0" a "1", para analizar los gráficos.



En este gráfico podemos apreciar que las salidas desde el inicio de la simulación hasta que "X" cambia, van tomando valores "0", "1", "2" y "3", que se repiten cíclicamente.

Al cambiar "X" a "1", las salidas toman valores en sentido contrario, es decir desde "3", "2", "1" y "0", volviendo a repetir este patrón.

Como conclusión:

El circuito analizado es un contador binario de 2 bits, que tiene una entrada de control "X" que, cuando es "0", hace que el contador cuente en forma progresiva; y, cuando "X" es "1", cambia a conteo regresivo.

La superación de dificultades

En estos últimos párrafos es nuestra intención plantearle algunas recomendaciones, que se agregan a las indicaciones que le hemos acercado para cada etapa de diseño y desarrollo del entrenador.

En la etapa de armado de las placas:

- Verifique que todas las conexiones de alimentación sean las correctas, antes de insertar el chip.
- Verifique que todas las vías (interconexiones entre pistas de ambos lados) estén bien soldadas.
- Realice la prueba de alimentación antes de colocar el EPM7128 y el oscilador a cristal.
- Verifique que la fuente de alimentación de 12 V de continua tenga el positivo en el centro del conector. Porque, si no es así, se aplica tensión inversa a la plaqueta principal; y, si bien está incluido el diodo 1N4007 como protección ante una inversión de polaridad, es recomendable que no deje de efectuar esta prueba.
- Alimente el equipo con una batería de 9 V de continua. Aplíquela respetando la polaridad que corresponda -considerando la misma advertencia que antes-.
- Considere que, si los proyectos incluyen diodos led y displays, éstos consumen alrededor de 10 mA por cada segmento, con lo cual puede disminuir rápidamente el tiempo de vida útil de la batería.
- Respete la posición relativa entre el chip y el zócalo PLCC, al realizar el montaje.

En la etapa de instalación del software:

- Recuerde que es necesario pedir la licencia a la empresa proveedora y que recibirá por e-mail un archivo de licencia a ser insertado en el programa. Como esta licencia sirve sólo para el programa que sea instalado en el disco rígido cuyo número de serie coincide con la licencia, va a tener que repetir este procedimiento para usar el programa en otra PC.

En la etapa de desarrollo de los proyectos:

- Es extremadamente importante que verifique que las asignaciones de pines en el EPM7128 sean las correctas. Si un pin asignado como salida se conecta a la sa-

lida de otro componente, el chip puede dañarse permanentemente.

- Considere que la gran flexibilidad de estos dispositivos permite configurar los pines como entrada-salida; pero, tenga en cuenta que el software no puede prever qué es lo que el usuario va a conectar ni dónde lo hará.
- No aplique cargas mayores a 1 mA (un miliampere) a los pines del chip configurados como salida. Por esta razón, en los ejemplos de control de diodos led se emplean los integrados ULN2803, ya que éstos sí están preparados para manejar grandes cargas.
- Tenga en cuenta que el programa *MAX+PLUS II* tiene un archivo de ayuda -*Help*- muy poderoso, que presta gran ayuda en cuanto a las dudas que puedan surgir en el momento del diseño.
- Prevea que el programa contiene todas las herramientas de desarrollo que ofrece la empresa para su línea de productos de lógica programada, pero que la licencia gratuita da posibilidad de editar, simular y programar sólo ciertos dispositivos. La lista de éstos y las autorizaciones correspondientes a qué operaciones se pueden realizar, están visibles en el menú *Option* → *License Setup*. El programa empleado para el diseño del chip EPM7128 de nuestro equipo contiene muchas más herramientas de las que hemos detallado aquí; sólo hemos presentado aquellas básicas para poder realizar los proyectos propuestos.

Sus alumnos pueden utilizar la opción de ayuda -*Help*, en el menú principal-, si están interesados en ahondar más sobre la lógica programada.

5. La puesta en práctica

Esta parte final de nuestro módulo de capacitación contiene un cuadernillo para la evaluación del recurso didáctico que le presentamos y, de las experiencias didácticas y contenidos propuestos a partir de él:

Esta evaluación tiene dos finalidades:

- Brindarle a usted, como docente que utiliza este material, la oportunidad de documentar el seguimiento de las actividades que realice con sus alumnos, a partir de nuestras propuestas y, en función de esta memoria de acciones, propiciar una reflexión acerca de los cambios, mejoras o enriquecimiento de su propia tarea de enseñanza.

- Obtener de su parte, como usuario de este material, información sobre todos los aspectos en torno a los cuales gira la propuesta.

Para este relevamiento de información, usted encontrará, a continuación, una serie de cuestionarios organizados básicamente en tablas o matrices para completar. Con los datos que usted exprese en ellos esperamos tener una realimentación que nos permita mejorar todos los componentes de la serie de publicaciones "Recursos didácticos" y enriquecerla con propuestas o documentación complementaria para aquellos docentes que planteen iniciativas, interrogantes o dificultades específicas con relación a la construcción del recurso didáctico, a las actividades de aula, a los contenidos científicos y tecnológicos, a la metodología de enseñanza, a los procedimientos incluidos, a la información sobre materiales y a otros aspectos.

Dada la importancia que esta información de retorno tiene para nuestro trabajo de seguimiento, mejora y actualización, le agradecemos que nos remita el cuadernillo con todas las observaciones, comentarios o sugerencias adicionales que nos quiera hacer llegar. Para ello puede remitirnos una copia, a través de correo postal, a

Área de Monitoreo y Evaluación –CeNET–
Oficina 112
Saavedra 789. C1229ACE.
Ciudad Autónoma de Buenos Aires.
República Argentina.

O, si lo prefiere, solicitarnos el archivo electrónico de las páginas que siguen a evcenet@inet.edu.ar, enviándonos la versión digitalizada de sus respuestas a través del mismo correo electrónico.

Desde ya, muchas gracias.

Identificación del material:

Las dimensiones que se consideran para la evaluación del módulo de capacitación y del recurso didáctico son:

1. Nivel educativo
2. Contenidos científicos y tecnológicos
3. Componentes didácticos
4. Recurso didáctico
5. Documentación
6. Otras características del recurso didáctico
7. Otras características del material teórico
8. Propuestas o nuevas ideas

1. Nivel educativo en el que trabajó el material:

Nivel educativo	EGB 2	EGB 3	Polimodal (*)			Escuela técnica (*)						Trayecto técnico-profesional (*)	Formación profesional (*)	Otra (*)
			1	2	3	1	2	3	4	5	6			
Nivel en el que usted lo utilizó														

Asignatura/espacio curricular en el que usted lo utilizó:.....

.....

(*) Por favor, indique la modalidad, la orientación, la especialidad, etc.

.....

2. Contenidos científicos y tecnológicos trabajados:

.....

.....

.....

.....

.....

3. Componentes didácticos:

3.1. Testimonios (situaciones problemáticas) presentados en el material

	Sí	No	Otro[1]
a. ¿Le resultaron motivadores para iniciar las actividades propuestas?			
b. ¿Le facilitaron el desarrollo de contenidos curriculares que usted tenía previstos?			
c. A su criterio, ¿están vinculados con el recurso didáctico que se le propone desarrollar?			
d. ¿Le facilitan la organización de situaciones didácticas para el trabajo de los contenidos científicos y tecnológicos propuestos?			
e. El nivel de las situaciones problemáticas que se plantean, ¿es el adecuado al nivel educativo para el que está previsto?			
f. En caso negativo, ¿permiten adecuaciones para ser trabajados en el nivel educativo de sus alumnos o en otro nivel educativo?			
g. Los testimonios iniciales, ¿permiten generar diferentes soluciones (soluciones tecnológicas o didácticas)?			

En caso que su respuesta sea negativa (en cualquier ítem), le pedimos que nos indique por qué (señale el número del ítem a que corresponde su comentario).....

.....

.....

.....

.....

.....

Otro (indique el ítem al que corresponde el comentario): .....

.....

.....

.....

.....

.....

[1] Utilice esta opción para indicar que agregará comentarios al final de este sector de la matriz.

3.2. Estrategias

A partir de la utilización de las propuestas de trabajo en el aula contenidas en el material y del recurso didáctico con el que se asocian, le solicitamos que nos indique (tomando como referencia su forma de trabajo anterior a disponer del material), cómo resolvió las actividades consignadas en la tabla siguiente:

3.2.1. Contextualización de la estrategia didáctica Con respecto a su forma habitual de trabajo, usted logró:	Mejor	Igual	No aplicado[2]	Incorporado[3]
a. Determinar las capacidades, habilidades, conocimientos previos necesarios para iniciar las actividades propuestas.				
b. Organizar, asociar, relacionar los conocimientos científicos y tecnológicos para resolver un problema tecnológico.				
c. Recortar (identificar) los contenidos científicos y tecnológicos a trabajar con sus alumnos para el desarrollo de un sistema/producto tecnológico como el propuesto por el material.				
d. Vincular estos conocimientos con los saberes previos de los alumnos.				
e. Establecer la secuencia adecuada de los contenidos científicos y tecnológicos, y de los procedimientos para generar una solución tecnológica (la propuesta por el material u otra diferente).				
f. Organizar una experiencia didáctica integrando conocimientos científicos y tecnológicos, metodología de resolución de problemas y procedimientos propios del trabajo tecnológico.				
g. Otras (que haya incorporado o hecho mejor con el recurso).				

[2] No aplicado: No lo hizo antes ni ahora con este recurso didáctico.

[3] Incorporado: Integró la estrategia a sus clases a partir de la utilización del recurso didáctico propuesto.

3.2.2. Desarrollo de la estrategia didáctica Con respecto a su forma habitual de trabajo, usted logró:	Mejor	Igual	No aplicado	Incorporado
h. Encuadrar la tarea a partir de la formulación de uno (o varios) problemas.				
i. Explicitar consignas de trabajo que plantean una situación problemática.				
j. Organizar las actividades de aprendizaje atendiendo a las etapas propias de la resolución de problemas.				
k. Utilizar técnicas de trabajo grupal.				
l. Promover el trabajo colaborativo y cooperativo.				
m. Otras (que haya incorporado o hecho mejor con el recurso).				

3.2.3. Aspectos cognitivos (proceso de aprendizaje de sus alumnos) Con respecto a su forma habitual de trabajo, usted logró:	Mejor	Igual	No aplicado	Incorporado
n. Estimular a sus alumnos en la búsqueda de información e investigación en torno al problema eje del material.				
o. Promover la consulta a variadas fuentes de información.				
p. Rescatar, incorporar los aportes del grupo para identificar aspectos o variables críticas del problema.				
q. Evaluar los conflictos cognitivos propios del proceso de aprendizaje.				
r. Detectar, evaluar, la comprensión asociativa.				
s. Promover la reflexión sobre las actividades realizadas y las estrategias utilizadas en cada parte del proceso.				
t. Otras (que haya incorporado o hecho mejor con el recurso).				

4. Recurso didáctico:

4.1. Construcción del recurso didáctico

Tomando en cuenta la finalidad prevista en el material para el recurso didáctico (equipamiento o software), le pedimos que nos indique si, a partir de la propuesta contenida en el material:

4.1.1. Utilizó:

a. ☐ **Un equipo ya construido, según la propuesta del material.**

b. ☐ **Un software.**

c. ☐ **Otro que ya tenía disponible (de características similares).**

d. ☐ **Ninguno.**

Si su respuesta fue "d." indíquenos la razón, por favor:

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

4.1.2. ¿Realizó todo el proceso de construcción del recurso didáctico con sus alumnos? (Conteste este apartado en caso de que haya construido un equipo igual al propuesto. En caso contrario, pase al apartado 5 "Documentación")

Sí	No

4.1.3. En caso de que su respuesta sea afirmativa, le pedimos que nos indique:

	Sí	No
a. ¿Pudo seguir sin dificultades los procedimientos indicados en el "Manual de construcción"?		
b. La secuencia indicada, ¿fue la adecuada para la construcción?		
c. El grado de complejidad, ¿fue el apropiado para el nivel educativo a que se dirige el recurso?		
d. Los contenidos científicos asociados, ¿son pertinentes para el desarrollo del recurso propuesto?		
e. Los contenidos tecnológicos asociados, ¿son pertinentes para el desarrollo del recurso propuesto?		
f. Con sus alumnos, ¿construyó el recurso didáctico siguiendo el proceso y la metodología de resolución de problemas?		
g. ¿Siguió todos los procedimientos propuestos para la construcción pero incorporó sus propios contenidos científicos y tecnológicos?		
h. Por el contrario, ¿hizo adaptaciones en los procedimientos de construcción pero mantuvo los mismos contenidos?		
i. ¿Realizó la construcción siguiendo las actividades de aula propuestas en el material?		
j. ¿Diseñó sus propias experiencias en función de su grupo de alumnos?		

¿Completó todas las etapas del proceso de construcción propuesta?

Sí	No

En caso negativo, indíquenos a qué fase llegó:

a. ☐ **Planificación.**

b. ☐ **Diseño en dos dimensiones.**

c. ☐ **Construcción, armado.**

d. ☐ **Ensayo y control.**

e. ☐ **Superación de dificultades** (evaluación del funcionamiento, siguiendo las indicaciones y la lista de control que brinda el material).

f. ☐ **Construcción de otro equipo que se adapta más a sus necesidades curriculares** (Si marcó esta alternativa, lo invitamos a responder, directamente, el apartado 4.1.5.).

4.1.4. Complete este ítem sólo si realizó el proceso de construcción del equipo siguiendo los procedimientos indicados en el Manual. Si no fue así, lo invitamos a responder el apartado 4.1.5.

Acerca de los materiales, herramientas e instrumentos:

	Sí	No
a. La especificación de los materiales para la construcción, ¿fue suficiente para conseguirlos?		
b. ¿Utilizó los mismos materiales (en calidad y tipificación) indicados en la documentación?		
c. ¿Reemplazó materiales, instrumentos, componentes, piezas, etc., sin alterar el resultado final previsto en el material?		
d. La especificación de las herramientas a utilizar, ¿le resultó adecuada?		
e. La cantidad de herramientas indicadas, ¿fue la necesaria?		
f. Los instrumentos, ¿estuvieron bien especificados?		
g. El tipo y cantidad de instrumentos, ¿fueron los adecuados para armar el recurso didáctico?		

4.1.5. En caso de que usted haya construido un recurso didáctico diferente al propuesto por el material de capacitación, le pedimos que nos indique si la razón fue:

a. ☐ **El propuesto no se ajustaba a sus necesidades curriculares.**

b. ☐ **No pudo conseguir los materiales o instrumentos indicados.**

c. ☐ **No pudo interpretar el manual de construcción.**

d. ☐ **Otra** (Por favor, especifíquela).

.....

.....

.....

.....

.....

.....

4.1.6. ¿Qué características específicas destacaría en este recurso didáctico diferente al propuesto por el material, que sus alumnos han construido. (Marque todas las opciones que considere necesarias):

a. ☐ **Se ajusta mejor a los contenidos curriculares que necesita trabajar.**

b. ☐ **Es más económico.**

c. ☐ **Permite su reutilización** (mediante el desarme y armado, en función de necesidades didácticas).

d. ☐ **Es más adaptable** (a diversos usos).

e. ☐ **Otra** (Por favor, especifique): .....

.....

.....

.....

.....

f. Descripción del recurso didáctico construido: .....

.....

.....

.....

.....

.....

g. Indique las principales diferencias con el equipo propuesto (estructurales, funcionales, didácticas): .....

.....

.....

.....

.....

.....

4.2. Utilización del recurso didáctico

4.2.1. ¿Cómo utilizó el recurso didáctico (hecho por usted o ya construido), en las experiencias didácticas que concretó? (Puede marcar todas las opciones que crea necesarias)

a. ☐ **Aprovechando todo el proceso y la secuencia de construcción propuestos en el material.**

b. ☐ **Aplicándolo** (como algo ya completo) **a la solución de problemas diferentes al propuesto en el material.**

c. ☐ **Utilizándolo como un sistema tecnológico** (ya construido) **en las funciones para las que está pensado** (manejo de las variables, control de operaciones, etc.).

d. ☐ **Otra** (Por favor, especifique)**:**

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

4.2.2. Ya sea que haya desarrollado el recurso didáctico con sus alumnos según las especificaciones del material, ya sea que haya construido otro diferente o que haya utilizado un equipo ya construido, en relación con las actividades que usted venía realizando, la utilización del recurso didáctico propuesto por el material le permitió (seleccione la opción que coincida con sus experiencias):

Con respecto a su forma habitual de trabajo, este recurso didáctico le permitió a usted, como docente:	Mejor	Igual	No aplicable[4]	Otro[5]
a. Integrar contenidos científicos y tecnológicos en la solución de situaciones problemáticas de carácter tecnológico.				
b. Diseñar situaciones de enseñanza y de aprendizaje centradas en la resolución de problemas tecnológicos.				
c. Planificar y promover en sus alumnos la organización del trabajo (planificación y secuenciación de tareas), según el proceso tecnológico.				
d. Favorecer la identificación de aspectos o variables críticas de una situación problemática.				
e. Organizar las actividades de manera que facilite la toma de decisiones por parte de los alumnos (determinación y selección de alternativas, opciones de diseño, materiales, etc.).				
f. Organizar la actividad de sus alumnos en función de soluciones diversas a los problemas planteados.				
g. Agregue otras que usted considere haber logrado de una mejor manera con este recurso didáctico				

[4] NA: No aplicable; es una actividad que no realizó antes ni ahora.

[5] Otro: Recuerde utilizar esta opción para indicar que agregará comentarios al final de este sector de la tabla.

Con respecto a su forma habitual de trabajo, este recurso le permitió a los alumnos (habilidades intelectuales):	Mejor	Igual	No aplicable	Otro
Capacidad de planificar				
h. Identificar variables o aspectos fundamentales de un problema tecnológico.				
i. Organizar su trabajo en etapas (identificar y seguir la secuencia de operaciones de un proceso).				
j. Ejecutar las actividades en los plazos o etapas previstas.				
k. Seleccionar materiales, herramientas y piezas, de acuerdo con las necesidades del diseño.				
l. Anticipar y resolver dificultades que podrían surgir en el proceso.				
m. Prever puntos críticos de todo el proceso.				
n. Agregue otras que considere que sus alumnos alcanzaron mejor con este recurso didáctico				

Capacidad para tomar decisiones	Mejor	Igual	No aplicable	Otro
o. Analizar alternativas en función de un problema.				
p. Seleccionar alternativas en función de las restricciones planteadas en el problema, o en el contexto de enseñanza y de aprendizaje.				
q. Adecuar la propuesta para la solución del problema planteado.				

r. Agregue otras que considere que sus alumnos alcanzaron mejor con este recurso didáctico

Capacidad de aplicar y transferir	Mejor	Igual	No aplicable	Otro
s. Interrelacionar los datos, técnicas y procedimientos en el diseño de la solución.				
t. Utilizar técnicas de representación adecuadas al equipo que se construye o en el ya construido que se utiliza.				
u. Integrar los conocimientos científicos y tecnológicos en los momentos pertinentes para el diseño de la solución.				
v. Relacionar, ensamblar componentes en la secuencia adecuada.				
w. Utilizar de manera correcta la simbología y los lenguajes propios de la tecnología (representación gráfica, simbólica, etc.).				
x. Transferir conocimientos científicos y tecnológicos en otras actividades similares.				
y. Agregue otras que considere que sus alumnos alcanzaron mejor con este recurso didáctico				

Otro (Por favor, exprese aquí los comentarios que tenga, identificando el ítem con la letra que corresponda):

5. Documentación (Material teórico, manual de procedimientos y propuestas didácticas):

5.1. ¿Cómo calificaría los aportes del material recibido (encuadre y desarrollo teórico, y experiencias propuestas para el aula)?

	MV[6]	V	PV
a. Por su potencialidad didáctica (sugerencias, propuestas de trabajo en el aula, papel motivador, etc.).			
b. Para sus necesidades curriculares (desarrollo de los contenidos y experiencias previstas en su planificación).			
c. Para organizar, planificar, concretar experiencias didácticas relacionadas con problemas de Educación Tecnológica.			
d. Para renovar, actualizar, ampliar (subraye el que se ajusta más a su experiencia) los contenidos que desarrolla en su área/ disciplina.			
e. Para trabajar conocimientos científicos y tecnológicos de manera asociada a un problema tecnológico.			
f. Para organizar experiencias de aprendizaje en torno a la utilización de recursos didácticos.			
g. Para utilizar un recurso didáctico en el marco de experiencias didácticas organizadas en función de la resolución de problemas.			
h. Para integrar mejor contenidos científicos y tecnológicos en la solución de problemas de carácter tecnológico.			
i. Para estimular la generación creativa de otros recursos didácticos.			

Otras (Especifíquelas, por favor)

.....

.....

.....

.....

.....

.....

.....

[6] Escala= MV: Muy valioso / V: Valioso / PV: Poco valioso

5.2. Manual de procedimientos para la construcción y el funcionamiento del recurso didáctico

En caso de que haya seguido los procedimientos contenidos en el Manual (ya sea para hacer un equipo igual o uno diferente al propuesto), le pedimos nos indique si:

	Sí	No	Otro
a. ¿Pudo seguir todos los procedimientos descriptos, sin dificultad?			
b. ¿La secuencia descripta le resultó la adecuada?			
c. ¿La secuencia establecida le planteó alternativas según algún criterio (disponibilidad de los materiales, trabajo de contenidos específicos, etc.)?			
d. ¿La finalidad (para qué sirve) del equipo está indicada con claridad?			
e. ¿Se establecen cuáles son los contenidos (científicos o tecnológicos) que se asocian al equipo a construir?			
f. ¿Se determina la relación entre conocimientos implicados, procedimientos a seguir, materiales a utilizar y experiencias posibles de realizar?			
g. ¿Considera que la relación anterior es pertinente (es la que corresponde) para la construcción que se propone?			
h. ¿La descripción de los procedimientos le facilitaron la organización de las experiencias de trabajo con sus alumnos?			
i. ¿Pudo seguir las indicaciones para la puesta en funcionamiento?			
j. ¿Todas las indicaciones para el uso son claras?			

Por favor, fundamente sus respuestas negativas o agregue los comentarios que crea pertinentes (identifique el ítem a que se refiere):

.....

.....

.....

Otro (identifique con la letra que corresponda el ítem sobre el que hace observaciones)

.....

.....

.....

6. Otras características del recurso didáctico:

6.1. Constructivas (Por favor, conteste sólo si realizó el proceso de construcción). Indique si el proceso de construcción reúne las siguientes características:

	Sí	No
a. Simplicidad. Es sencillo de construir por parte de los alumnos.		
b. Economía. Es posible hacerlo con materiales de bajo costo.		
c. Compatibilidad. Todos los componentes, bloques y sistemas permiten ser integrados entre sí.		
d. Acoplabilidad. Puede ser unido o combinado con otros recursos didácticos.		
e. Sencillez. Permite combinar diferentes tipos de materiales (madera, cartón, plástico, otros similares).		
f. Facilidad de armado y desarmado. Permite, sencillamente, realizar pruebas, correcciones, incorporación de nuevas funciones, etc.		

Si su respuesta es negativa en alguna de ellas, indique por qué (Por favor, identifique su comentario con la letra del rasgo aludido): .....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

6.2. Técnicas (Por favor, complete tanto si construyó el equipo como si utilizó uno ya construido)

	Sí	No
a. Portabilidad. Puede ser utilizado en el taller, aula, laboratorio.		
b. Modularidad. Puede ser adaptado a diversos usos; para trabajar diversos contenidos curriculares o para realizar diferentes experiencias didácticas; para aprendizaje, demostraciones, análisis, etc.		
c. Reutilización. Posee partes, componentes, bloques o subsistemas que pueden ser desmontados para volver a su estado original, y usados en sí mismos o en forma independiente.		
d. Incrementabilidad. Puede complejizarse agregando piezas o completando el sistema para mejorar su funcionalidad, rendimiento, precisión o calidad.		
e. Aplicabilidad múltiple. Como sistema tecnológico, permite que usted seleccione las variables con las que desea trabajar (algunas de las que maneja el sistema, todas las previstas o agregar otras).		

Si su respuesta es negativa en alguna de ellas, indique por qué, identificando su comentario con la letra correspondiente:

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

6.3. Didácticas (Por favor, complete tanto si construyó el equipo como si utilizó uno ya construido)

	Sí	No
a. Congruencia. Tiene relación con los testimonios de realidad incluidos en el módulo de capacitación.		
b. Pertinencia. Los componentes, bloques funcionales y sistemas son adecuados para el trabajo con los contenidos curriculares de la educación técnico-profesional.		
c. Integración. Posibilita el tratamiento asociado de los conocimientos científicos y tecnológicos propuestos en el material.		
d. Escalabilidad. Es posible utilizarlo con proyectos o problemas con diferentes niveles de complejidad.		
e. Complejidad creciente. Las soluciones alcanzadas para una parte del problema, sirven de base para las siguientes o permite que, agregando componentes, sea utilizado como solución a problemas más complejos.		
f. Adaptabilidad. Permite su adaptación a soluciones diversas en torno a las problemáticas planteadas.		

Si su respuesta es negativa en alguna de ellas, indique por qué, identificándola con la letra correspondiente:

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

7. Otras características del material teórico:

¿Cómo calificaría el diseño del módulo escrito (desarrollo de contenidos científicos y tecnológicos, y propuestas de experiencias didácticas)?

	MB[7]	B	R	M
a. Formato gráfico del material (distribución del contenido, márgenes, distribución de texto e imágenes, inserción de gráficos, diseño gráfico global, etc.).				
b. Lenguaje utilizado (claridad, adecuación al destinatario).				
c. Organización (secuencia entre cada parte).				
d. Adecuación al destinatario (evidencia que se toma en cuenta que es un material para ser trabajado en un ámbito escolar).				
e. Pertinencia de los conocimientos científicos con las problemáticas planteadas.				
f. Pertinencia de los conocimientos tecnológicos con las problemáticas planteadas.				
g. Vinculación (pertinencia) del recurso didáctico que propone con las situaciones didácticas planteadas.				
h. Congruencia (vinculación) de los contenidos propuestos con el recurso didáctico.				
i. Aporte metodológico para enriquecer sus estrategias didácticas.				
j. Aporte teórico (en general) para su trabajo docente.				
k. Valor motivador para el trabajo con sus alumnos.				
l. Valor orientador para generar sus propios recursos didácticos.				
m. Concepción innovadora para el trabajo didáctico en la educación técnico-profesional.				

Si marcó la opción "Malo", le pedimos que nos explique por qué:

.....

.....

.....

[7] Escala= MB: Muy bueno / B: Bueno / R: Regular / M: Malo

8. Propuestas o nuevas ideas:

Tanto para los autores de este material, como para el CeNET como institución responsable de su elaboración y distribución, una de las finalidades más importantes es suscitar en los educadores nuevas ideas, aplicaciones o propuestas creativas a partir de la lectura o el trabajo con el módulo.

En función de ello, le solicitamos que nos indique:

Si a partir del módulo (contenido teórico y recurso didáctico) usted, en su calidad de (marque todas las opciones que correspondan):

a. ☐ **docente a cargo de un grupo de alumnos**

b. ☐ **directivo**

c. ☐ **responsable de la asignatura:**

.....

d. ☐ **lector del material**

e. ☐ **otro (especifique):**

.....

.....

ha generado nuevas ideas o propuestas:

Respecto de los contenidos (independientemente del recurso didáctico):

	Sí	No
a. Organización de su asignatura.		
b. Contenidos científicos y tecnológicos (formas de asociarlos, ampliarlos, desarrollarlos, etc.)		
c. Planificación de las experiencias didácticas.		
d. Trabajo con resolución de problemas.		

Otras (Por favor, especifique en qué ámbitos ligados con los contenidos ha generado estas nuevas ideas o propuestas):

Si su respuesta fue afirmativa le pedimos que la amplíe:

En relación con el recurso didáctico. Le pedimos que nos relate (libremente) las nuevas ideas o propuestas que el trabajo con este material le ha suscitado:

	Sí	No
¿Puso en práctica alguna de estas ideas o propuestas?		

¿Cuál/es?

En caso negativo, por favor, indíquenos por qué:

Títulos en preparación de la serie "**Desarrollo de contenidos**".

- Colección: **Tecnología química en industrias de procesos**
  - El aire como materia prima
  - El azufre como materia prima
  - Los minerales como materia prima –bauxita y minerales de hierro

- Colección: **Construcciones**
  - Construcción de edificios. Cómo enseñarla a través de la resolución de problemas
  - Construcciones en hormigón armado: tecnología, diseño estructural y dimensionamiento

- Colección: **Telecomunicaciones**
  - Técnicas de transmisión banda base aplicadas a redes LAN y WAN
  - Cálculo de enlaces alámbricos

- Colección: **Materiales**
  - Fundamentos y ensayos en materiales metálicos

- Colección: **Tecnología en herramientas**
  - Historial de las herramientas de corte
  - Diseño y fabricación de herramientas de corte

- Colección: **Electricidad, electrónica y sistemas de control**
  - Instalaciones eléctricas
  - Familia TTL (Lógica transistor-transistor)
  - Familia lógica CMOS

MINISTERIO de EDUCACIÓN CIENCIA y TECNOLOGÍA PRESIDENCIA de la NACIÓN

UNDP Argentina

inet Instituto Nacional de Educación Tecnológica