UNIVERSIDAD DE MENDOZA

FACULTAD DE INGENIERÍA

MAESTRÍA EN TELEINFORMÁTICA

"Laboratorios remotos, un aporte para su diseño y gestión"

Autor. Osvaldo L. Marianetti

Director: Dr. Carlos García Garino

2006

INDICE GENERAL

AGRADECIMIENTOS

Este trabajo desarrollado durante los últimos 10 meses no hubiera sido posible sin la colaboración de quienes de alguna u otra manera me ayudaron a concretarlo. Por este motivo debo agradecer profundamente a todos los que me brindaron su apoyo.

A mi director de tesis, Carlos Garcia Garino, por todo el tiempo que le dispuso a mi tarea, por sus aportes, sugerencias y correcciones. Por su dedicación, que fue más la de un compañero de trabajo que la de un asesor.

A las autoridades de la Facultad de Ingeniería de la Universidad de Mendoza, Decano Salvador Navarría, Secretaria Administrativa Cristina Párraga y Secretario Académico Alfredo Iglesias. Por su constante apoyo y por la confianza que depositan en mis actividades académicas. En particular a Alfredo, por darle crédito a mis inquietudes tecnológicas.

Al Director de Posgrado de la Facultad de Ingeniería de la Universidad de Mendoza Carlos Palacio por su constante y cordial incentivo a concretar esta tesis.

Al Director General del Instituto Tecnológico Universitario, Guillermo Cruz, por brindarme la posibilidad de capacitarme, compartir experiencias y recursos aquí y en el exterior.

A Oscar León, a quien debo agradecerle además que su colaboración profesional, su amistad.

A mis compañeros del LAPIC, Claudio Careglio, Julio Monetti y Carlos Catania. Por las horas compartidas en el “lugar del pensar”.

Finalmente un agradecimiento muy especial a mis hijos Lucía y Franco, por hacer fácil una de las tareas más difíciles y a mi esposa Sonia, por todo lo compartido en estos últimos 10 meses y en los 19 años anteriores.

RESUMEN

Actualmente es frecuente el uso de las nuevas tecnologías como recurso didáctico alternativo. Es común el empleo de entornos de aula virtual, foros, chats, etc. Sin embargo todas estas ayudas son útiles en entornos de enseñanza-aprendizaje para impartir contenidos conceptuales, presentar pequeñas simulaciones, comparar respuestas de problemas, etc.

En cambio es relativamente complejo disponer de laboratorios para su empleo de manera no presencial. Los laboratorios remotos son una alternativa atractiva, propia de estas nuevas tecnologías, que permite acceder a distancia a los laboratorios. De esta forma se facilita el acceso de los estudiantes a los laboratorios, resolviendo dificultades habituales para su uso presencial: la disponibilidad de recursos, el elevado número de estudiantes por práctica, etc.

Los laboratorios remotos son un entorno didáctico que requiere muchos recursos tecnológicos. En la arquitectura de un laboratorio remoto se necesitan herramientas de gestión, de apoyo didáctico, como así también hace falta hardware y software que permita el acceso remoto interactivo a los componentes y dispositivos del laboratorio físico.

Los dispositivos de control con servicios de Internet embebidos presentan características interesantes para la implementación de un laboratorio remoto, ya que permiten la integración de las funciones más importantes que necesita un sistema digital para lograr la interacción con el medio exterior mediante la adquisición de datos, la toma de decisiones y la actuación sobre elementos de control que modifiquen las condiciones de su entorno.

En esta tesis se propone un modelo conceptual para la descripción y el diseño de estos sistemas. También se desarrolla una alternativa para la

implementación de laboratorios remotos utilizando las facilidades que presentan los dispositivos con servicios de Internet embebidos.

Como ejemplo de estos sistemas, se desarrolla un prototipo de laboratorio remoto de bajo costo en hardware, que permita la realización de prácticas remotas sobre dispositivos lógicos programables.

CAPÍTULO 1

INTRODUCCIÓN

En este capítulo se exponen los objetivos y alcances de esta tesis a partir de la presentación de las posibilidades que aportan Internet y las tecnologías de la información y las comunicaciones al desarrollo de nuevos recursos aplicables en los procesos de enseñanza – aprendizaje, capacitación, formación y experimentación de estudiantes, docentes y profesionales.

Para ello se discute la aplicación de estas tecnologías en la educación a distancia, las aulas no tradicionales, los cursos en línea, los campus virtuales, los laboratorios virtuales y los laboratorios remotos, permitiendo extender a contextos globales los recursos locales.

1.1 Internet como recurso en los procesos de enseñanza – aprendizaje. Posibilidades.

Los investigadores en el campo de la educación plantean algunos interrogantes sobre cuestiones que ayudan a decidir la validez de usar Internet y las tecnologías de la información y las comunicaciones en los procesos de aprendizaje. Resulta interesante responder a las siguientes preguntas:

- ¿Pueden las TICs e Internet mejorar realmente la apropiación de conocimientos en disciplinadas que no estén específicamente relacionadas con sus propias contenidos?
- ¿Pueden las TICs e Internet promover tipos de cursos que se adecuen a distintos estilos de aprendizaje, hacer el aprendizaje accesible a estudiantes no tradicionales o con capacidades distintas?
- ¿Puede las TICs e Internet bajar los costos de la educación, manteniendo la calidad? [1]

En primera instancia la respuesta es “depende”. Depende del nivel de aplicación y utilización que se desea de estos recursos en los procesos de formación, es decir qué y cuánto se pueden usar estas tecnologías. En el caso de Internet, algunos de los posible usos pueden ser:

- Búsquedas en la Web de artículos actualizados, referencias a sitios de interés, etc.
- Promover intercambio de e-mails, suscripción a listas de correos y grupos de discusión y foros.
- Implementar cursos mediante un sitio Web, con una página principal con enlaces a páginas o funciones específicas de la temática desarrollada.
- Utilizar sitios Webs y protocolos de transferencias de archivos para que los estudiantes puedan ejecutar programas, tutoriales o simuladores en línea o descargados en su computadora personal.
- Diseñar bases de datos para que se pueda realizar un seguimiento de las actividades llevadas a cabo por los alumnos.
- Crear un laboratorio virtual donde las animaciones y simulaciones reemplacen a los experimentos físicos.
- Poner en funcionamiento un laboratorio remoto que permita a los estudiantes definir valores de variables para el desarrollo de experiencias en sitios remotos y obtener los resultados de esa experiencias.

La o las utilidades que se quieran aplicar dependerán de la claridad que se posea en los objetivos perseguidos, es decir porqué, para quienes y para qué se deben utilizar estas herramientas.

El éxito en la aplicación de estos recursos también es función del dominio de las tecnologías subyacentes en dichos recursos, ya que involucran componentes que integrados conforman un sistema de considerable complejidad.

1.2 Sitios Webs orientados a procesos de enseñanza - aprendizaje

Un sitio web es una colección de páginas webs con una página principal (homepage), que funciona como índice (index.htm) para el acceso a las demás. En el caso particular de un sitio web orientado a la formación o capacitación, las páginas que contiene sirven para propósitos educativos y de gestión.

Si bien los estilos de la página principal son muy variados, en general se trata de una página estática de presentación de la temática del sitio, datos administrativos y fechas de interés. Además incorpora enlaces a documentos relacionados con el sitio, a presentaciones o demostraciones, software de utilidad y dirección de correo del responsable del sitio. También presenta esta página botones de navegación del sitio.

Algunos componentes comunes en estos sitios son:

- Material de apoyo, demostraciones interactivas y tutoriales.
- Ejercicios y evaluaciones. En este caso es necesario una aplicación que permita tomar las entradas que generen los estudiantes para que pueda recibirlas el responsable de las evaluaciones y además enviarles los resultados a los alumnos. Algunas aplicaciones avanzadas permiten generan evaluaciones en forma dinámica y los estudiantes reciben los resultados en forma inmediata.
- Líneas de comunicación entre estudiantes asincrónicas (BBS) o sincrónicas (chats).
- Bases de datos con las estadísticas de los procesos cumplimentados por cada estudiante o de los accesos al sitio en general. [2]

1.3 Construcción de sitios Webs

La páginas webs se construyen con lenguajes de programación, como por ejemplo el HTML (hipertext markup languaje) para que su contenido codificado

pueda ser transferido por Internet. Las páginas interactivas incluyen código escrito (scripts) en lenguajes de programación como por ejemplo Java. El software del navegador (web browser) decodifica el HTML y los scripts para mostrar los contenidos de la página en la pantalla de una computadora.

Existen herramientas que facilitan la construcción de páginas webs como los procesadores de textos que incluyen la posibilidad de guardar los documentos con formato HTML. Más potentes son los editores de páginas webs, que contienen un conjunto de herramientas para la creación de páginas y sitios webs, incluso con plantillas que ayudan a una construcción rápida de las páginas deseadas. También es posible utilizar software para el diseño de aplicaciones multimedia compatibles con los navegadores. Estas aplicaciones se almacenan en un servidor web y pueden ser ejecutas por un usuario que en su computadora cuente con unos programas instalados, denominados “plug-in”. Un plug-in extiende las capacidades de un navegador. Es un programa que interactúa con otro programa para aportarle una función o utilidad, generalmente muy específica. Este programa adicional es ejecutado por la aplicación principal. Los plug-ins típicos tienen la función de reproducir determinados formatos de gráficos, reproducir datos multimedia, codificar/decodificar emails, filtrar imágenes de programas gráficos. Se utilizan como una forma de expandir programas de forma modular, de manera que se puedan añadir nuevas funcionalidades sin afectar a las ya existentes ni complicar el desarrollo del programa principal. Los plug-ins en general se consiguen gratuitamente y los enlaces a los sitios de descarga forman parte de la información que entrega el sitio a quienes acceden al mismo.

Finalmente se encuentran los sistemas de aulas virtuales, que son aplicaciones que reúnen prácticamente todas las funciones que deben contener los sitios Webs orientados a procesos de enseñanza – aprendizaje. Entre las herramientas que presentan estas aplicaciones se encuentran: agenda de eventos, anuncios para los usuarios registrados con envío de correo

electrónico, publicación de documentos, generación de ejercicios interactivos con evaluación automática, recepción de trabajos presentados, foros de discusión, administración de grupos de usuarios, estadísticas de acceso y resultados.

Otras de las posibilidades más interesantes que presenta Internet en este campo, son los tutoriales, los laboratorios virtuales y laboratorios remotos.

1.4 Laboratorios virtuales

Los tutoriales pueden ser pasivos (incluyendo textos, audios o videos) o interactivos, donde se puede ingresar distintos valores para determinadas variables y observar los efectos que resultan de su aplicación facilitando de esta manera la compresión de conceptos abstractos. Los tutoriales interactivos se dividen en aquellos que requieren un software que se ejecute en una máquina local o aquellos que se pueden ejecutar directamente desde la Web usando Java applets. Un applet es un componente de software que se ejecuta en el contexto de otro programa, por ejemplo un navegador web. El applet debe procesarse en un contenedor, que es proporcionado por un programa anfitrión, mediante un plug-in. Un applet normalmente lleva a cabo una función muy específica que carece de uso independiente. Ejemplos comunes de applets son los applets de Java y las animaciones Flash. Otro ejemplo es el Windows Media Player utilizado para desplegar archivos de video incrustados en los navegadores como el Internet Explorer.

Los laboratorios virtuales son simulaciones de software del comportamiento de dispositivos físicos. Si la simulación es muy detallada pueden constituir un sustituto de los laboratorios tradicionales, en particular si contienen animaciones. Estos laboratorios a los que se accede mediante Internet son un modo de reducir los costos del equipamiento que requieren las prácticas de laboratorio. Usar un software standard de navegación (web browser) para acceder al laboratorio virtual presenta la ventaja que los requerimientos en la

máquina de los estudiantes son mínimos y que la mayoría de los usuarios está familiarizado con las herramientas de navegación. Sin embargo, es necesario que los estudiantes ingresen datos y obtengan resultados de la simulación. El código HTML no permite realizar estas tareas. Para subsanar este inconveniente, las páginas web interactivas con presentaciones gráficas para el ingreso de datos se pueden escribir usando programas CGI (Common Gateway Interface) o Java Applets. Los programas CGI son ejecutables que residen en el servidor web y actúan como un protocolo de comunicaciones entre las aplicaciones del servidor web y la ventana gráfica presentada en el cliente. Los scripts CGI se escriben en lenguajes comunes como Visual Basic, Perl o C++. Los programas CGI residen en el servidor, mientras que los applets de Java o los Java scripts estan incrustados o embebidos en la página web.

1.5 Laboratorios remotos

Los laboratorios remotos que utilizan una interfase web para su implementación son relativamente nuevos. Algunos ejemplos incluyen verificación de circuitos analógicos, laboratorio de medidas remoto, control de robots, otros. Los componentes de estos sistemas son un usuario remoto con una conexión a Internet, un servidor web y un equipo que controla los componentes de laboratorio sobre los que se experimenta. [3]

Algunos de los inconvenientes que presentan los laboratorios presenciales en cuanto a los procesos de enseñanza – aprendizaje se deben a que los recursos en los laboratorios suelen estar disponibles en determinados horarios pocas veces en la semana, elevado número de estudiantes en cada sesión o estudiantes que provienen de localidades ubicadas geográficamente distantes. Otra variable a tener en cuenta es la heterogeneidad de los alumnos en cuanto a edades y habilidades motoras, alumnos que provienen de instituciones de nivel medio donde no han tenido suficiente contacto con el uso adecuado de material de laboratorio. Esta realidad, más la optimización de recursos

materiales y humanos, seguramente se han tenido en cuenta por aquellas instituciones que desarrollaron experiencias en laboratorios remotos.

En la arquitectura de un laboratorio remoto se requieren: herramientas de gestión (administrador de bases de datos para gestionar: datos de usuarios, reserva de recursos, archivos de experiencias o trabajos prácticos), herramientas de apoyo o soporte (páginas dinámicas que permitan seleccionar distintos módulos o experiencias y que posibiliten interactuar con los dispositivos físicos del laboratorio, material multimedial que ayude a la mediación de los contenidos desarrollados en el módulo o la experiencia), herramientas de acceso remoto (hardware y software que permitan el acceso remoto interactivo a las herramientas y dispositivos del laboratorio físico) y hardware de control de los dispositivos físicos del laboratorio. [4]

Estos requerimientos hacen de los laboratorios remotos uno de los entornos didácticos que más recursos tecnológicos necesitan, ya que en el caso de los laboratorios virtuales la interactividad del alumno con los recursos virtuales se logra con herramientas de software. En los laboratorios remotos, además de los recursos de gestión, apoyo y acceso se debe considerar la interacción del usuario con los recursos físicos. Se hacen necesarios componentes de hardware para materializar este requerimiento. Una posibilidad interesante está dada por la tecnología de los sistemas embebidos que ha permitido integrar en una sola pastilla de silicio o en un simple módulo, gran parte de las unidades funcionales de un sistema digital. Esto significa que en un solo circuito integrado (SOC: system on chip) o en un módulo, se incorpora, además de la electrónica necesaria para la ejecución de instrucciones de lenguaje de máquina, aquel hardware que permite la gestión de operaciones de entrada-salida en diferentes modos de transferencia, ya sea mediante simples registros de almacenamiento temporario (puertos) o interfaces de comunicaciones que soportan diferentes protocolos (interfaces seriales o ethernet). También algunos de estos dispositivos cuentan con administración de memoria y

sistemas de archivos. Las especificaciones pueden variar según los fabricantes; sin embargo todos tienen en común la integración de las funciones más importantes que necesita un sistema digital que permita interactuar con el medio exterior mediante la adquisición de datos, la toma de decisiones y la actuación sobre elementos de control que modifiquen las condiciones de su entorno. Actualmente existe una importante oferta de dispositivos con Internet embebida y los costos se tornan accesibles. Algunos ejemplos son los desarrollados por: Rabbitt , Net Silicom, Microchip, entre otros.

Considerando las características intrínsecas de estos dispositivos con Internet embebida, los mismos constituyen una alternativa muy interesante en la construcción de laboratorios remotos, ya que facilitan los procesos involucrados en su diseño y puesta en marcha.

1.6 Objetivos de la Tesis

Los objetivos de esta tesis son:

a) Presentar las características que presentan los laboratorios remotos, no sólo en cuanto a su utilización como recurso didáctico, sino fundamentalmente en cuanto a su diseño e implementación.

b) Presentar un modelo conceptual para la descripción y diseño de laboratorios remotos.

c) Diseñar e implementar un laboratorio remoto de bajo costo, utilizando dispositivos con Internet embebida, orientado a prácticas relacionadas con la educación superior, en particular a las técnicas de diseño de circuitos digitales basados en dispositivos lógicos programables.

1.7 Justificación

Los laboratorios de las cátedras dedicadas a la enseñanza de los circuitos digitales en la formación de grado de ingeniería normalmente cuentan con instrumentos de medición, software de simulación, circuitos integrados de pequeña y mediana escala de integración, placas experimentadoras, kits didácticos de lógica discreta y microcontroladores. En el caso de los

microcontroladores, un gran porcentaje de las prácticas se basa en una familia de microcontroladores particular, seguramente debido a la disponibilidad y los costos accesibles de las herramientas de desarrollo para dichos microcontroladores. No sucede lo mismo cuando se trata de Dispositivos Lógicos Programables. Si bien se pueden obtener programas de compilación y simulación fácilmente, el acceso a los recursos de hardware de estos componentes no es tan sencillo.

Una alternativa para contar con herramientas didácticas para el desarrollo de los contenidos relacionados con Dispositivos Lógicos Programables son los laboratorios remotos. Además del costo, algunos de los inconvenientes que presentan los laboratorios presenciales en cuanto a los procesos de enseñanza – aprendizaje se deben a que los recursos en los laboratorios suelen estar disponibles en determinados horarios pocas veces en la semana, con elevado número de estudiantes en cada sesión. En estos ámbitos educativos este sistema permite un seguimiento detallado de las actividades realizadas por cada alumno.

La presencia del alumno en las aulas y en los laboratorios es imprescindible, pero plantear nuevas alternativas no está en confrontación con lo anterior. Se han de complementar y no sustituir los hábitos educativos.

Este desarrollo forma parte de proyectos que se llevan a cabo en el Instituto de Microelectrónica de la Facultad de Ingeniería de la Universidad de Mendoza y el Laboratorio LAPIC del Instituto Tecnológico Universitario de la Universidad Nacional de Cuyo, dentro del marco de los convenios existentes entre ambas instituciones y se presenta como tesis de la Maestría de Teleinformática de la Facultad de Ingeniería de la Universidad de Mendoza.

1.8 Estructura de la tesis

Este capítulo describe las distintas posibilidades que Internet y las tecnologías de la información y las comunicaciones brindan a los procesos de enseñanza –

aprendizaje y cuales son las herramientas que requiere su aplicación. Luego se proponen y justifican los objetivos de esta investigación.

En el segundo capítulo se presenta una caracterización de los entornos de aprendizaje y en particular de los laboratorios remotos. Se muestran ejemplos en los que se puede apreciar el estado del arte de algunas implementaciones de entornos remotos que permiten el desarrollo de distintas experiencias de laboratorio. En base a los ejemplos se describen los componentes que requiere la arquitectura de los laboratorios remotos.

En el capítulo 3 se sugiere una clasificación para los entornos de prácticas de laboratorio y a partir del análisis de la arquitectura de los laboratorios remotos se propone un modelo conceptual de capas, que puede ser una herramienta útil en los procesos involucrados en el diseño e implementación de laboratorios remotos. Se presenta un ejemplo sencillo donde se aplica este modelo.

El capítulo cuatro describe la implementación de un prototipo con el que se logra el acceso remoto a los recursos de un entorno de desarrollo para la programación de Dispositivos Lógicos Programables (kit con herramientas de software y hardware) y la utilización de un dispositivo con Internet embebida para la verificación remota del comportamiento de los circuitos programados remotamente en el kit. En el diseño de este prototipo se aplica el modelo de capas definido en el capítulo 3.

Finalmente en el capítulo 5 se presentan las conclusiones de este trabajo.

Como complemento se incluyen apéndices referidos a “Microcontroladores”, “Dispositivos Lógicos Programables“ y “Lenguajes de Descripción de Hardware”.

CAPÍTULO 2

LABORATORIOS REMOTOS – ESTADO DEL ARTE

Este capítulo caracteriza los laboratorios remotos y presenta aplicaciones de laboratorios remotos en cursos de educación superior. En todas las soluciones propuestas un usuario remoto (usuario cliente) se conecta a un servidor web dedicado que interactúa con la computadora del laboratorio utilizada para controlar y monitorear los experimentos.

2.1 Descripción de los entornos para experiencias o prácticas de laboratorio

Para describir los posibles entornos en los cuales se pueden llevar a cabo experiencias de laboratorio es posible recurrir a una clasificación según los siguientes criterios:

a) el modo de acceso a los recursos que requiere la experiencia.
b) la naturaleza del sistema físico o los recursos empleados (elementos, dispositivos, materiales de laboratorio).

El modo de acceso puede ser local o remoto. Mientras que la naturaleza del sistema puede ser simulada o real. [5]

Combinando estos criterios se obtienen cuatro clases de entornos que involucran todas las posibles maneras de realizar experiencias de laboratorio.

- Acceso local – recursos reales: este es el modo tradicional de prácticas de laboratorio.
- Acceso local – recursos simulados: no existen recursos físicos reales. Hay un software y una interfase que trabajan en la simulación. Se lo denomina “entorno virtual monousuario”.
- Acceso remoto – recursos reales: Se accede realmente a equipamiento de laboratorio a través de Internet. El usuario puede controlar y operar

en modo remoto este equipamiento mediante una interfase de experimentación. Se lo denomina " laboratorio remoto" o "teleoperación".

- Acceso remoto – recursos simulados: Es similar al acceso local con recursos simulados, salvo que los usuarios operan a través de Internet el simulador. Una diferencia importante es que varios usuarios pueden interactuar simultáneamente con el mismo sistema virtual. Se lo denomina "laboratorio virtual multiusuario"

Los entornos con acceso remoto son de gran interés y elevada demanda porque este tipo de sistemas, sobre todo si permiten interactuar con recursos reales, hacen posible realmente "traer el laboratorio a casa", ya que si el laboratorio remoto está bien construido se puede experimentar con una computadora conectada a Internet en cualquier hora del día, cualquier día del año. Además, la experimentación remota o "teleoperación" no se limita al universo educativo, sino que tanto en la industria como en los centros de investigación el acceso remoto a través de Internet al control de determinados dispositivos es la única opción. La razón de esto es el costo. En algunos casos la adquisición de equipamiento no es posible y el mismo se encuentra disponible en otras organizaciones. Se reducen así costos en compras de equipos o movilizaciones hacia los sitios donde se encuentran los recursos.

Algunas desventajas de estos entornos de aprendizaje es que pueden producir aislamiento, trabajo solitario o falta de motivación. Para reducir estos efectos, los requerimientos básicos del sistema de laboratorio remoto deben ser:

- Instalación y operación simples: el entorno debe tener los medios necesarios para que no se requiera la presencia de un tutor con demasiada frecuencia.
- Acceso a través de Internet: En lo posible la única herramienta necesaria para experimentar debe ser un navegador de Internet.
- Sin costo: el único costo para el alumno debe ser el acceso a Internet.

- Interactividad y realismo: las respuestas del entorno deben promover el interés y motivación del alumno.
- Disponibilidad total: No deberían existir restricciones en el uso del laboratorio. Sólo aquellas que se originen cuando otro estudiante está ocupando el entorno y el tipo de experiencia así lo requiera o en el caso de realizarse tareas de mantenimiento del sistema.

2.2 Posibilidades y características de los laboratorios remotos

Como se indicó en el Capítulo 1, con la ayuda de un laboratorio en línea, los docentes pueden diseñar cursos que combinen la teoría con los experimentos, obteniendo mayor flexibilidad para preparar asignaturas que requieren de prácticas de laboratorio. Se alcanza un mayor aprovechamiento de los equipos de laboratorio. Compartiéndolos, disminuyen los costos por alumno que involucra el valor de la alta tecnología de los mismos, la duplicación de los recursos necesarios, su obsolescencia y la provisión de soporte técnico.

Una consideración importante es que los materiales utilizados para prácticas locales tienen que adaptarse para poder utilizarse en prácticas remotas. Además debe tenerse en cuenta qué sucede ante condiciones de error en el desarrollo de la práctica en modo remoto. Por ello es importante la confiabilidad del vínculo entre el cliente y el servidor, como así también el monitoreo y control de los resultados de los procesos que va llevando a cabo el alumno y su efecto sobre el equipamiento.

Si bien las experiencias remotas no sustituyen las prácticas locales, existen factores que demandan y justifican su aplicación. Algunos de ellos pueden ser:

- Elevado número de estudiantes en relación a los recursos: no todos los participantes de la práctica acceden a los materiales, algunos no pueden terminar sus experiencias debido a los tiempos asignados.

- Educación a distancia: muchos estudiantes requieren este tipo de modelo educativo por limitaciones de tiempo, distancia o discapacidades físicas.
- Recursos económicos insuficientes: con poco equipamiento y poco espacio físico, la realización de prácticas presenciales no se puede desarrollar adecuadamente. [6]

2.3 Ventajas de los laboratorios remotos

Como se comentó en el Capítulo 1, algunas de las características que hacen que los laboratorios remotos sean una alternativa para reducir los problemas existentes en las prácticas de laboratorios son las siguientes:

- Laboratorios accesibles 24 horas al día todos los días del año.
- Los estudiantes no tienen que movilizarse a la sede para realizar sus actividades prácticas.
- Optimización del aprovechamiento de los recursos.
- Acceso a diferentes clases de experimentos independientemente de las limitaciones en los recursos.
- Permiten una aproximación previa a las prácticas locales, de modo de lograr un mejor rendimiento en la experiencia con dispositivos reales.
- Los procesos de aprendizaje se mejoran, ya que se logra un nexo constante entre la práctica y la teoría. Los estudiantes no deben esperar que el laboratorio local esté disponible.
- Adaptación y personalización del entorno de laboratorio a estudiantes con discapacidades.

Otra posibilidad importante es la posible creación de redes de laboratorios remotos interuniversidades.

2.4 Desventajas de los laboratorios remotos

Obviamente el uso de estos entornos también presenta desventajas. Algunas de ellas son:

- No hay contacto físico con el experimento. La inevitable separación espacial reduce la sensación de realismo de la práctica. Por este motivo es importante la correcta utilización de material multimedia que aumente la sensación de presencia de modo que el estudiante perciba los efectos y resultados de sus acciones sobre el componentes reales del laboratorio.
- No todas las experiencias se pueden implementar en la modalidad remota por las características intrínsecas de los elementos que requieren las mismas. Por ejemplo, experiencias de laboratorio en asignaturas como química donde las actividades involucran manipulación de elementos que es difícil su operación remota.
- Inestabilidad e imprevisibilidad en el vínculo entre cliente y servidor. Este factor depende de la calidad del servicio del enlace.
- Cambio de mentalidad del docente y del alumno. Siempre que se adoptan nuevas metodologías es necesaria la predisposición de los actores que intervienen en su aplicación.

2. 5 Ejemplos de laboratorios remotos.

Son muchas las organizaciones que han considerado justificada la incorporación de las posibilidades que brindan los laboratorios remotos en el desarrollo de sus actividades. En particular, las universidades son un ejemplo que confirma lo expresado. Las experiencias concretadas en ese sentido abarcan alguna disciplinas como las que se describen a continuación.

2.5.1 Prácticas de programación de microcontroladores y microprocesadores.

Un laboratorio remoto de microprocesadores y microcontroladores es un entorno orientado al aprendizaje de la programación de sistemas microprocesadores y microcontroladores, con la particularidad de que puede ser accedido a través de Internet desde cualquier ordenador. Este tipo de implementación se presenta en la Figura 2.1.

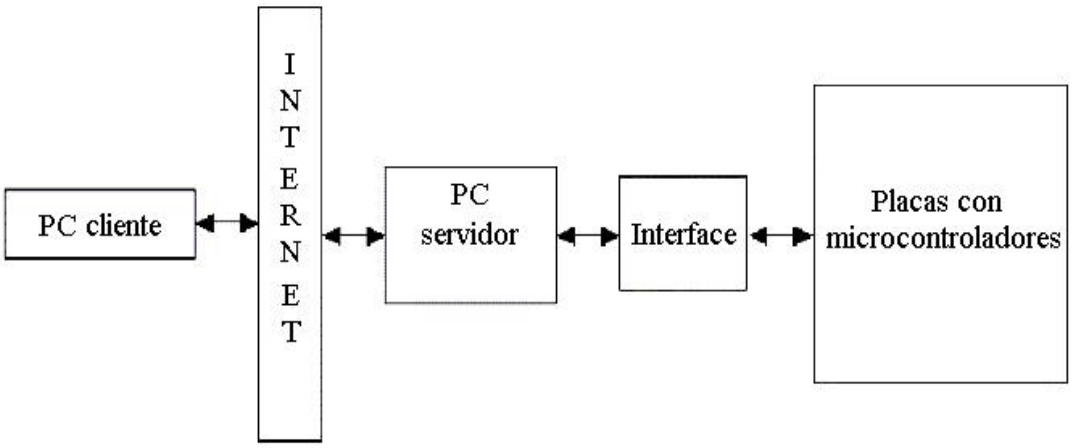


Figura 2.1 - ILboratorio remoto de microprocesadores y microcontroladores

El objetivo de estos sistemas es que el usuario disponga de una plataforma remota de microprocesadores y microcontroladores sobre los cuales pueda aprender a diseñar y programar aplicaciones, además de conocimientos para el diseño de sistemas basados en microprocesadores y microcontroladores.

En algunos casos se usa como plataforma un único microprocesador o microcontrolador y en otros, diferentes plataformas (Intel, Microchip, Motorola, Texas), de modo que los conocimientos sobre microcomputadores que adquiera el usuario de la aplicación sean amplios y no estén supeditados a la plataforma que utilice.

Las prácticas que normalmente permiten hacer estos sistemas es la programación de los recursos de los microprocesadores – microcontroladores para le ejecución de un proceso especificado, como lo plantea A. Grau[7]. Para

ello, el cliente escribe el código y lo envía al servidor que procesa las aplicaciones relacionadas con la programación de los microprocesadores – microcontroladores y le devuelve al cliente los resultados de estos procesos (presencia o no de errores en el código, datos o valores de variables que resultan de la ejecución de la aplicación en el microcontrolador o microprocesador programado).

2.5.2 Prácticas de programación de Dispositivos Lógicos Programables

Un caso similar es el de laboratorios remotos que permiten programar Dispositivos Lógicos Programables (PLDs), ya sea CPLDs o FPGAs en lugar de microcontroladores, como lo presentan I.González et al.[8] y J. Olivares et al.[9]. En estos casos, para la obtención de los resultados de la programación remota de estos componentes se recurre a dispositivos que permitan la visualización de los mismos, como por ejemplo webcams y su software asociado, como se observa en la Figura 2.2.

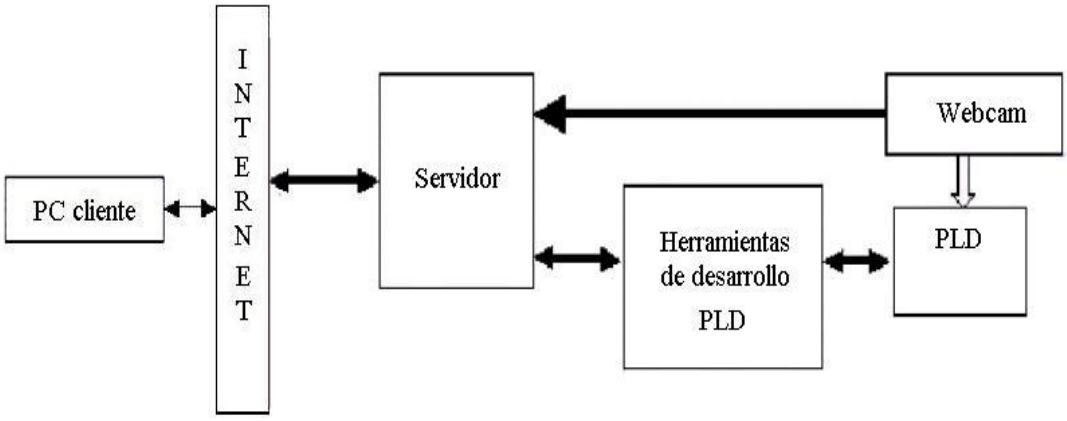


Figura 2.2 - Laboratorio virtual para la programación de PLDs

2.5.3 Laboratorio de experimentos de física para ingeniería.

Existen aplicaciones en las que mediante una computadora personal y una interfase adecuada se controlan dispositivos físicos de laboratorio permitiendo le realización de prácticas relacionadas con la física. A partir de estas

herramientas y una arquitectura cliente – servidor se llevan a cabo las actividades de control y monitoreo remoto que requieren algunas de las prácticas de un laboratorio de física. Un ejemplo se presenta en la Figura 2.3. En base a esta arquitectura se pueden llevar a cabo experiencias como las que presentan A. Azad et al.[10] para un laboratorio de física, como el monitoreo del comportamiento de un motor. En este caso se monitorea remotamente la condición de funcionamiento de un motor eléctrico mediante el análisis del espectro de frecuencia de la vibración de la carcasa del motor. El funcionamiento del motor se controla con una tarjeta de entrada – salida conectada una PC, la tarjeta permite la lectura de la salida amplificada de un acelerómetro.

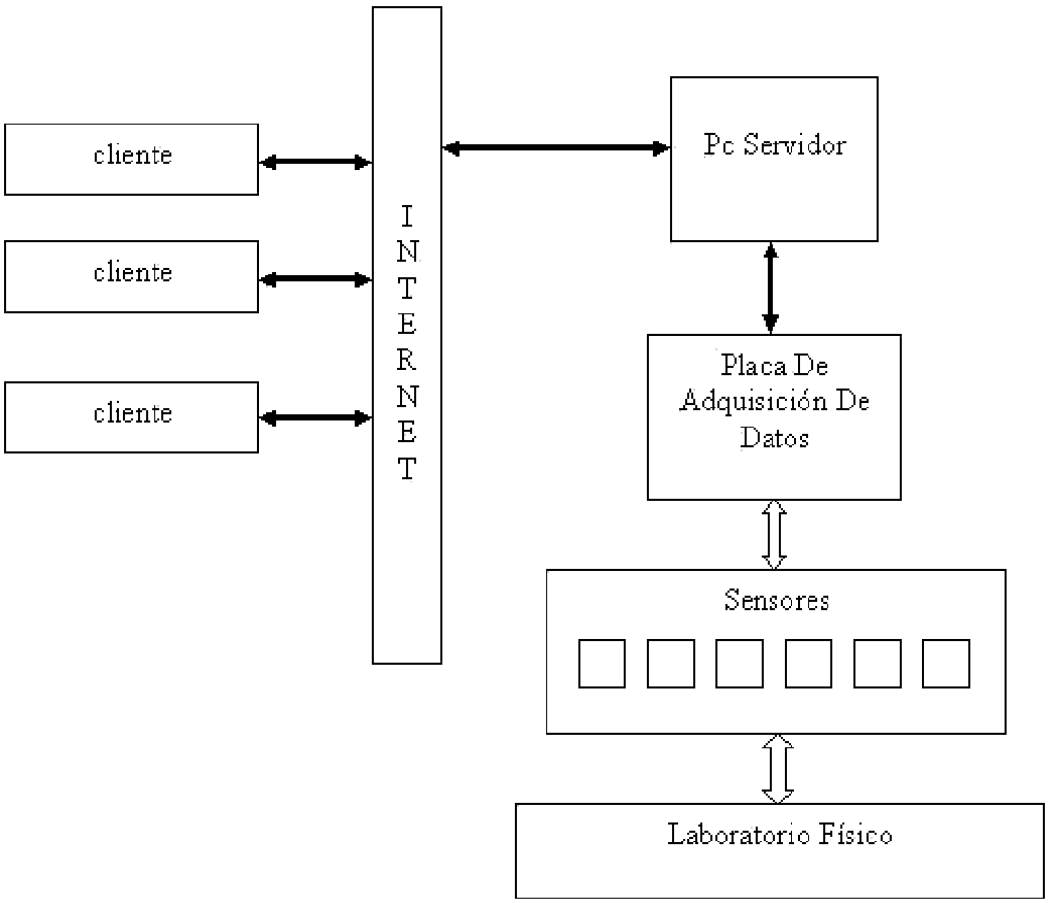


Figura 2. 3 - Laboratorio de experimentos de física para ingeniería.

Esta salida se puede visualizar en el dominio del tiempo y en el dominio de la frecuencia a través de LabVIEW Virtual Instrument [11]. La característica en el domino de la frecuencia se puede analizar para conocer el comportamiento del motor. La ejecución de este experimento se solicita desde una PC cliente y además los datos se almacenan en el disco de una PC remota. Este ejemplo es una muestra de como con la aplicación de distintos tipos de sensores adecuados y placas de adquisición de datos instaladas en una PC, se generan diferentes actividades para experiencias relacionadas con sonido, calor, posición, tiempos, etc., que han expandido sus posibilidades a prácticas remotas.

2.5. 4 Laboratorios de electrónica básica y de potencia

Otros casos de laboratorios basados en web con ejercicios a los que se puede acceder remotamente se han implementado para estudiantes de ingeniería eléctrica o electrónica, donde se introducen tanto en modo teórico y práctico, aspectos fundamentales de la electrónica de potencia, como el caso que desarrollan W. Hurley y C. Kwan Lee[12]. En particular su conformación contiene los elementos que se presentan en la figura 2.4.

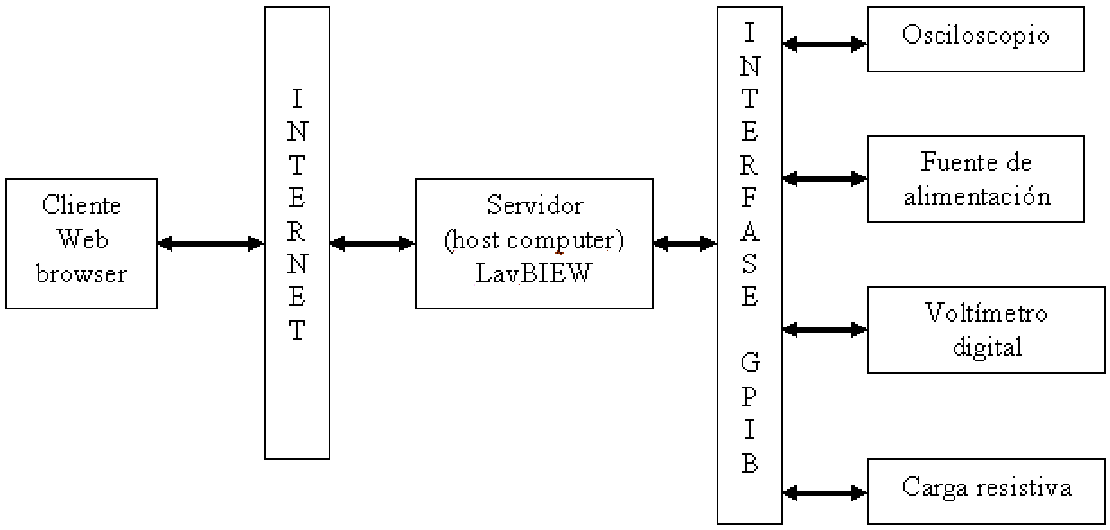
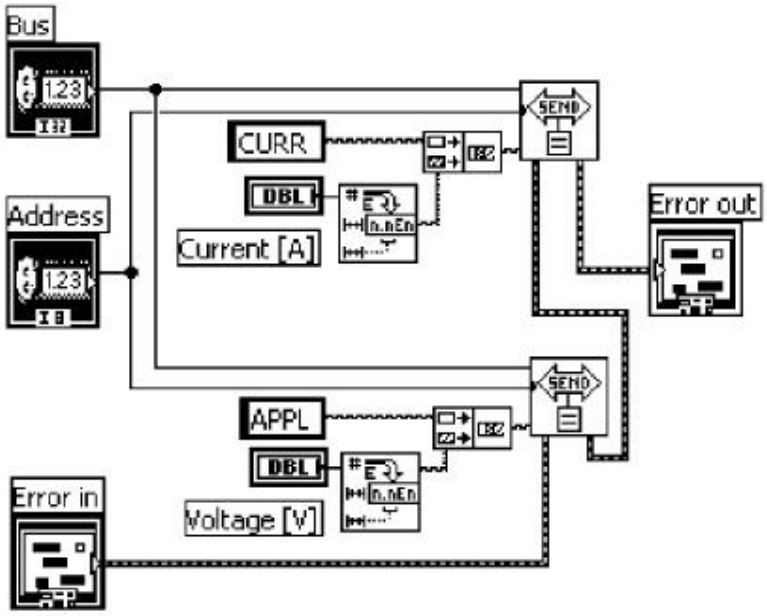


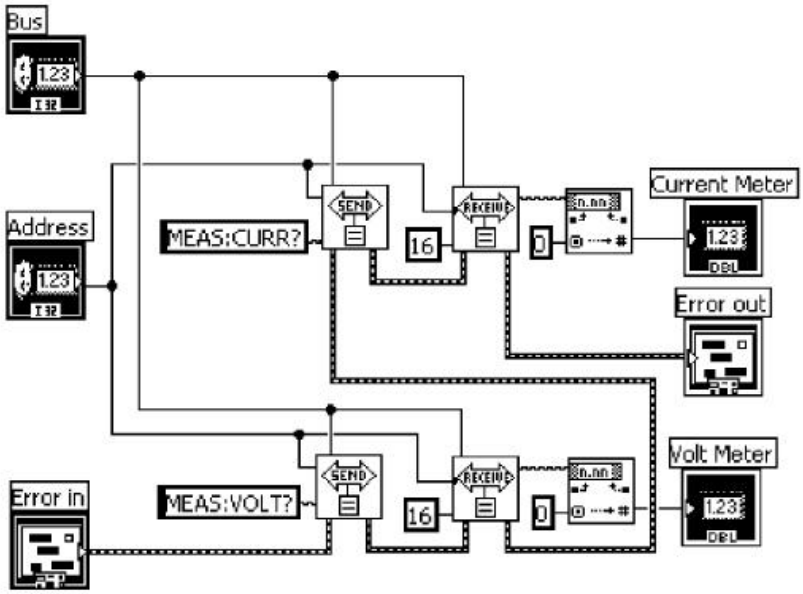
Figura 2.4 - Laboratorio remoto para prácticas de electrónica de potencia.

Esta estructura es la de un laboratorio de tiempo real basado en web, donde todos los instrumentos utilizados en los experimentos son accedidos remotamente y cada estudiante pueda tomar sus propias mediciones mientras va modificando el modo en que sus mediciones son realizadas. Los estudiantes visualizan todos los instrumentos en la pantalla de su computadora y controlan las entradas requeridas para ver el resultado de la acción llevada a cabo. El desarrollo de acceso remoto al laboratorio está basado en instrumentos con bus de interfase de propósito general (GPIB) y el software LabVIEW 7. Todos los instrumentos, incluyendo un osciloscopio de 4 canales, una fuente de alimentación y un voltímetro digital se comunican con el servidor (computadora host) usando LAbVIEW 7 y la interfase GPIB. La corriente de entrada, la tensión de entrada, la corriente de salida, la tensión de salida y la forma de onda del ruido de la tensión de salida son medidas. LabVIEW es un entorno gráfico de desarrollo de National Instruments para la adquisición de datos, el control de instrumentos, el análisis de mediciones y la presentación de datos.

La figura 2.5 muestra dos ejemplos de la configuración virtual de instrumentos (VI), las cuales se usan para enviar comandos y recibir datos desde los instrumentos conectados a la interfase GPIB. En la figura 2.5 (a) se ve el ingreso de valores de voltaje y de corriente máxima a la fuente de alimentación. Las salidas de tensión y corriente de la fuente pueden medirse y recibirse usando un comando de recepción, como se observa en la figura 2.5 (b). Toda la funcionalidad, la apariencia y la configuración del sistema se ensambla por la simple conexión de los diferentes bloques.



(a)



(b)

Figura 2.5 - Configuración del LabWIEW

El LabVIEW Internet Toolkit incorpora la configuración virtual de instrumentos VI mediante un web browser. El panel de control y la instrumentación como aparecen en la interfase del navegador se muestran en la figura 2.6.

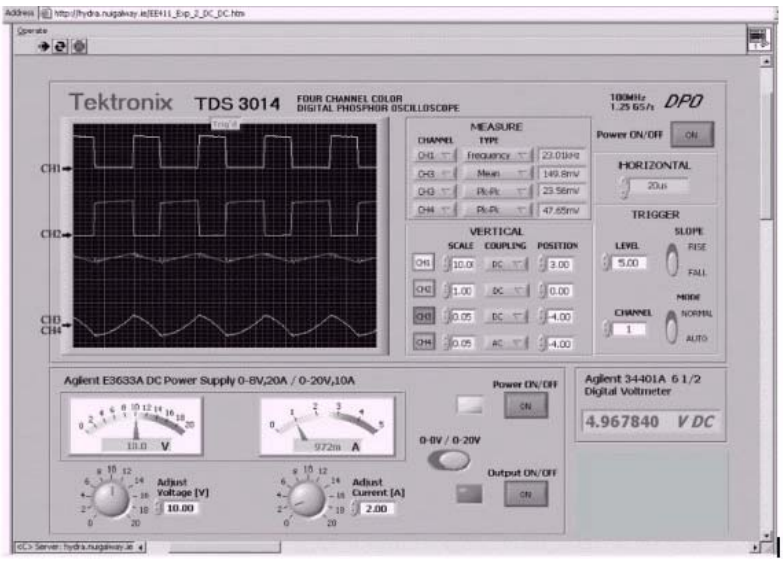


Figura 2.6 - Web browser de LabWIEW

Para este tipo de prácticas de laboratorio también se pueden realizar aproximaciones de pre-laboratorio a través de simulaciones y análisis con aplicaciones como Pspice y/o MATLAB. En las sesiones de laboratorio, los estudiantes observan y graban las formas de ondas de los puntos de prueba y pueden repetir las mediciones para varios valores de la tensión de entrada.

2.5.5 Laboratorios remotos de redes de computadoras e internetworking

Por las características intrínsecas de los componentes que intervienen en su desarrollo, una aplicación difundida de los laboratorios remotos es su utilización como recurso para cursos relacionados con temáticas de redes de computadoras o internetworking. Se ofrecen cursos sobre: Arquitectura de Redes, Telecomunicaciones y Redes WAN, Protocolos de Comunicaciones de Internet, Administración y Seguridad en Redes, Simulación y Análisis de Modelos, etc.

El equipamiento y los recursos que requieren en su implementación son computadoras personales y servidores, dispositivos de redes incluyendo routers y switches, hubs, analizadores de redes LAN y WAN y software de simulación redes. Estos equipos se instalan en racks. Distintas topologías de redes se configuran por la simple conexión de nodos usando los cables adecuados. Un circuito de cables y una regleta de conexión entre los equipos de cada rack permite conexiones entre los distintos racks. La configuración de los dispositivos de redes se hace mediante una interfase de línea de comando con una PC conectada al puerto de consola del dispositivo que se desea configurar, permitiendo que los estudiantes establezcan una comunicación con este componente.

Se puede describir uno de estos sistemas de enseñanza remota a partir de una arquitectura como la que se observa en la figura 2.7.

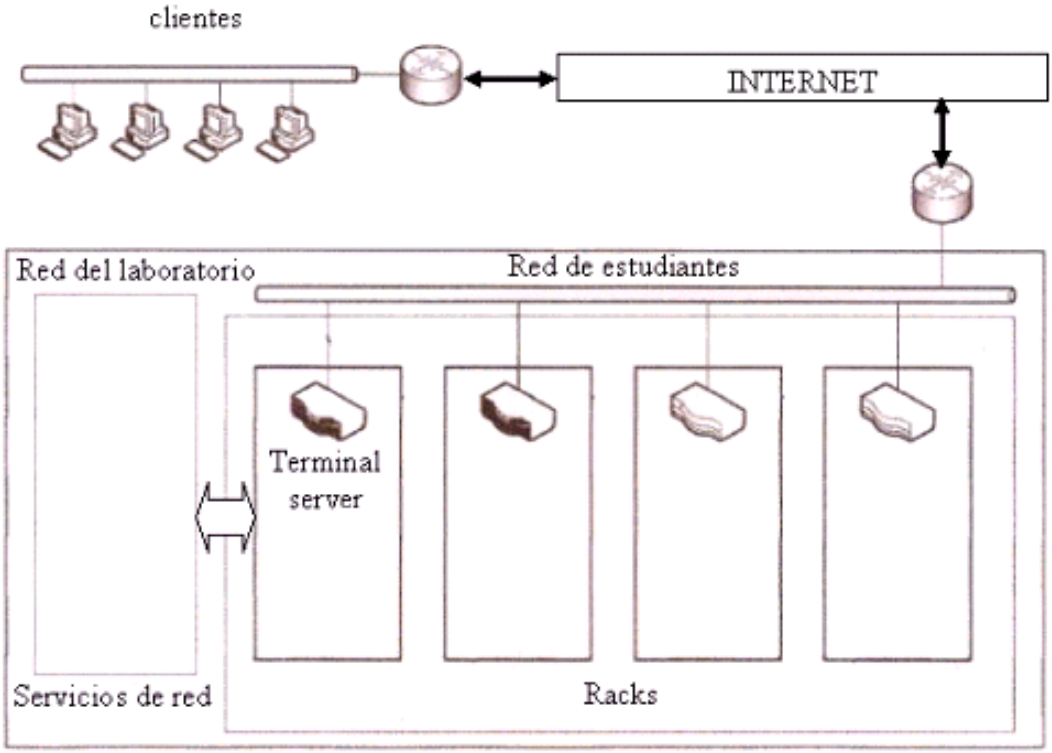


Figura 2.7 - Laboratorio remoto para prácticas de Internetworking

La arquitectura de los laboratorios de redes para prácticas remotas como los que han desarrollado S. Sivakumar et al.[13], consta de los siguientes componentes:

red del laboratorio: se puede construir alrededor de un router multiservicio. Este router puede conmutar paquetes de datos sobre ATM, Frame Relay y/o redes Ethernet permitiendo construir múltiples redes lógicas independientes usando los mismos dispositivos físicos. A cada red lógica se le asigna su propia interfase física, su conjunto de protocolos y su esquema de direcciones. En este caso requiere tres backbones, uno Ethernet, otro ATM y otro Frame Relay. Cada uno con sus switches, que se conectan entre si y a su vez con los racks de los estudiantes.

Red de estudiantes: Consiste en equipamiento instalado en racks. El diseño de esta red soporta sesiones múltiples simultaneas con este equipamiento y los estudiantes pueden acceder remotamente a través de Internet. Este acceso remoto al hardware de los racks se logra con un terminal server que conecta un puerto del dispositivo a Internet. Este acceso a los dispositivos (por ejemplo: switches, routers o analizadores de protocolos) se realiza mediante una interfase Web. Estas páginas webs permiten la validación para el ingreso al sistema mediante un nombre de usuario y una clave de acceso. Cada rack puede contener un router, switches multilayer y hubs.

2.5.6 Telerobótica con fines docentes

Se han implementado sistemas que se basan en una arquitectura cliente-servidor, en la que el cliente puede acceder a una interfaz de experimentación que le permita generar la trayectoria que debe seguir el robot móvil. El servidor se encargará de cargar en la memoria de un robot y ejecutar un programa que permita el control de los servomotores para conseguir que el robot real siga la trayectoria decidida por el cliente, que podrá visualizar el resultado de su actuación mediante imágenes adquiridas por el servidor.

Se muestra en la figura 2.8 un esquema de una arquitectura del sistema cliente-servidor utilizado en experiencias de programación remota de microrobots, que es la sugerida por J.Moreno et al.[14]

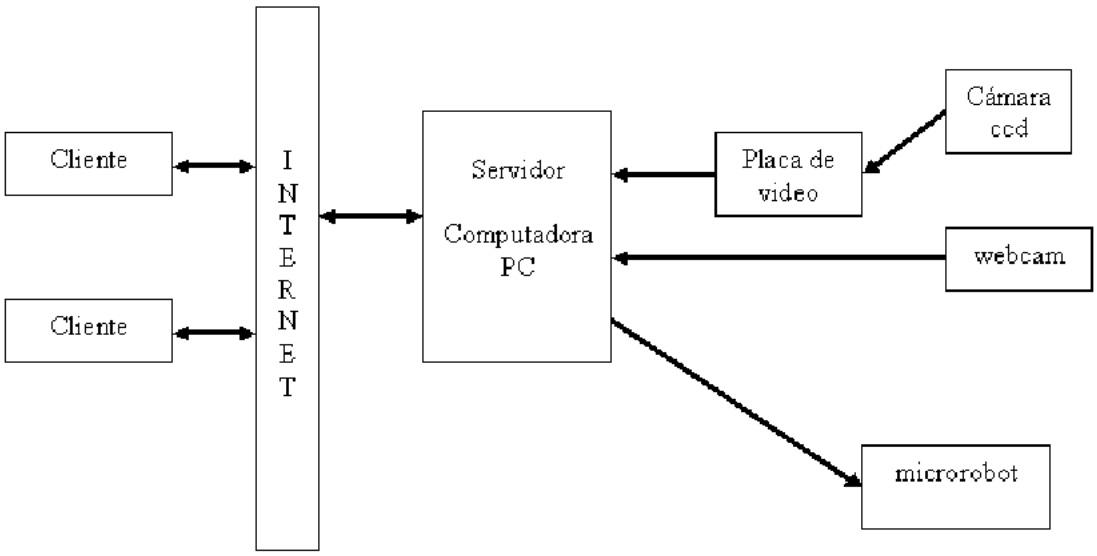


Figura 2.8 - Telerobótica

La arquitectura se implementa a través de una red basada en protocolos TCP/IP. El servidor está constituido por una PC, que dispone de una tarjeta de procesamiento de imágenes conectada a una cámara CCD o una webcam usb, para la visualización de los desplazamientos del microrobot. La conexión al sistema de experimentación (robot) se hace a través del puerto serie RS-232, puerto paralelo, de tarjetas de adquisición de datos o en forma inalámbrica. En cuanto al software se emplean interfases gráficas que se desarrollan con herramientas como controles *ActiveX* (limitados al empleo de navegadores de Microsoft y a plataformas Windows), *applets* Java (cualquier tipo de navegador con soporte Java), VRML (navegadores con intérprete de VRML) para el desarrollo de interfaces gráficas tridimensionales, y lenguajes de *scripts* como JavaScript o Jscript cuando se realiza la comunicación con el sistema remoto por medio de formularios y CGIs.

Una de las opciones más sencillas consideradas es utilizar mecanismos para dotar de capacidad de proceso tanto al servidor como al cliente (CGIs, JavaScript, etc.). Estos mecanismos permiten acceder a información almacenada en bases de datos, de modo que el usuario define los criterios de selección, que son enviados al servidor utilizando el protocolo HTTP; los resultados de la consulta son convenientemente formateados en HTML y presentados en el navegador del cliente.

Al conectarse a la página correspondiente el cliente podrá rellenar un formulario con los parámetros seleccionados y visualizar los resultados reales en imágenes sucesivas obtenidas del robot. Al enviar el formulario, se ejecuta en el servidor una función CGI que toma como argumentos los parámetros rellenados en el formulario, ejecuta la aplicación en modo local en el servidor y obtiene el camino a seguir por el robot. A continuación ejecuta localmente un archivo por lotes que carga en la memoria del robot el algoritmo del camino a seguir, poniendo finalmente en funcionamiento el robot.

En general el único medio de realimentación que dispone el cliente para analizar el resultado de la implementación del algoritmo es la visualización mediante algún sistema de video del comportamiento del robot, mientras que el supervisor del sistema puede comprobar las acciones llevadas a cabo por el cliente por la grabación en vídeo de toda o parte de la experiencia.

2.6 Tecnologías emergentes

Finalmente, tecnologías emergentes como la de Grid Computing también aportan a la implementación de laboratorios remotos. Se están desarrollando entornos distribuidos que consisten en un servidor virtual principal, que es el encargado de administrar los accesos, la validación de usuarios y la búsqueda de recursos disponibles. Existen uno o más servidores de laboratorios reales, que prestan los servicios de acceso a los recursos físicos. Finalmente están las

estaciones de los usuarios clientes. Todos estos componentes unidos por Internet, como se observa en la Figura 2.9.

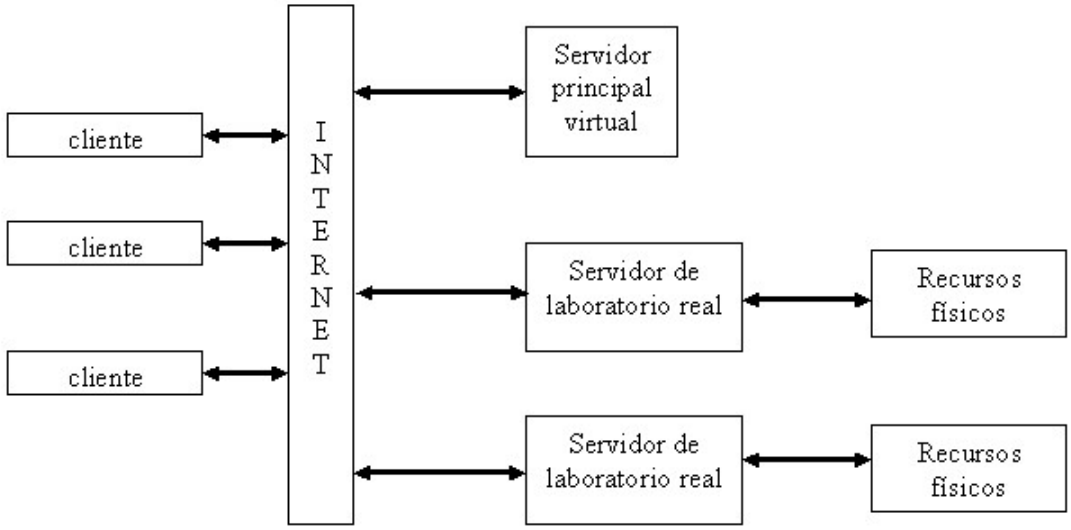


Figura 2.9 - Aproximación Grid Computing

Las principales características de estos modelos son: gran área geográfica para los experimentos que se puedan concretar, administración de recursos heterogéneos, sin límites para la localización de los laboratorios y sólo se necesita acceso a Internet para la comunicación entre los clientes y los servidores de los recursos. Los usuarios no requieren recursos de hardware y software particulares. Se trabaja en modo transparente con la locación real del dispositivo bajo control en un modo multiusuario concurrente. El entorno permite dos tipos de usuarios. Uno con privilegios que puede permiten agregar controles , modificar parámetros o instrumentos y otro en el cual sólo se pueden ver resultados en su pantalla para condiciones determinadas.

La coherencia de los resultados de cada sesión de usuario la garantiza el servidor local real ya que le asigna a cada usuario un espacio de datos diferente, de modo que visualiza los resultados de sus propios comandos.

Además, es escalable, ya que se pueden ir agregando servidores con más recursos iguales o diferentes, o aumentar las experiencias en número o en

complejidad. Para mejorar la respuesta del sistema se controla el número de usuarios conectados al sistema.

Representado un diagrama de capas de este ejemplo de laboratorio remoto junto con un diagrama de capas del paradigma de Grid Computing, se observan similitudes significativas, tal como se puede observar en la Tabla 2.1

Modelo de Grid Computing		Modelo de laboratorio remoto
Aplicaciones de usuario	Aplicaciones que controlan los componentes de las capas inferiores	Interfases gráficas de usuario
Servicios colectivos	Protocolos y servicios que implementan la interacción a través de una colección de recursos	Servidor de laboratorio virtual
Recursos y protocolos de conectividad	Protocolos para la autentificación y comunicación requeridas para las transacciones en Grid	Protocolos específicos basados en TCP
Fabric	Dispositivos físicos y recursos a los que los usuarios de Grid pueden acceder (computadoras sistemas de almacenamiento, redes, sensores, etc)	Servidores de laboratorios reales Recursos físicos de laboratorio

Tabla 2.1. Modelos de Grid y laboratorios remotos

Este modelo, propuesto por E. Bagnasco y A. Scapolla[15], presenta una visión similar a un esquema de grid computing, donde el servidor virtual principal actúa como un superoganizador. Un usuario descubre los recursos disponibles mediante este servidor virtual principal que tiene la responsabilidad de autorizar, filtrar y seleccionar el sistema donde el trabajo debe ejecutarse. Si más de un servidor de laboratorio real tiene el mismo experimento, el servidor virtual principal debe considerar la calidad de servicio de cada servidor de laboratorio real, en términos de número de usuarios y de ancho de banda. Una vez que se eligió el servidor de laboratorio real, el usuario comienza su tarea. Para mejorar la performance y la confiabilidad el servidor virtual principal, se recurre a técnicas de redundancia.

2.7 Requisitos de los laboratorios remotos

La implementación exitosa de estos entornos y su reemplazo por entornos locales de experiencias de laboratorio debe cumplir con los siguientes requisitos:

- Facilidad en la operación y en la compresión. Deben presentar una interfase amigable y el estudiante debe avanzar en la resolución de problemas sin ayudas externas al entorno.
- Adaptación de los materiales didácticos tradicionales al nuevo contexto. Si bien el entorno debe permitir la autogestión en el proceso de aprendizaje, también debe presentar la posibilidad de comunicación con tutores, compartir experiencias con otros estudiantes o consultar material de lectura.
- Supervisión en línea del entorno de laboratorio. Los usuarios deben conocer en cada momento que está sucediendo con el sistema físico, por ello tienen que recibir información sobre el estado del mismo.
- Desarrollo de políticas de seguridad del sistema físico y del servidor. Los usuarios no deben tener acceso directo a recursos privados del sistema (datos personales, archivos del sistema, red local, hardware de bajo nivel). El software del servidor debe proteger al sistema de acciones hostiles o accidentales que puedan dañarlo. Se debe encontrar un punto de equilibrio entre seguridad y flexibilidad.
- Empleo de software cliente multiplataforma. El entorno debe poder utilizarse desde distintos sistemas operativos (Windows, Linux, MacOs).
- Empleo de software cliente de fácil instalación. El usuario debe operar una computadora, pero no tiene que ser un experto. No debe requerirse terceras partes ni de paquetes complejos de instalación.

- Seguridad del cliente: el cliente debe estar protegido de descargas y ejecución de nuevo software en su computadora.
- Software cliente libre. El cliente sólo necesita una computadora con acceso a Internet y ningún software adicional.
- Acceso a librerías de procesos o actividades previas. Es una buena opción dotar al sistema de la posibilidad de disponer de los resultados de experiencias realizadas por otros usuarios, generando, un ambiente colaborativo.
- Objetos persistentes. El sistema debe almacenar los trabajos desarrollados por los usuarios de modo persistente. Esto es necesario para que cada usuario pueda conocer su evolución y también para que el docente tenga acceso a las tareas de los alumnos.
- Políticas de mantenimiento y versiones. La posibilidad de introducir mejoras y modificaciones en el entorno deben ser previstas.
- Arquitectura modular y abierta. Se deben poder incluir nuevos componentes y ejercicios con el mínimo esfuerzo y la menor discontinuidad del servicio.
- Percepción sobre el sistema físico. Se debe garantizar la percepción de los efectos que se realizan sobre el sistema físico.
- Aceptables parámetros de calidad de servicio. Una parámetro indicador es la velocidad de respuesta en la recepción.

Dependiendo de la clase de laboratorio y de los objetivos perseguidos, algunos de estos requisitos pueden omitirse. Esta lista enumera las características que deberían considerarse como paso preliminar al diseño de un laboratorio remoto.

En definitiva, un laboratorio remoto es “un sistema que permite el acceso a elementos de prácticas de laboratorio utilizando recursos de Internet con un suficiente grado de presencia para desarrollar actividades prácticas en la

misma manera que si se realizaran en modo tradicional en un laboratorio presencial". [16]

2.8 Arquitectura de los laboratorios remotos

La realización de prácticas de laboratorio en modo remoto implica resolver distintas cuestiones técnicas, como por ejemplo la seguridad en el acceso a los recursos, la reinicialización y configuración del equipamiento al comenzar cada nuevo ejercicio o práctica, la reserva de ejercicios y recursos en una agenda de laboratorio o permitir que los usuarios puedan realizar varias prácticas en modo simultaneo.

Además, la metodología para desarrollar las prácticas se puede resumir en un proceso donde los alumnos acceden al laboratorio utilizando un web broser (navegador), pasan por un proceso de identificación (login y password), llegan a una página web de ejercicios propuestos, seleccionan uno de ellos y finalmente acceden a la página del ejercicio elegido. Esta página debe incluir los objetivos, actividades, vínculos al sistema físico real y a material de apoyo relacionado con la temática que trata la ejercitación propuesta.

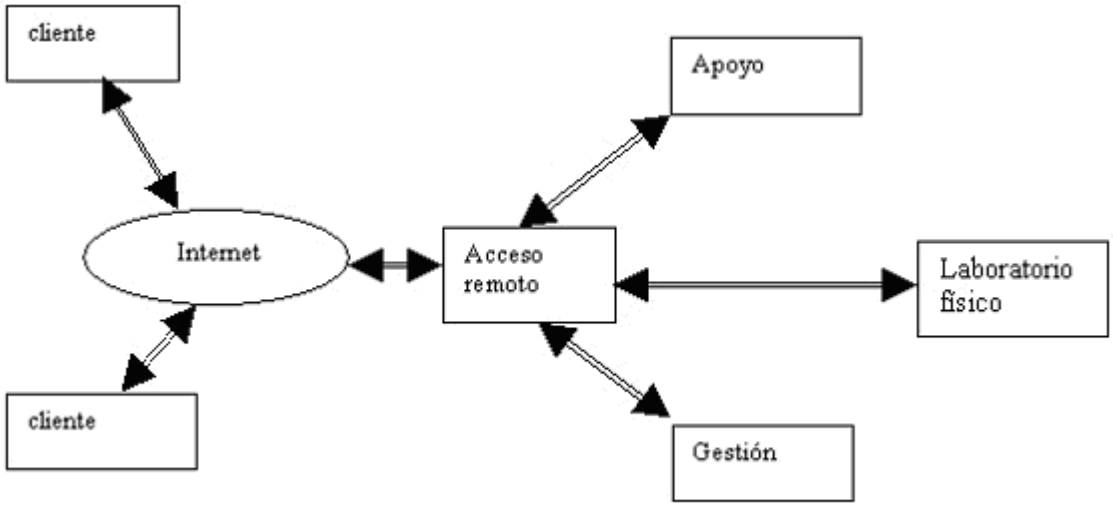


Figura 2.10 - Arquitectura básica de un laboratorio remoto

La figura 2.10 representa una aproximación a la arquitectura de los laboratorios remotos. En ella se representan los requerimientos de software como de hardware de la misma.

En resumen, la arquitectura de un laboratorio remoto requiere:

a) Herramientas de gestión:

Administrador de bases de datos para gestionar: datos de usuarios, reserva de recursos, archivos de experiencias o trabajos prácticos (configuración, resultados, informes, estadísticas).

b) Herramientas de apoyo o soporte:

Páginas dinámicas que permitan seleccionar módulos o experiencias y que posibiliten interactuar con los dispositivos físicos del laboratorio.

Material multimedia que ayude a la mediación de los contenidos desarrollados en el módulo o la experiencia.

c) Herramientas de acceso remoto:

Hardware y software que permitan el acceso remoto interactivo a las herramientas y dispositivos del laboratorio físico.

d) Hardware de control de los dispositivos físicos del laboratorio. [17]

2.9 Componentes de los laboratorios remotos

En general un laboratorio remoto debería contar con cuatro módulos que se detallan a continuación:

a. Sistema físico real o laboratorio físico

Son los componentes sobre los cuales se debe realizar la práctica, incluyendo los elementos necesarios para su control y monitoreo. Es importante la flexibilidad de estos elementos para posibilitar diferentes configuraciones que abarquen un número de ejercicios adecuados para los contenidos que se quieren alcanzar.

En el caso de disponer de recursos duplicados, es necesario el soporte físico para la distribución de los procesos que requieren los distintos usuarios y/o diferentes ejercicios, ya que se tienen que realizar simultáneamente sin

interferencia entre los mismos. Por otra parte las diferentes prácticas necesitan distintas configuraciones. Por ello hace falta un componente que permita disponer de las configuraciones correspondientes a cada ejercicio. Las mismas pueden ser archivos de plantillas o archivos de solo lectura. Estos archivos se buscan y transfieren desde un directorio de archivos mediante algún protocolo (por ejemplo TFTP) o encontrarse embebidos en algún hardware que cuente con este tipo de soporte.

b. Acceso remoto.

Este bloque se puede describir en dos partes, una es la de la arquitectura cliente-servidor y la otra, es la del sistema de seguridad.

El usuario cliente mediante un programa navegador carga la página correspondiente. El navegador solicita la página al servidor. Se introducen los datos en la página y se envían, los cuales son procesados como un formulario en el servidor.

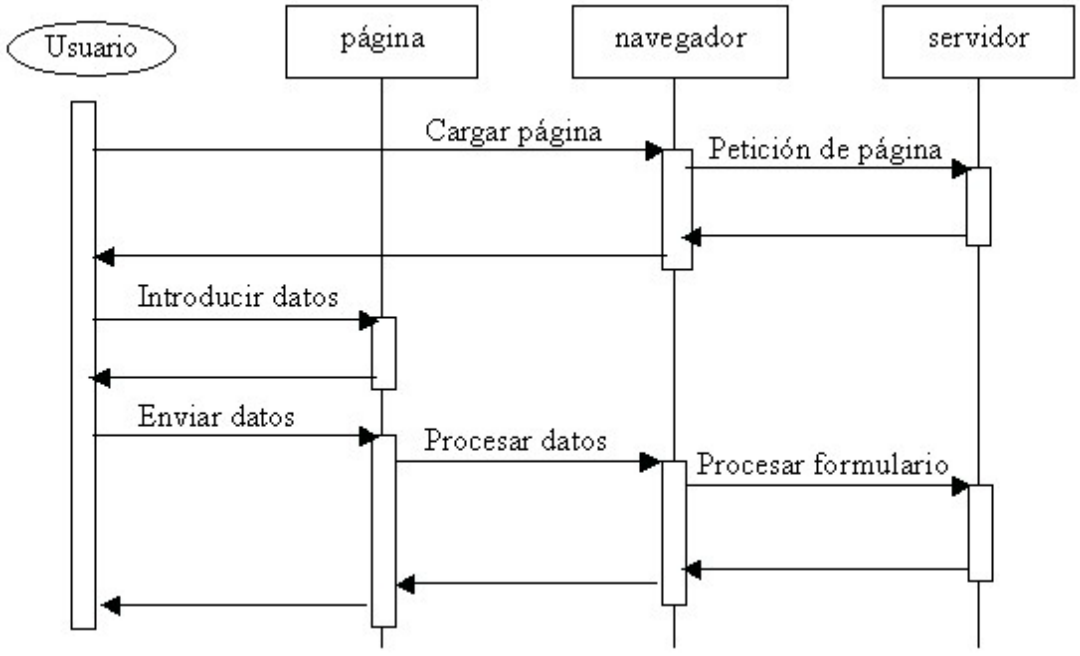


Figura 2.11 – Diagrama de ejecución desde el cliente

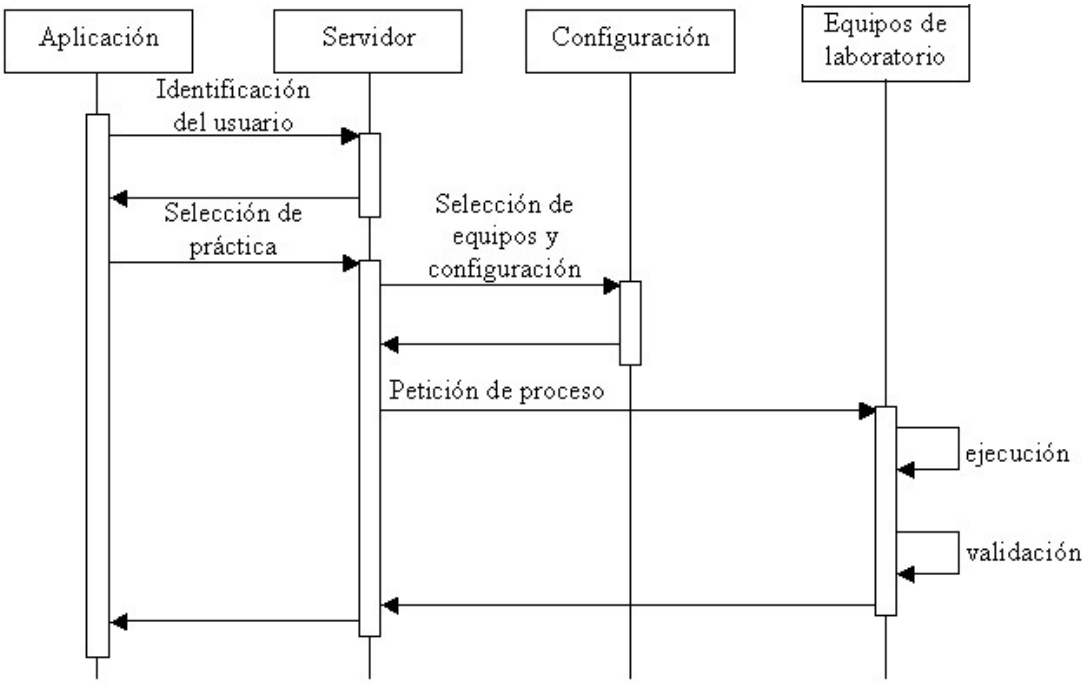


Figura 2.12 – Diagrama de ejecución desde el servidor

El diagrama de ejecución de la aplicación desde el cliente hasta el servidor se presenta en la figura 2.11, donde el canal de comunicación es Internet.

La aplicación que se ejecuta en el servidor permite la identificación y verificación del usuario, la configuración de los equipos de laboratorio en función del ejercicio que se desea realizar y finalmente la ejecución y validación de la práctica.

La figura 2.12 muestra el diagrama de ejecución desde el servidor al equipamiento de laboratorio, que se realiza en forma local utilizando la infraestructura de conexión del sistema físico real.

En cuanto a la implementación del mecanismo de seguridad de acceso se deben considerar diferentes aspectos. Este mecanismo tiene como propósito fundamental controlar el tipo de acceso y por otro lado debe prevenir el mal uso de los recursos. El control de acceso se puede llevar a cabo mediante la identificación de un usuario por login y password. Por otra parte se tiene que

garantizar que el usuario validado realizará prácticas sin dañar el equipamiento físico del laboratorio, para ello se deben desarrollar acciones que eviten estos inconvenientes.

c. Gestión de laboratorio.

Este módulo puede contar con dos subsistemas que interactúan: un subsistema de gestión de accesos y un subsistema de gestión de configuraciones. Estos subsistemas utilizan información de gestión que se almacena en una base de datos relacional. Cada subsistema se ocupa de un aspecto diferente de la gestión del laboratorio.

El subsistema de gestión de acceso cumple las funciones de controlar el acceso de los usuarios a los recursos y la agenda que permite planear su utilización, es decir qué recursos del laboratorio puede usar y durante cuanto tiempo.

El subsistema de control de configuraciones se ocupa de la reinicialización y configuración del equipamiento físico del laboratorio en función de la práctica que se desea realizar. Una aproximación a la interacción entre estos módulos se representa la figura 2.13.

El servidor se encarga de la comunicación con los clientes a través de Internet e interactúa con los componentes de gestión del laboratorio. En este servidor se ejecuta una aplicación que incluye un conjunto de módulos, que se pueden implementar por ejemplo en PHP, que procesan los pedidos de los usuarios. Los accesos de los usuarios se producen por vínculos o links HTML hacia el servidor. A su vez esta aplicación accede a los datos almacenados en la base de datos. El código PHP es embebido en páginas HTML y procesado en el servidor, siendo la aplicación transparente a los usuarios. Esta aplicación debe permitir tres niveles de acceso: profesor, alumno y administrador. Para ello interactúa con el mecanismo de control de acceso, que además de la identificación de los usuarios, le permite al administrador o al profesor crear y mantener cuentas de los usuarios y definir los tiempos de utilización para cada

usuario. Para ello interactúa con una agenda que planifica las sesiones (tipo de ejercicio y tiempo para cada usuario). La aplicación también se vincula con el sistema de seguridad para controlar el tipo de configuración según el tipo de acceso.

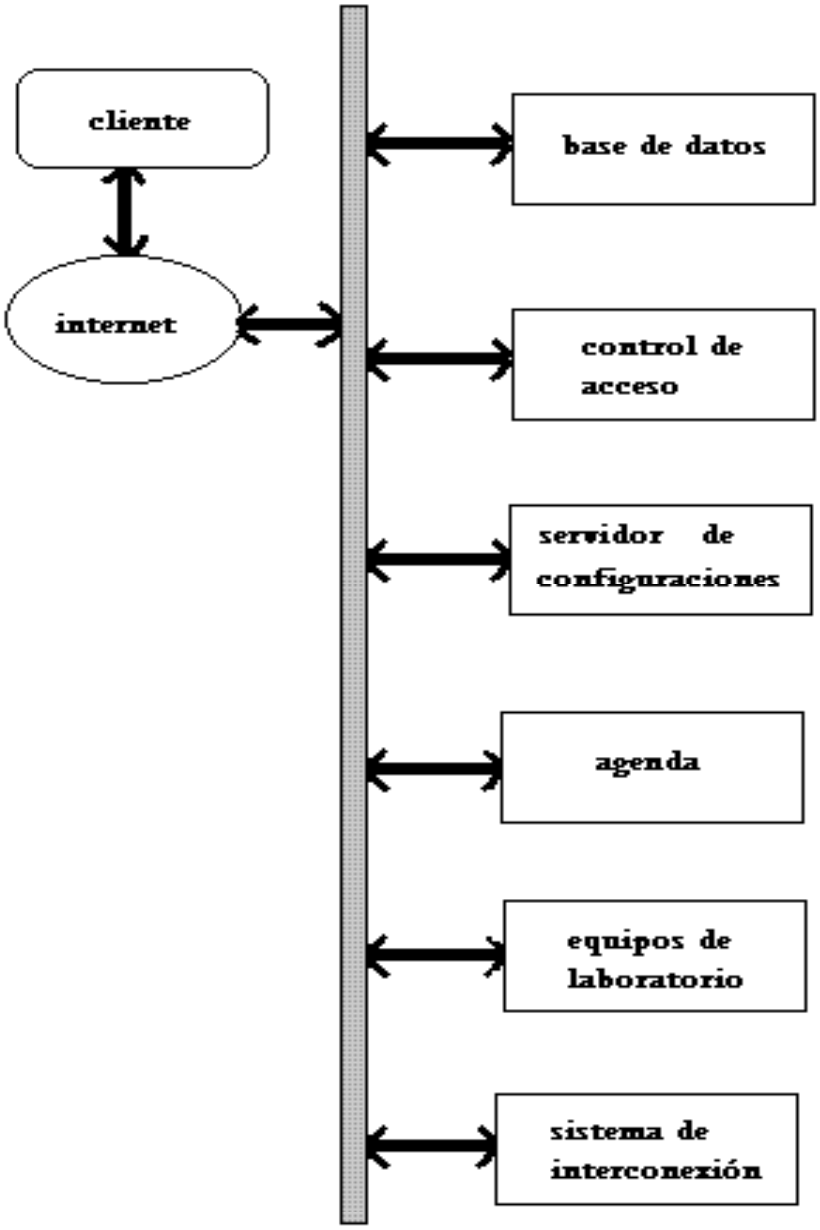


Figura 2.13 – Módulos de configuración

La agenda es un servicio que dirige la utilización del laboratorio y efectúa la reserva de los recursos. Este servicio usa la base de datos, la cual mantiene información de usuarios, ejercicios, recursos, configuraciones y otra información relevante del sistema. [6]

Antes de iniciar cada ejercicio se debe reinicializar y configurar el equipamiento del laboratorio según la práctica que se va a realizar. Esto implica que una vez autorizado el usuario, seleccionado el ejercicio y asignado el equipo, el laboratorio debe ser configurado. Por ello se requiere un servidor de configuraciones y un sistema de interconexión que materialice cada configuración solicitada. Esta característica de reconfiguración hace a la flexibilidad del laboratorio remoto.

d. Ambiente de apoyo multimedial.

Uno de los incovenientes de los laboratorios remotos es la posible falta de percepción de los resultados de las acciones ejecutadas sobre los recursos físicos reales. Por ello es necesario el monitoreo de los mismos y su visualización por parte de los usuarios. Por ello hace falta un ambiente multimedia que permita satisfacer esta necesidad.

Además este software también debe incluir material de apoyo que abarque los contenidos que se quieren desarrollar, ya sea en forma teórica o práctica, incluso con simulaciones.

Todas estas necesidades son las que hacen al entorno del laboratorio remoto un sistema complejo en su implementación. Dicha complejidad implica un análisis previo que justifique porqué utilizar esta manera de realizar experiencias de laboratorio, el contexto en el cual se va a utilizar y la elección de las tecnologías y herramientas de diseño, montaje y puesta en marcha del sistema en cuestión.

2.10 Conclusión

En base a la caracterización desarrollada de los laboratorios remotos y los ejemplos descriptos en este capítulo se puede concluir que las TICs e Internet ayudan a mejorar la apropiación de conocimientos en disciplinadas que no estén específicamente relacionadas con sus propios contenidos, promueven tipos de cursos que se adecuan a distintos estilos de aprendizaje, hacen el aprendizaje accesible a estudiantes no tradicionales o con capacidades distintas y permiten disminuir los costos de la educación, manteniendo la calidad. De este modo se contestan afirmativamente aquellas preguntas que se plantearon en la introducción.

CAPÍTULO 3
DISEÑO E IMPLEMENTACIÓN DE LABORATORIOS REMOTOS: MODELO CONCEPTUAL

En este capítulo se sugiere una clasificación para los entornos de prácticas de laboratorio que permite contextualizar los distintos modos de acceder a los recursos reales o físicos de un laboratorio.

Además se propone un modelo conceptual que sea de utilidad para el diseño, la implementación y el análisis de laboratorios remotos y de los componentes que los constituyen, las funciones que desarrollan dichos componentes y el modo en que interactúan. También se presenta un ejemplo básico de diseño.

3.1 Acceso a los recursos reales

Si se hace foco en ambientes de instituciones educativas y en el acceso a recursos reales, los entornos para experiencias de laboratorio se pueden clasificar en función de los actores que intervienen en las mismas: el alumno que realiza la experiencia y los recursos reales que requiere llevar a cabo la misma. Estos recursos pueden ser físicos (herramientas, instrumentos, textos, hardware y software, guías, etc.), como así también humanos (docente que cumple las funciones de guiar, supervisar y/o evaluar).[18] A partir de esta consideración los entornos son:

- Entorno local monousuario
- Entorno local multiusuario
- Entorno local multiusuario con recursos múltiples
- Entorno remoto monousuario
- Entorno remoto multiusuario
- Entorno remoto multiusuario con recursos distribuidos

En cualquier tipo de práctica presencial, los recursos del laboratorio están a disposición del alumno. Se trata de una situación ideal para experiencias de

aprendizaje, ya que el acceso a los recursos es total. Llamamos a estas condiciones “entorno local monousuario”, donde un alumno (usuario) accede localmente a los recursos de un laboratorio local (recursos reales físicos y profesor). Este entorno se muestra en la Figura 3.1.

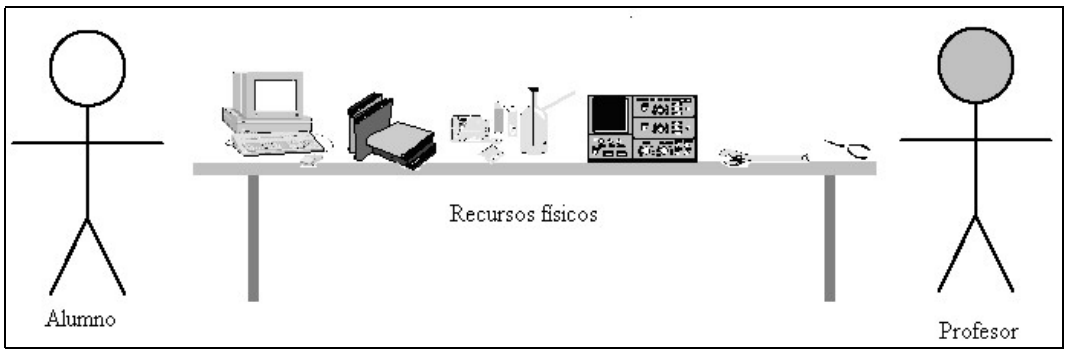


Figura 3.1 - Entorno local monousuario

Este entorno no es el que se presenta generalmente, ya que comúnmente los recursos del laboratorio local son compartidos por un determinado número de alumnos. Se pasa a un “entorno local multiusuario” donde el acceso se torna limitado. Además a las funciones del docente se suma la de administrar los recursos físicos del laboratorio, como se observa en la Figura 3.2.

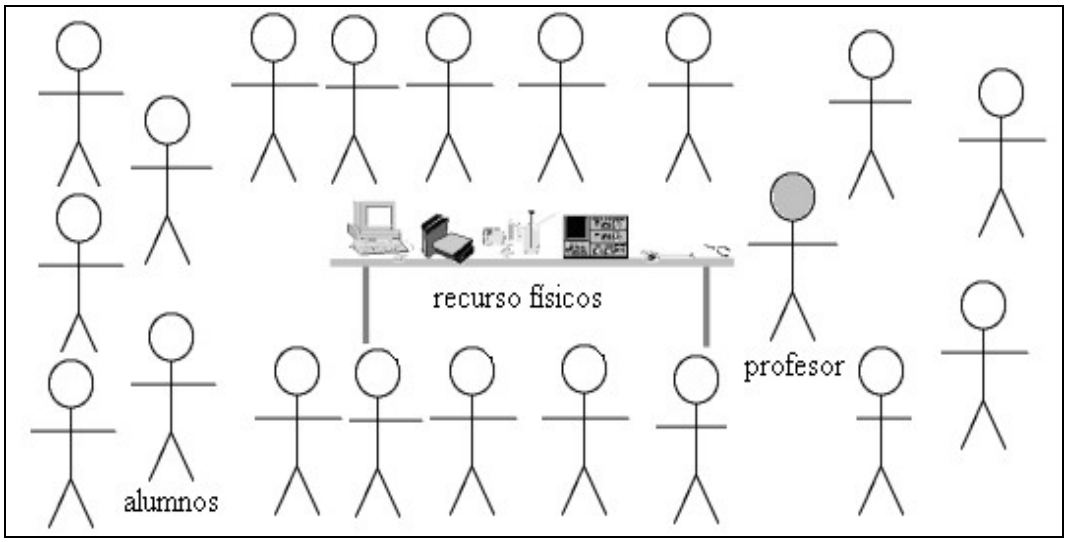


Figura 3.2 - Entorno local multiusuario

Este entorno no es el más adecuado para la realización de prácticas presenciales debido a las limitaciones que se generan en el acceso a los recursos del laboratorio local. Para mejorar la situación anterior, una alternativa es aumentar cuantitativamente la cantidad de recursos, de modo de facilitar el acceso a los recursos locales compartidos, configurando un “entorno local multiusuario con recursos múltiples”, el cual se presenta en la Figura 3.3.

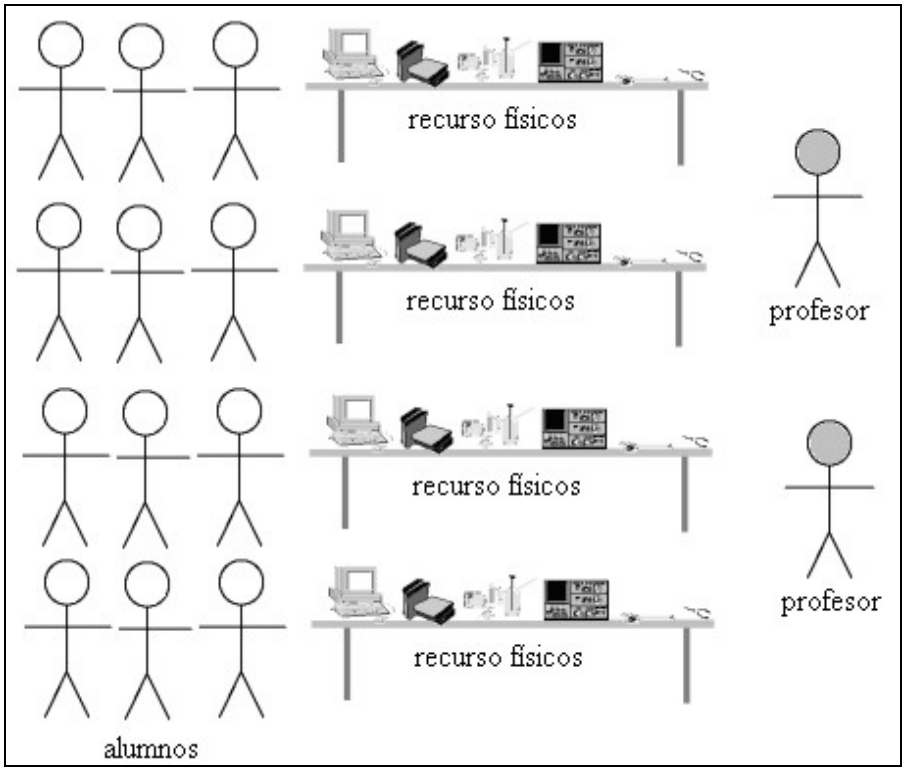


Figura 3.3 - Entorno local multiusuario con recursos múltiples

Otra alternativa es que el acceso a los recursos no sea solamente local sino que se materialice en forma remota. La tecnología teleinformática permite este tipo de entorno, el cual requiere un sistema de acceso y un sistema de administración y control de los recursos reales físicos.

Ambos sistemas configuran un laboratorio remoto componiendo un “entorno remoto monousuario”. Si las condiciones que presenta un laboratorio remoto son tales que los recursos se pueden compartir y administrar de manera que cada alumno concrete su experiencia personal sin necesidad de replicar los

recursos se obtiene un “entorno remoto multiusuario”, como se ve en la Figura 3.4.

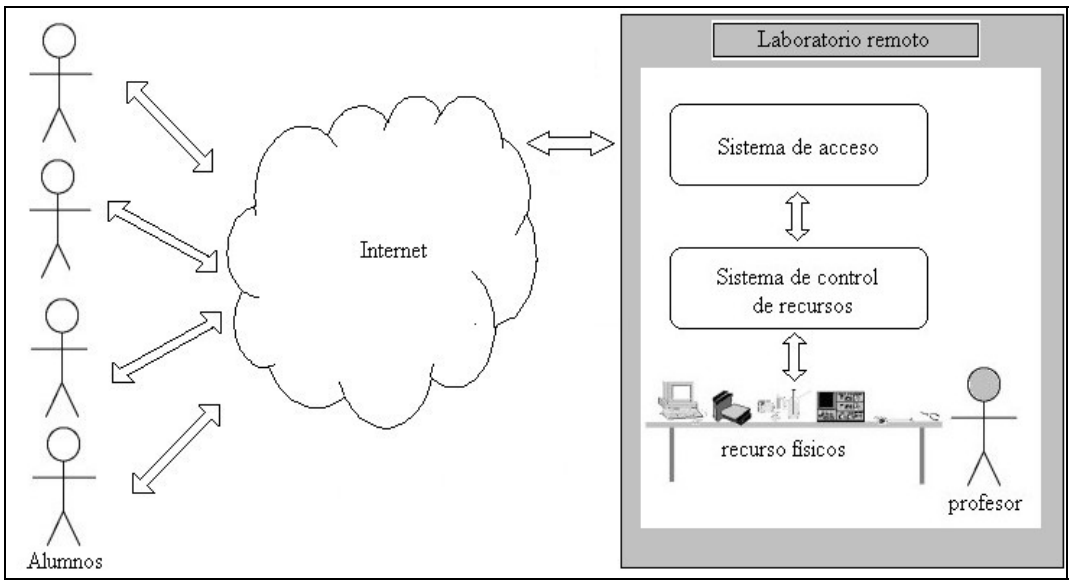


Figura 3.4 - Entorno remoto multiusuario

Finalmente otra variante a estos entornos es considerar la existencia de laboratorios remotos ubicados en otras sedes de la misma institución u otras

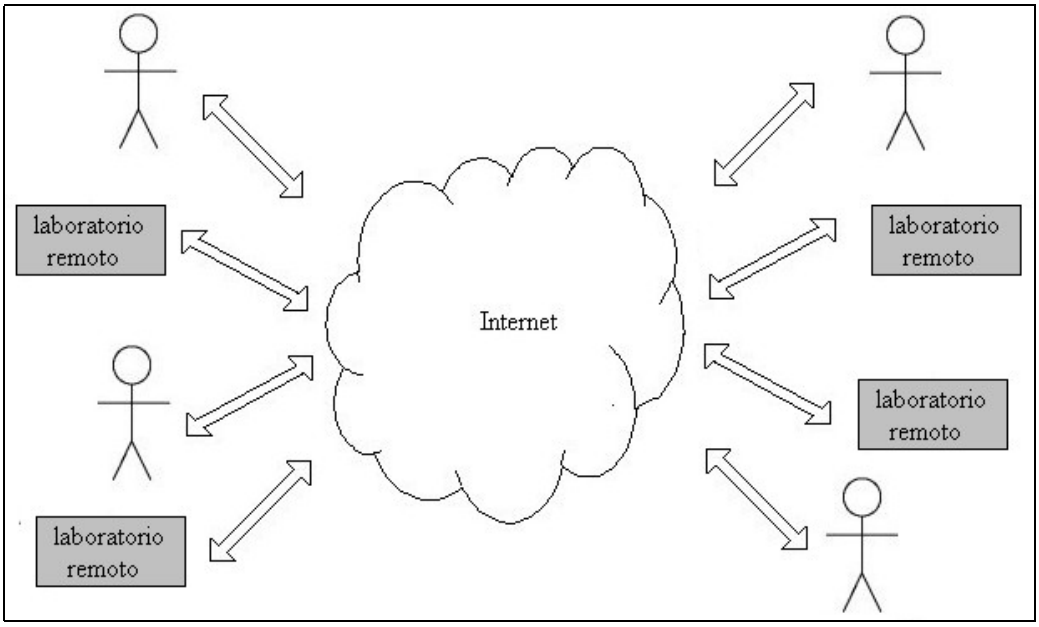


Figura 3.5 - Entorno remoto multiusuario con recursos distribuidos

organizaciones, conformando una “malla de laboratorios” de manera que no solamente se multiplican los usuarios, sino también los recursos, componiendo un “entorno remoto multiusuario con recursos distribuidos”. Dicho entorno se presenta en la Figura 3.5.

Como se puede apreciar, este entorno presenta características comunes a las descriptas en el Capítulo 2 en el apartado 2.6 referente a la aplicación de tecnologías como las de Grid Computing. Aquí se plantea la posibilidad de buscar recursos distribuidos globalmente.

3.2 Estructura de capas

Para el diseño de entornos remotos como los descriptos se puede decir que una arquitectura del tipo cliente – servidor es adecuada. Hemos expresado que la metodología para desarrollar las prácticas se puede resumir en un proceso en el cual los alumnos acceden al laboratorio utilizando un navegador, pasan por un proceso de identificación, acceden a una página web de ejercicios propuestos, seleccionan uno de ellos y finalmente acceden a la página del ejercicio elegido. Esta página debe incluir los objetivos, actividades, vínculos al sistema físico real y al material de apoyo relacionado con la temática que trata la ejercitación propuesta. También hemos afirmado que la realización de prácticas de laboratorio en modo remoto implica resolver distintas cuestiones técnicas, como por ejemplo la seguridad en el acceso a los recursos, la reinicialización y configuración del equipamiento al comenzar cada nuevo ejercicio o práctica, la reserva de ejercicios y recursos en una agenda de laboratorio o permitir que los usuarios puedan realizar varias prácticas en modo simultaneo. Recapitulando, los componentes que requiere entonces este tipo de entorno son:

- Un programa de navegación web en el cliente, desde el cual poder acceder en modo remoto a los recursos físicos mediante páginas dinámicas.

- Un sistema de control de acceso remoto. Este sistema cumpliría con dos funciones. Una es la de implementar mecanismos de seguridad que controlen el tipo de acceso, la identificación y validación de usuarios y prevenir contra usos indebidos de los recursos. La otra función es permitir la ejecución en modo remoto de los procesos permitidos para la práctica seleccionada por el usuario.
- Un sistema de gestión, que a su vez se divide en dos: un subsistema de gestión de accesos, que se ocupa de la operación de una agenda de reservas que permite planificar cuando y durante cuanto tiempo se pueden utilizar los recursos. El otro subsistema se encarga del control de configuraciones ya que antes de iniciar cada ejercicio se debe reinicializar y configurar el equipamiento del laboratorio según la práctica que se va a realizar. Esto implica que una vez autorizado el usuario, seleccionado el ejercicio y asignado el equipo, el laboratorio se debe configurar. Por ello hace falta un componente que permita disponer de las configuraciones correspondientes a cada ejercicio.
- Un Servidor de configuraciones y sistema de interconexión: Como al comenzar cada práctica los equipos del laboratorio físico se deben adecuar a la misma, se necesita un servidor que contenga la correspondiente configuración en base al tipo de ejercicio seleccionado por el usuario. Cada configuración puede ser una plantilla o archivo de solo lectura. Estos archivos se transfieren desde un directorio de archivos mediante algún protocolo o se encuentran embebidos en algún hardware que cuente con este tipo de soporte
- Un sistema de interconexión capaz de modificar la estructura de conexiones en función de la configuración solicitada. Esta característica de reconfiguración hace a la flexibilidad del laboratorio remoto.

- Herramientas de seguimiento. Es muy útil la posibilidad de poder llevar una traza de los procesos realizados por cada uno de los usuarios que acceden al laboratorio remoto.
- Ambiente de apoyo multimedia. Este software debe incluir material de apoyo que abarque los contenidos que se quieren desarrollar, ya sea en forma teórica o práctica, incluso con simulaciones.
- Base de datos. Este componente permite almacenar toda la información referente a la gestión del laboratorio (usuarios, accesos, reservas, y recursos) y prestar servicios de consulta a los demás componentes del sistema.
- Infraestructura de red y sistema físico real. Son los recursos físicos con los cuales se desea interactuar en modo remoto, incluyendo las interfases necesarias para su control y monitoreo, como así también los componentes de la red local del laboratorio. En la medida que las interfaces sean flexibles, existirá la posibilidad de diferentes configuraciones que abarquen un número de ejercicios apropiados para los contenidos que se quieren alcanzar.

Al citar los componentes que requiere un laboratorio remoto se pone de manifiesto la complejidad del sistema.

Para allanar el abordaje al estudio de este tipo de entorno, se propone a continuación un modelo de capas que permite considerar en modo integral los procesos y tareas realizadas tanto en el cliente como en el servidor, el cual se presenta en la Figura 3.6. Esta estructura de capas tiene como objetivo facilitar los procesos de análisis, diseño e implementación de laboratorios remotos. En la capa superior (o capa 6) se encuentran los procesos que lleva a cabo el cliente, es decir que esta capa debe permitir el acceso a los recursos físicos del laboratorio en modo remoto y además la obtención de los resultados de las experiencias que realice el cliente sobre dichos recursos.

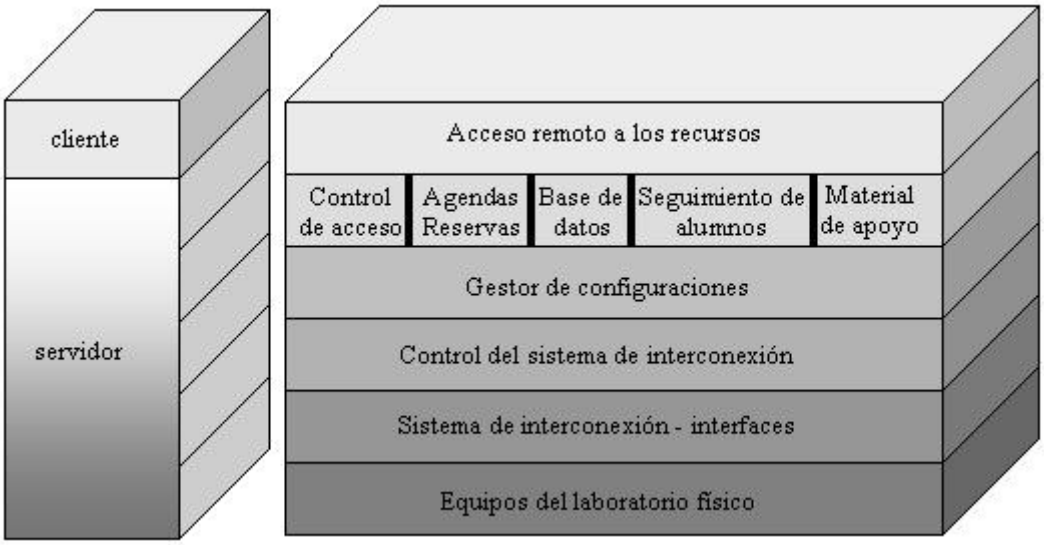


Figura 3.6 - Estructura de capas

En la siguiente capa (capa 5) se encuentran todas las funciones relacionadas con el control de acceso y la reserva de los equipos del laboratorio físico; como así también la base de datos con los distintos tipos de prácticas que puede seleccionar el cliente, los resultados de las experiencias realizadas por los alumnos para el seguimiento por parte del docente y además el material de apoyo para cada uno de los ejercicios que se puedan desarrollar mediante el acceso remoto. Por debajo de esta capa debe operar un gestor de configuraciones (capa 4), que se debe encargar buscar y encontrar la configuración del laboratorio remoto necesaria para la implementación de la práctica seleccionada por el cliente-alumno y también se debe ocupar de entregar a la capa inferior, que realiza el control del sistema de interconexión, (capa 3) los parámetros y variables requeridos en función de la configuración elegida. Una capa por debajo se encuentra el Sistema de interconexión y las interfaces de hardware (capa 2) que son las responsables de la configuración física que permite el acceso a los dispositivos con que se desea experimentar. Por último la capa inferior (capa 1) incluye los equipos y componentes físicos del laboratorio.

Este modelo de capas también es útil para representar algunas de las herramientas que pueden usarse para el diseño y puesta en funcionamiento de un laboratorio remoto, como se observa en la Figura 3.7.

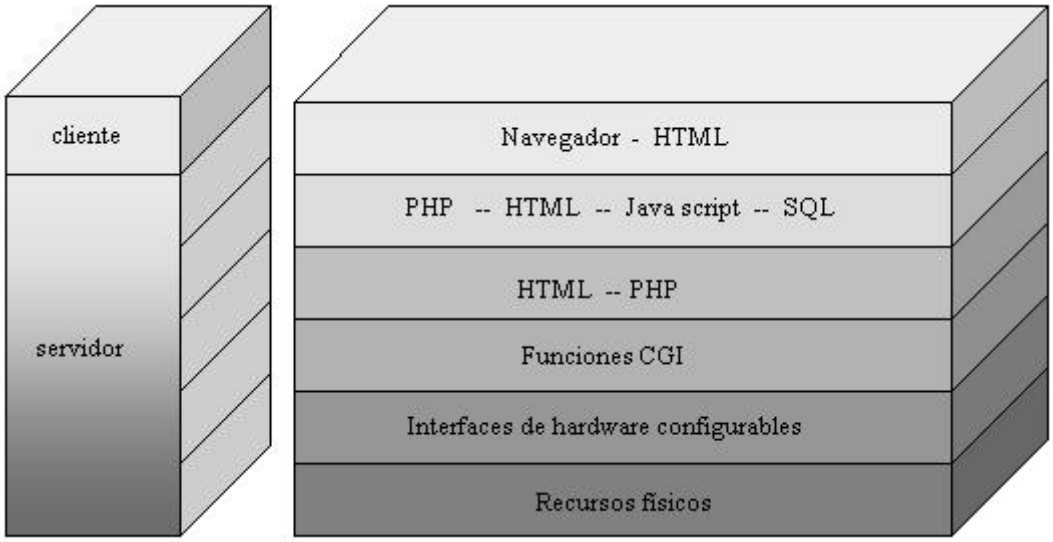


Figura 3.7 - Herramientas para el diseño y puesta en funcionamiento de un laboratorio remoto

En el capítulo 2 se han presentado algunos ejemplos de implementaciones de laboratorios remotos. La mayoría de ellos se relacionan con prácticas que tienen que ver con temáticas como las de redes de computadoras, circuitos eléctricos y electrónicos, sistemas digitales o sistemas de control. La naturaleza intrínseca de estos componentes y dispositivos permite que el desarrollo de experiencias de laboratorio mediante “teleoperación” puedan concretarse con mayor facilidad que en otros disciplinas. Esta afirmación se justifica mediante el siguiente ejemplo, el cual describe de que modo se pueden realizar prácticas remotas sobre circuitos eléctricos básicos.

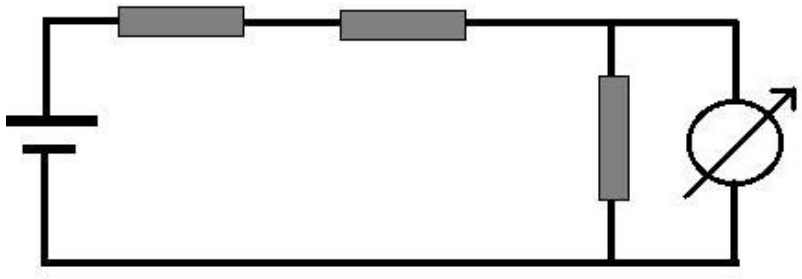
3.3 Ejemplo básico: Simulación de “Prácticas remotas sobre circuitos eléctricos sencillos”

El siguiente ejemplo es una simulación del diseño de un laboratorio remoto para prácticas sobre circuitos eléctricos sencillos. Al finalizar con la descripción se verificará si la simulación se adapta al modelo de estructura de capas.

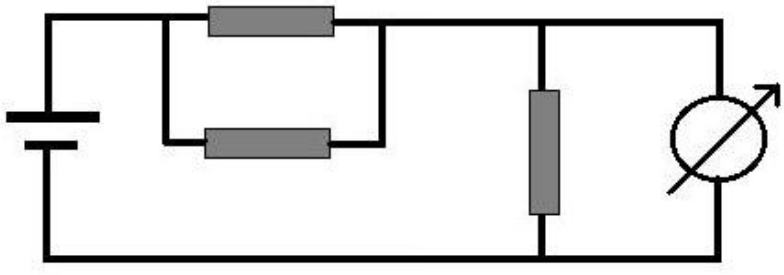
El punto de partida son de los recursos físicos.

a) Laboratorio físico:

Los recursos físicos para las prácticas son: resistencias eléctricas, voltímetro y fuente de alimentación. Las prácticas que se realizan tienen que ver con el comportamiento de los circuitos eléctricos serie y paralelo. La figura 3.8 presenta el circuito serie y el circuito paralelo que se quiere ensayar.



circuito serie



circuito paralelo

Figura 3.8 - Circuitos eléctricos básicos

Localmente la práctica consiste en armar las dos configuraciones y registrar o leer las indicaciones del voltímetro. Conociendo los valores de la tensión de

alimentación, las resistencias y la lectura, aplicando la ley de Ohm, se pueden obtener conclusiones respecto del funcionamiento de los circuitos serie y paralelo. Para llevar este ámbito a un entorno remoto, una posibilidad es la que se presenta en la figura 3.9

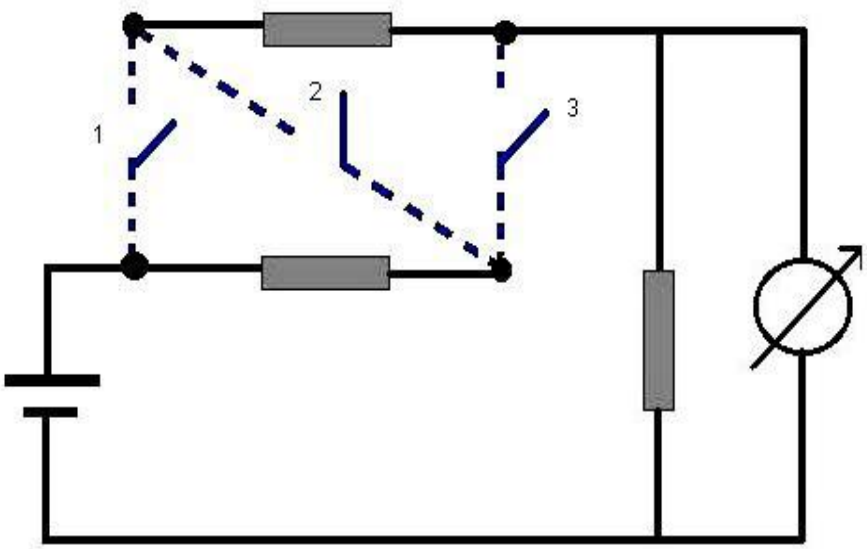


Figura 3.9 - Circuito configurable.

Los interruptores 1, 2 y 3 permiten la configuración del circuito en serie o en paralelo. Mientras los interruptores 1 y 3 permanecen abiertos y 2 se encuentre cerrado se está en presencia de un circuito serie. Si 1 y 3 se cierran y 2 se abre se obtiene un circuito paralelo. Mediante la lectura de la tensión de salida en el voltímetro conectado en paralelo con la resistencia de carga se puede analizar el comportamiento de los dos tipos de circuitos. Estos conmutadores pueden ser los contactos de reles. El control de los mismos se puede implementar de modo automático mediante algún sistema digital, como por ejemplo los bits de salida de un registro puerto de un microcontrolador, los que se conectan a la base de un transistor que excita la bobina de los reles, como se observa en la figura 3.10.

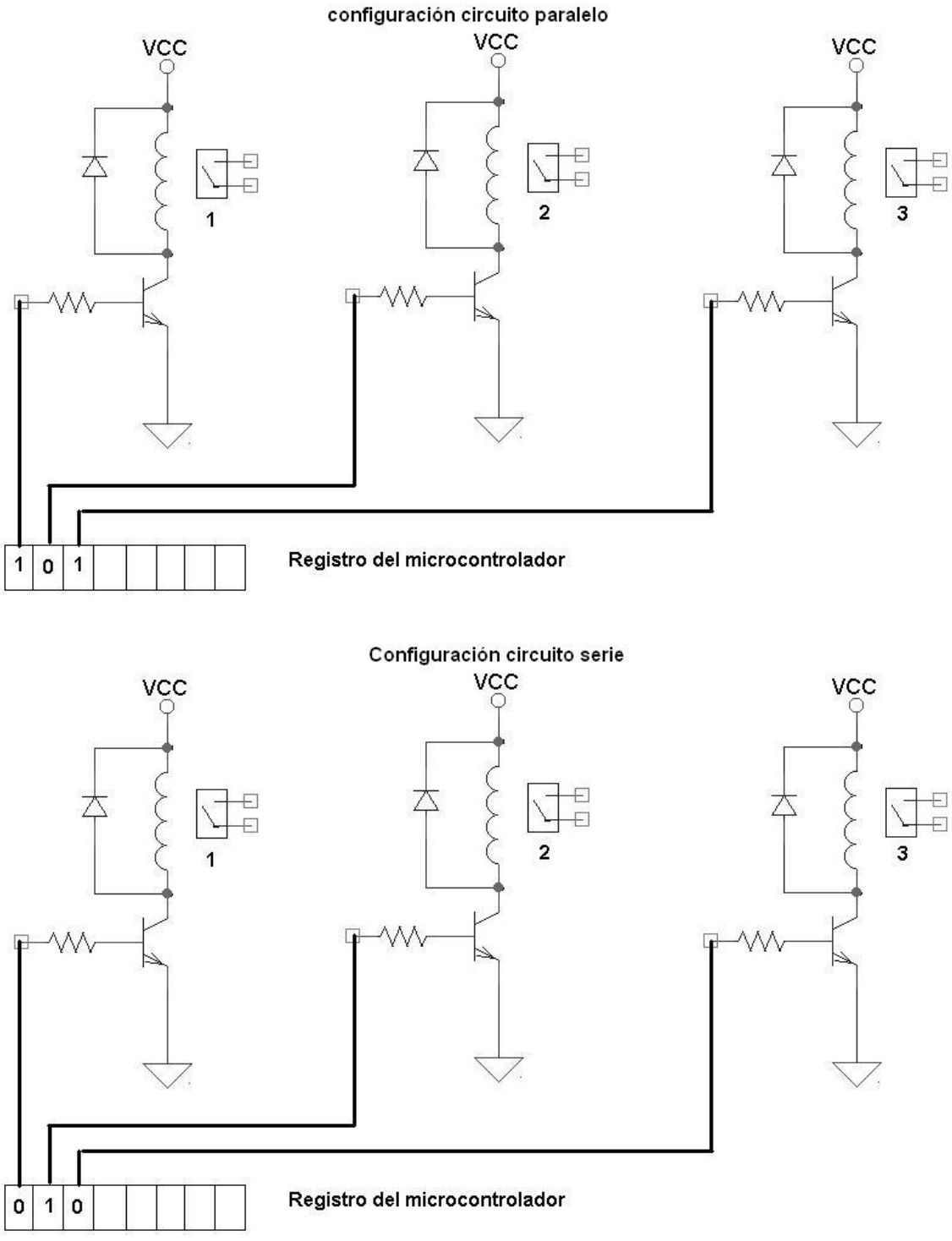


Figura 3.10 - Control de configuración de circuitos básicos

El dato escrito en el registro (puerto) del microcontrolador configura la conexión paralelo (dato =101) o serie (dato = 010) del circuito físico. El dato depende del código que se graba en la memoria flash del microcontrolador, como se ve en la Figura 3.11. La configuración de la práctica depende del código. Si los diferentes códigos se pueden elegir, se podrán configurar diferentes ejercicios.

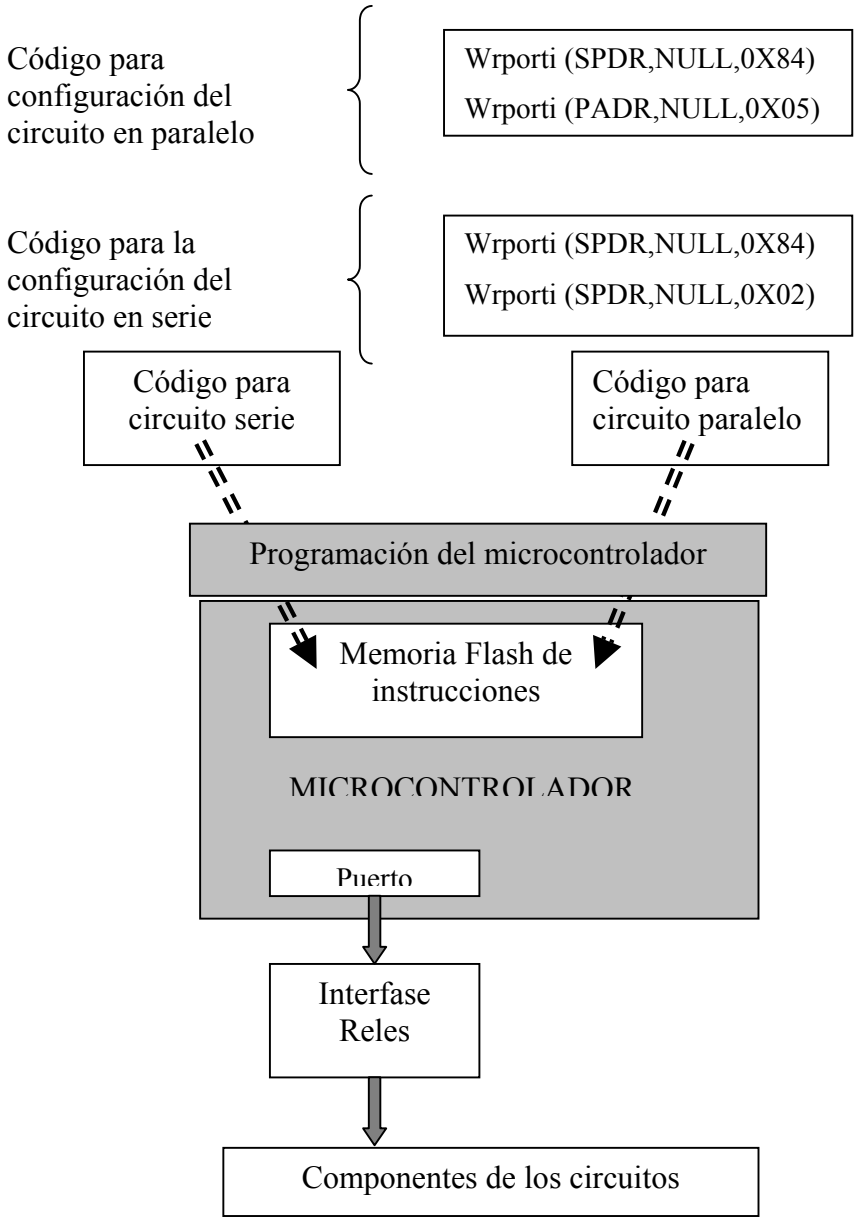


Figura 3.11 - Programación de la configuración

Si el microcontrolador tiene características de control embebido, los códigos se pueden embeber en el mismo y entonces el alumno podrá seleccionar la práctica y reconfigurar los componentes según lo que desea experimentar.

El alumno puede obtener la lectura del instrumento mediante un sistema de adquisición de datos, en este caso un conversor analógico – digital o un microcontrolador que cuente con este recurso, de modo que el dato resultante de la conversión lo procese el microcontrolador. O bien, dicha lectura, se alcanza visualmente a través de un sistema de video, como una webcam, tal es el ejemplo que presenta en la Figura 3.12.

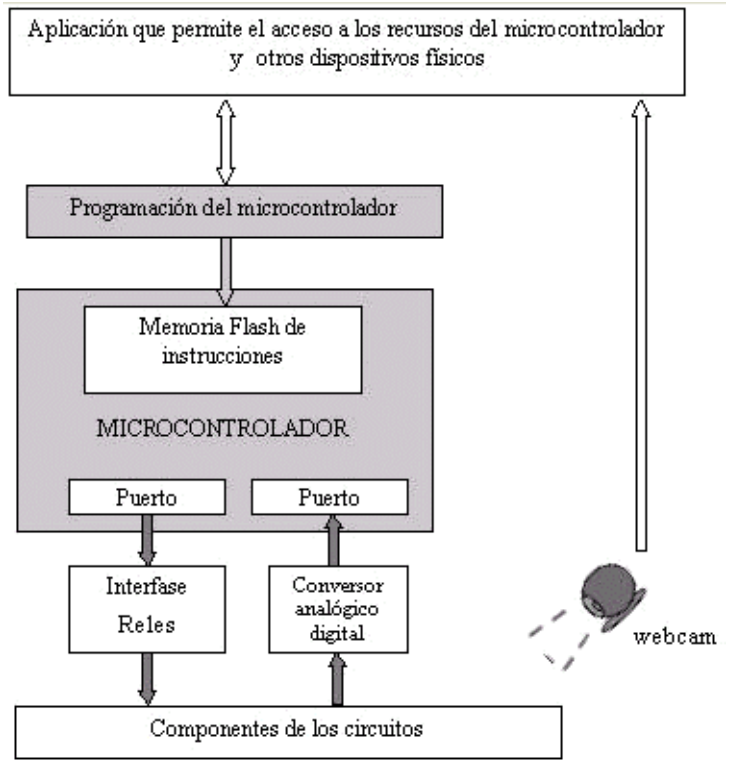


Figura 3.12 - Acceso a los recursos físicos

La aplicación que permite acceder a los recursos físicos puede procesarse en una computadora localmente o puede ejecutarse de modo remoto con una arquitectura cliente – servidor.

Además del acceso a los dispositivos físicos, se pueden incorporar algunas herramientas para el control de acceso, agenda de actividades y reservas de

recursos. También un sistema de bases de datos para el seguimiento de alumnos y material de apoyo. Para todas estas funciones se puede recurrir a algún sistema de aulas virtuales ya desarrollado de manera de aprovechar su funcionalidad.

La Figura 3.13 presenta la utilidad de acceso al laboratorio remoto gestionada por el sistema de aula virtual.

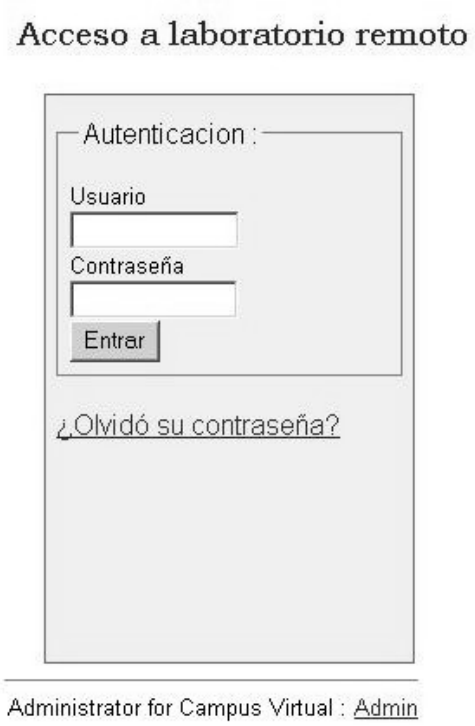


Figura 3.13 - Identificación y validación de usuario

La Figura 3.14 muestra como el usuario realiza la reserva de recursos. En el caso de que en la fecha y horarios seleccionados los recursos estén reservados, el sistema lo tendría que informar para que el usuario intente nuevamente con otra fecha y horario. Esta utilidad normalmente no se encuentra en los sistemas de aula virtual, por lo que debería ser desarrollada para el sistema de laboratorio remoto.

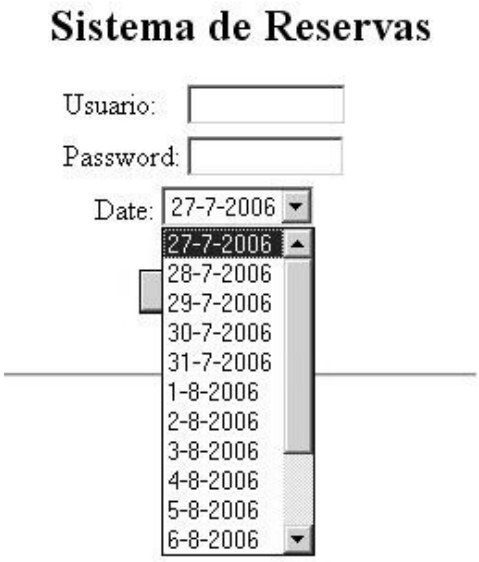


Figura 3.14 - Solicitud de reserva de recursos

Si los recursos están disponibles se permite el acceso a la página del laboratorio remoto en la cual se selecciona la práctica que el alumno quiere desarrollar, como se ve en la Figura 3.15. En este caso solamente se elige entre un circuito serie o un circuito paralelo mediante uno u otro hipervínculo.

Figura 3.15 - Página de acceso a la ejercitación remota

Cada uno de los hipervínculos dirige a una página html como la que se presenta en la Figura 3.16. En la misma se representa el circuito ensayado y el resultado de la medición en un instrumento.

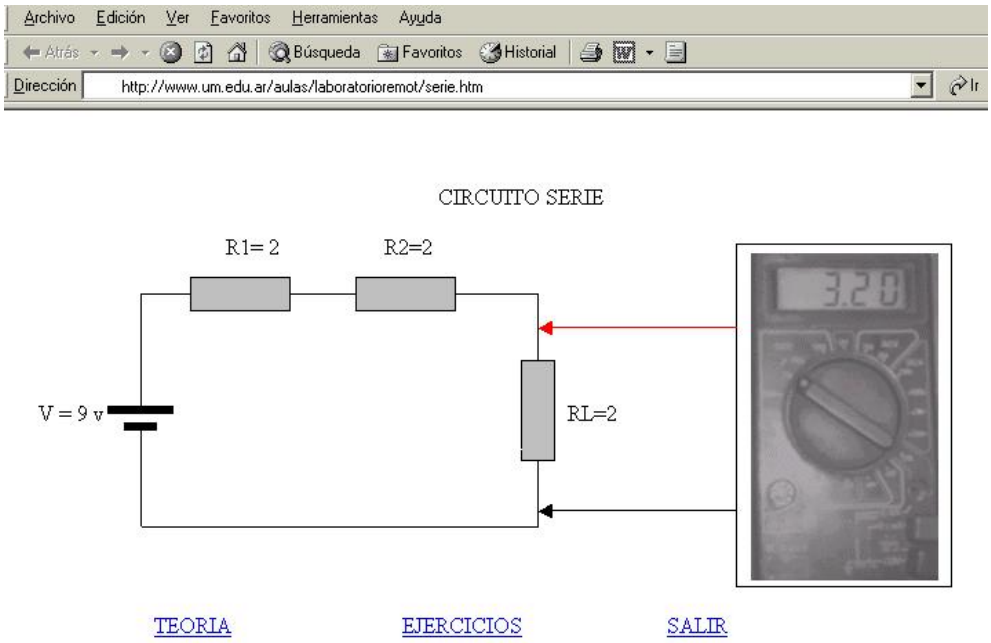


Figura 3.16 - Página de práctica de circuito serie

Estos hipervínculos pueden llamar a funciones que ejecuta el microcontrolador. De esta manera al seleccionar la página para el circuito serie o el paralelo se configura el sistema de interconexión, proceso que se lleva a cabo en modo transparente para el usuario.

La visualización de la medición del instrumento puede obtenerse de un sistema de captura de video o que el microcontrolador lea la salida de un conversor analógico digital en uno de sus puertos de entrada – salida. También la página ofrece links a material de teoría del tema objeto de estudio y a ejercicios que permitan complementar la experiencia la experiencia de laboratorio.

La Figura 3.17 presenta un ejemplo de material de apoyo. Es una ejercitación del tipo de opción múltiple relacionada con la resolución de circuitos series y paralelo. Es interesante este tipo de ejercitación ya que el sistema de aula

virtual le indica al alumno, una vez finalizado el ejercicio los resultados obtenidos y las respuestas correctas. Además todo el proceso queda registrado.

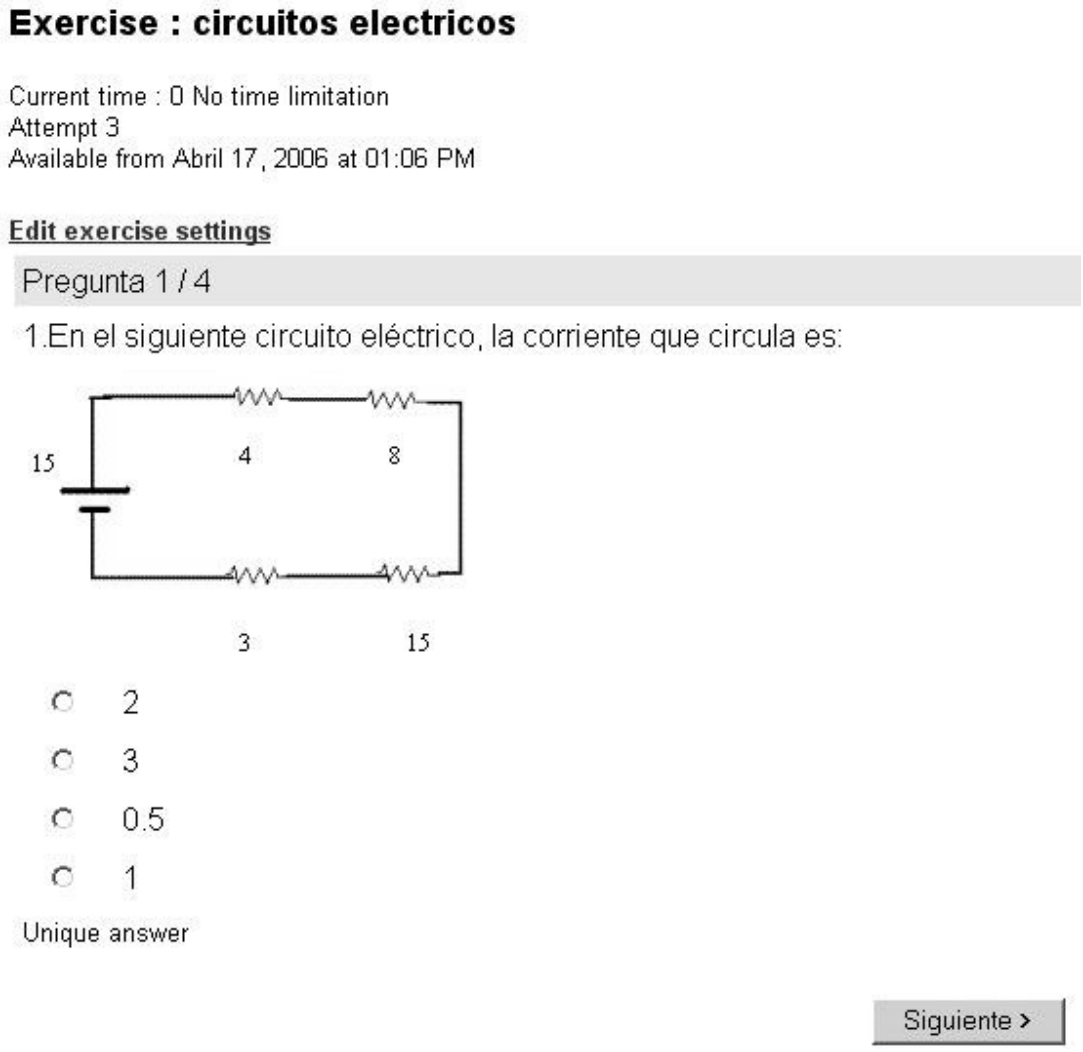


Figura 3.17 - Ejercitación relacionada con la práctica desarrollada

La Figura 3.18 muestra el seguimiento que se puede obtener de los usuarios.

La Figura 3.19 presenta las estadísticas de los accesos al sistema.

En estas imágenes se observa el modo en que las utilidades de control de acceso, material de apoyo, ejercicios, y seguimiento de un sistema de aulas

virtuales [19] pueden emplearse para algunas de las funciones que requiere la arquitectura de un laboratorio remoto. [20]

Statistics of exercise
Statistics of exercise : circuitos electricos

- Weighting : 10
- Minimum : 0
- Maximum : 5
- Average : 2.08
- Average Time (s.) : 38

- User attempts : 3
- Total attempts : 6

Estudiante	Minimum	Maximum	Average	Attempts	Average Time (s.)
Marianetti Osvaldo	0	2.5	1.25	2	37.5
Millan Emmanuel Nicolas	0	0	0	0	0
Perez Juan	2.5	5	3.33	3	45
Sanchez Jose	0	0	0	1	18

Figura 3.18 - Estadísticas con los resultados de los alumnos para el seguimiento de los mismos

Traffic Details

Abril 2006

Período : [Año] [Mes] [Día] || Visto por : [Día] [Hora] || [Mes anterior] [Mes siguente]

Día	Accesos	%
6 Abr 2006	7	19 %
10 Abr 2006	3	8 %
11 Abr 2006	1	3 %
12 Abr 2006	1	3 %
17 Abr 2006	18	49 %
18 Abr 2006	4	11 %
26 Abr 2006	2	5 %
28 Abr 2006	1	3 %
Total	37	

Figura 3.19 - Estadísticas con el detalle de los accesos a los recursos

Integrando todos los componentes descriptos se está en presencia de un entorno remoto multiusuario. En el diseño de esta simulación se ha seguido un proceso de abajo hacia arriba y se han sumado módulos con distintos servicios, algunos desarrollados especialmente para este entorno en particular y otros que pueden estar implementados previamente. Esta característica es interesante ya que permitiría una implementación modular, de manera de incorporar o modificar funciones al entorno remoto. También se observa en la

Figura 3.20 que el conjunto de estos módulos verifica la estructura de capas propuesta para el diseño e implementación de los laboratorios remotos.

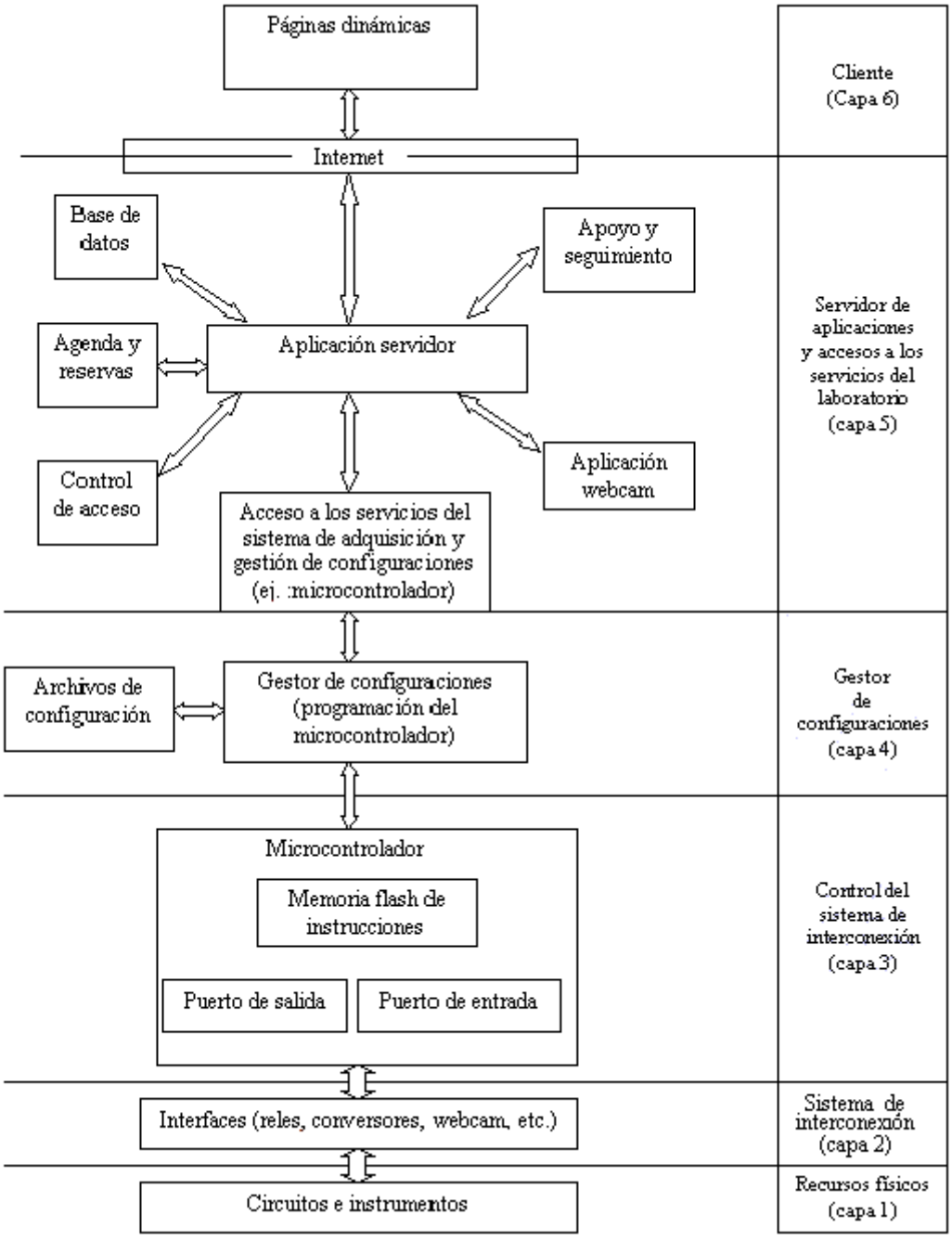


Figura 3.20 - Arquitectura de un laboratorio remoto. Modelo de capas.

Componentes del laboratorio

En base a este modelo de capas se pueden incorporar o quitar funcionalidades en forma modular, sin tener que modificar todos los niveles de la implementación. Por ejemplo, para modificar el material de apoyo se podría hacer actuando sobre el módulo correspondiente de la capa 5 (servidor de aplicaciones). Para cambiar las experiencias de laboratorio, se debería actuar sobre los archivos de configuración y el gestor de configuraciones en la capa 4. En la medida que el control del sistema de interconexión y el sistema de interconexión presenten la posibilidad de alterar sus conexiones a partir de las distintas configuraciones, el laboratorio remoto se hace flexible y se pueden cambiar o incorporar recursos físicos (circuitos o instrumentos) que permitan aumentar la cantidad y el tipo de prácticas que se quieran realizar.

La complejidad de esta estructura comparada con la simplicidad de las prácticas sugeridas en la simulación descripta en este apartado pueden poner en dudas o plantear reservas respecto de la validez de este tipo de implementación. Pero se debe tener en cuenta las ventajas de los laboratorios remotos citadas anteriormente y que, con los recursos intrínsecos que contienen los microcontroladores actuales, el tipo de experiencias y ejercicios de laboratorios que se pueden implementar configuran un amplio espectro de posibilidades en relación a la diversidad y la cantidad de las mismas.

CAPÍTULO 4
DISEÑO DE UN PROTOTIPO

Este capítulo trata sobre la implementación de un entorno que permite acceder en forma remota a los recursos de un kit de desarrollo de dispositivos lógicos programables y que admite la programación en lenguaje VHDL de un dispositivo lógico programable más la verificación remota del comportamiento del circuito programado.

4.1 Programación de PLDs

En el diseño de los sistemas digitales se distinguen las siguientes alternativas:

- Circuitos digitales discretos (SSI, MSI) (TTL, HCMOS, etc,).
- Circuitos microcontroladores
- Circuitos PLDs (dispositivos lógicos programables).

Las técnicas de diseño e implementación de circuitos digitales utilizando componentes discretos y/o microcontroladores encuentran una alternativa en los distintos tipos de circuitos PLDs. Estos dispositivos permiten en algunas aplicaciones bajar los costos del sistema y los tiempos de desarrollo mientras que en otras, según la complejidad y los requerimientos de velocidad del sistema digital son la única posibilidad de implementación. [21]

Si bien un entorno industrial (agroindustrias, industrias metalmecánicas, industrias del sector primario, empresas de servicios) con restricciones económicas y operativas es reticente a la incorporación de nuevas tecnologías por motivos de relación costo/beneficio, confiabilidad, tiempos de desarrollo y mantenimiento, también es un medio que puede requerir soluciones que involucren sistemas digitales (control , instrumentación, desarrollos propios). Aquí es donde valen algunas ventajas comparativas de los sistemas basados en PLDs.

La proliferación de las aplicaciones de estos circuitos digitales (procesamiento de imágenes, codificación y procesamiento de imágenes sensoriales, procesadores matriciales, control de servomecanismos, cómputo reconfigurable) y el aumento de la complejidad de dichos circuitos ha impulsado cambios en los métodos y herramientas utilizados en el diseño digital. Los circuitos integrados a gran escala contienen millones de transistores. Los métodos de diseño asistido por computadora y los dispositivos lógicos programables han permitido el rápido paso del concepto al circuito terminado, poniendo énfasis en los diseños modulares jerárquicos que utilizan bibliotecas estándar y otros módulos prediseñados. [22]

Ante esta perspectiva se hace necesario adecuar y actualizar el curriculum de las carreras de formación de grado en ingeniería incorporando las temáticas relacionadas con el diseño basado en PLDs. Se deben abarcar los cambios citados, pero también conservar los fundamentos. Un buen ingeniero de diseño digital necesita bases sólidas de teoría fundamental, junto con el conocimiento de los principios de diseño del mundo real. [23]

4.2 Introducción de prácticas remotas

En este apartado se describe la implementación de un entorno remoto para prácticas de la laboratorio de programación de dispositivos lógicos programables (PLDs) utilizando el lenguaje de descripción de hardware VHDL.

Como primera etapa se seleccionan los contenidos que se incorporan a la cátedra de Circuitos Digitales.

- PLD. Descripción y clasificación
- VHDL
- Herramientas de diseño
- Sistemas de desarrollo
- Diseño de aplicaciones

Estos contenidos se desarrollan en un cursado presencial normal, pero además se puede acceder a material didáctico como apuntes, tutoriales y/o trabajos prácticos de gabinete presentes en un sistema de aula virtual.

Para realizar prácticas presenciales se cuenta solamente con un Started – kit basado en tecnología de Altera Corp. Esta limitación hace que se haga foco en la descripción y la simulación de ejemplos y que sea muy restringido el acceso al sistema básico de desarrollo. Para mejorar esta situación se considera la implementación de un acceso remoto al kit mediante una aplicación tipo cliente-servidor, a la cual se puede ingresar también desde el sistema de aula virtual antes citado. Se ha escogido un entorno de aula virtual debido a las características generales que presentan estas herramientas: publicación de documentos, agendas de actividades, listas, foros, etc. En el caso particular del acceso al kit de PLD, se toma ventaja de la utilidad que gestiona el acceso a los usuarios, limitando el uso remoto de la aplicación solamente a quienes están autorizados como se muestra en la Figura 4.1.

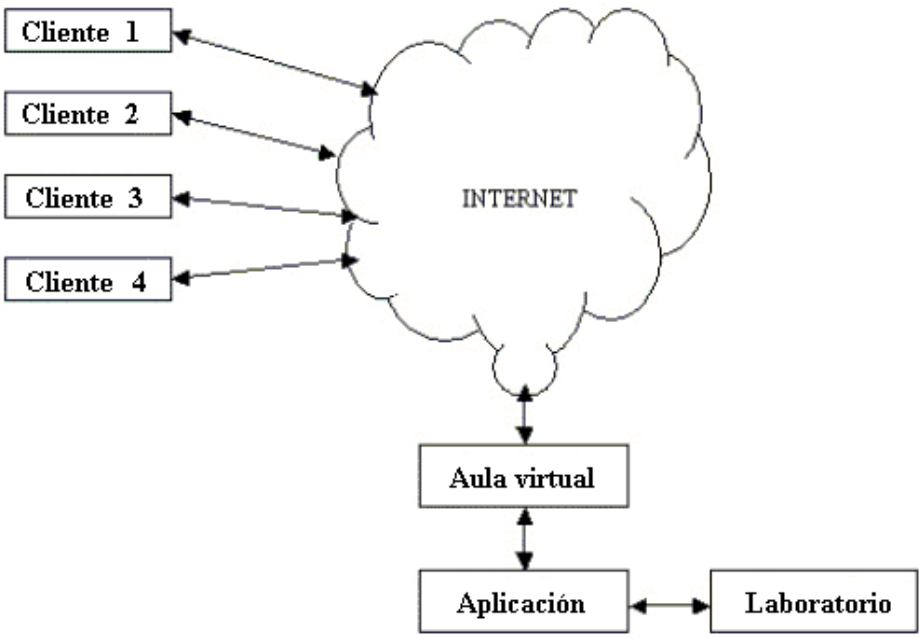


Fig. 4.1- Componentes del laboratorio remoto.

4.3 Recurso físico local

El recurso físico al que se debe acceder en forma remota, como ya se señaló, es un Started – kit Altera [25] MAX 7000 EPM128 [24], que dispone de un PLD Altera EPM7128AETC100-10, 100 pin TQFP package. Este kit de desarrollo permite el acceso a los pines del PLD mediante cuatro conectores de expansión de 2 x 20 x 0.1 inch pitch. El kit que se presenta en la Figura 4.2 posee 4 pulsadores para entradas de señales, cuatro displays de 7 segmentos y un conector Byte BlasterMV que se conecta al puerto paralelo de una PC, configurado como puerto JTAG, para la programación del PLD.

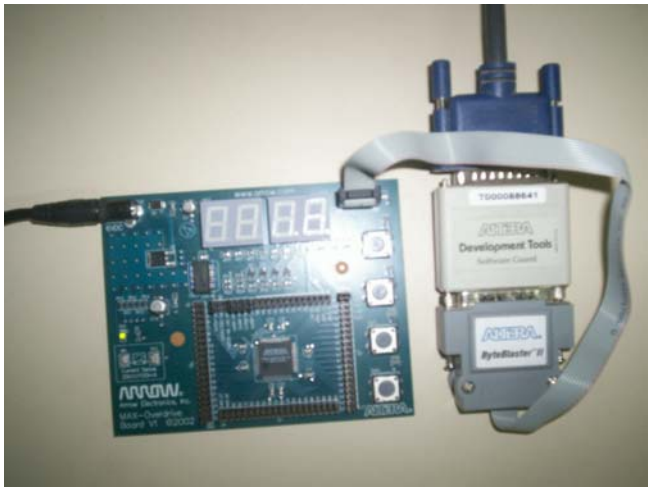


Figura 4.2 - Imagen del kit de PLD

El entorno integrado de desarrollo (IDE) utilizado para realizar las experiencias con el kit se basa en las herramientas de Altera: MaxPlus II [26] y/o Quartus II [27]. Estos entornos permiten la edición, síntesis, compilación, simulación, análisis de tiempo, configuración y programación de archivos con formato VHDL. Además presentan una interface gráfica de usuario (GUI) que es muy adecuada para su operación en modo local. Sin embargo también permiten realizar procesos utilizando una línea de comandos o scripts de comandos, característica que es útil para implementar una aplicación que permita el acceso remoto a los recursos del Kit.

4.4 El laboratorio remoto para la programación de PLDs

La arquitectura del laboratorio remoto implementado para la programación de PLDs se basa en tres servicios:

a) los servicios de gestión de identificación y validación de usuarios, para lo cual se toma ventaja del entorno de aula virtual.
b) los servicios del entorno IDE para la generación, compilación y programación de un proyecto aplicado al kit de desarrollo del PLD.
c) la verificación de las prácticas realizadas mediante la generación de los vectores de prueba que verifiquen el funcionamiento del PLD una vez programado.

Esta arquitectura se representa en la Figura 4.3.

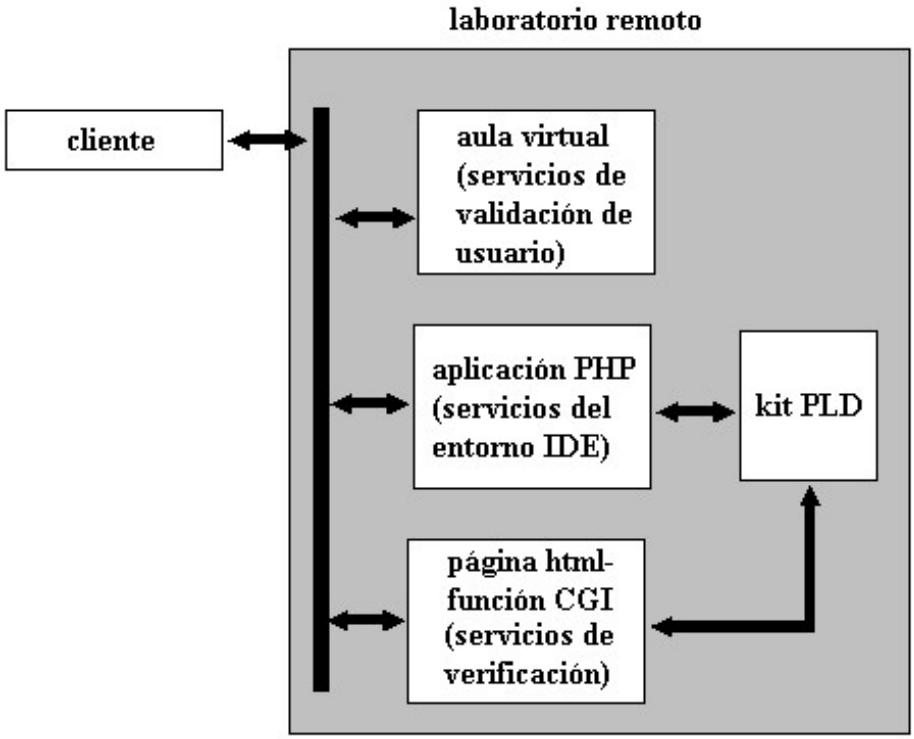


Fig. 4.3 - Arquitectura del prototipo de laboratorio remoto

4.5 Aula virtual. Validación de usuarios

El entorno de aula virtual permite el acceso a los usuarios registrados a aquellos cursos en los que se pueden inscribir. Mediante su nombre de usuario y su contraseña ingresan a los cursos permitidos. En este caso a la cátedra de Circuitos Digitales. En las Figuras 4.4, 4.5 y 4.6 se puede observar la pantalla de validación de usuario, los cursos a los que puede acceder el usuario validado y la página de la Cátedra de Circuitos Digitales del sistema de aula virtual, respectivamente.

Figura 4.4 - Validación de usuario

Figura 4.5 - Cursos inscriptos del usuario

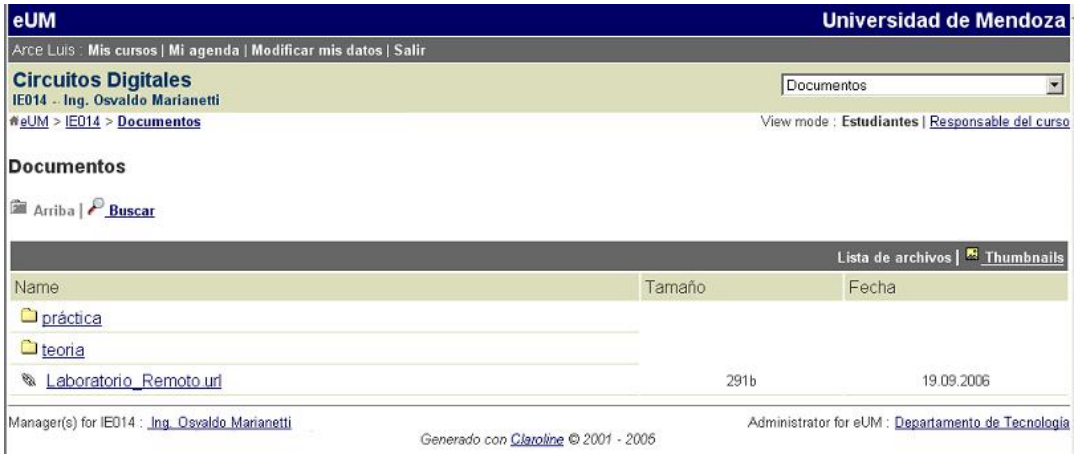


Figura 4.6 - Cátedra virtual de la asignatura Circuitos Digitales

Al acceder al sitio de la cátedra de Circuitos Digitales, el alumno se encuentra con un hipervínculo a la página principal del Laboratorio Remoto. Dicha página se presenta en la Figura 4.7.

Figura 4.7 - Imagen de página principal del laboratorio remoto

De esta manera se ingresa al servidor que permite utilizar los servicios del entorno de desarrollo. En esta página principal se presentan las prácticas que se pueden experimentar.

4.6 Acceso remoto al entorno de desarrollo. Aplicación PHP

El diseño de la teleoperación de los recursos físicos se basa en una aplicación cliente – servidor programada en PHP [28]. En el servidor está instalado el entorno IDE Quartus II 3.0 y el puerto JTAG de programación del kit está conectado al LPT1 de esta computadora para materializar el acceso.

La página principal del laboratorio remoto simplemente presenta un conjunto de trabajos prácticos de laboratorio, que pueden seleccionarse mediante enlaces dispuestos con ese fin.

Al elegir el trabajo práctico se accede a una página en la que se debe escribir el código VHDL de circuito que se debe implementar sobre el kit de PLD, como se observa en la Figura 4.8.

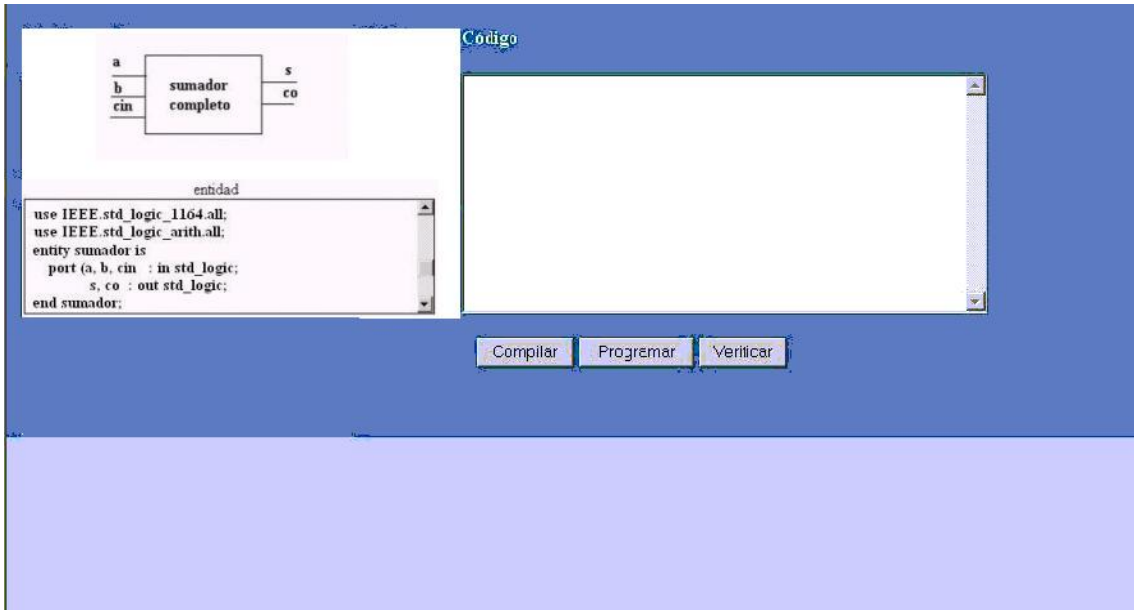


Figura 4.8 - Página dinámica para introducir el código VHDL

Esta elección de trabajo práctico, además de permitir enviar al servidor el código que se quiere ensayar, permite en modo transparente para el usuario, la configuración de los pines del kit de PLD para la experiencia elegida. Cada trabajo práctico tiene asociado un proyecto predefinido en el entorno Quartus II. Esta situación se refleja en la Figura 4.9.

La página de cada trabajo práctico presenta un esquema del circuito que se va a ensayar y además se entrega la parte del código VHDL correspondiente a la entidad de ese circuito. El hecho de utilizar una entidad prefijada permite asegurar los pines de entrada y salida del PLD, evitando posibles daños en el hardware del laboratorio.

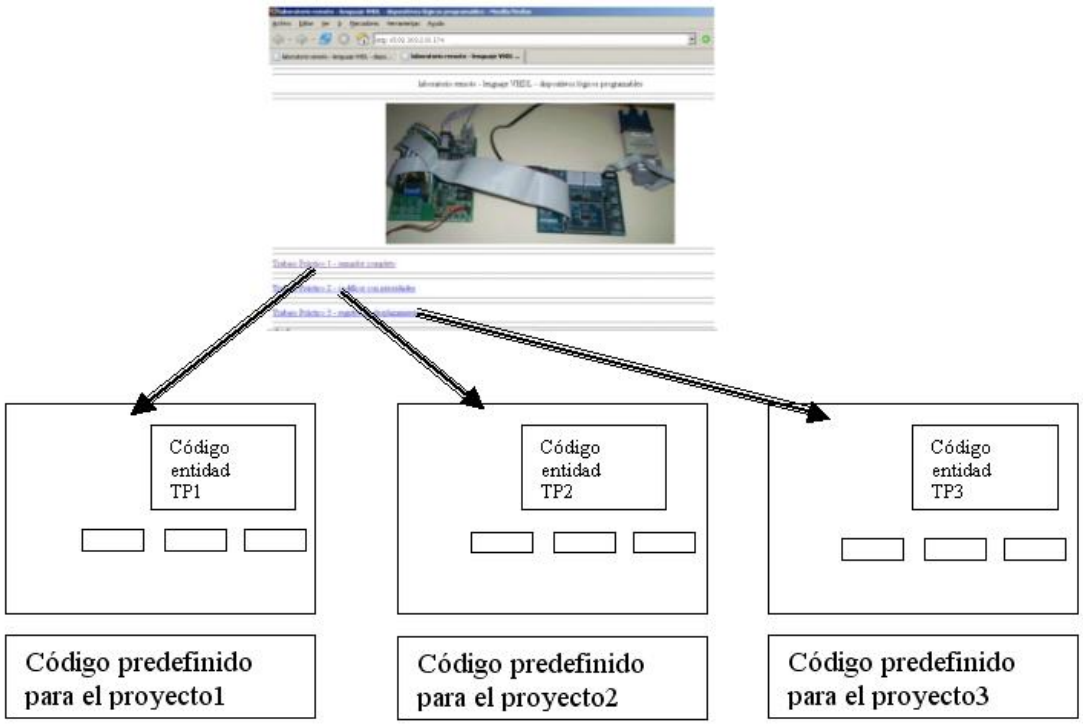


Figura 4.9 - Selección de proyectos predefinidos

Para ampliar este concepto, se pueden comparar los reportes generados por el IDE Quartus II, para el mismo código VHDL con una entidad definida por el usuario y otra definida para su aplicación en el laboratorio remoto.

Descripción de un sumador (denominado proyecto1) definida previamente por la aplicación instalada en el servidor.	Descripción de un sumador totalmente escrita por el usuario.

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
 	
entity proyecto01 is
port (a : in std_logic;
b : in std_logic;
cin : in std_logic;
s : out std_logic;
cout : out std_logic);
end proyecto01;
architecture rtl of proyecto01 is
 	
end rtl;
	library IEEE;
	use IEEE.std_logic_1164.all;
	use IEEE.std_logic_arith.all;
	
	entity sumador1 is
	port (a : in std_logic;
	b : in std_logic;
	cin : in std_logic;
	sum : out std_logic;
	cout : out std_logic);
	end sumador1;
	architecture rtl of sumador1 is
	begin
	sum <= (a xor b) xor cin;
	cout <= (a and b) or (cin and a) or (cin and b);
	end rtl;

Al compilar el proyecto “proyecto1”, en el reporte de pines se obtiene que la distribución de pines es a => 16, b => 17, cin => 14, s => 8 y cout => 9. Estas asignaciones de pines se corresponden con las implementadas físicamente en la interfase entre el PLD y los dispositivos de verificación. De esta manera, limitando el código del usuario a la descripción del circuito, se asegura que la

distribución de pines del PLD sea la adecuada para evitar errores de configuración y posibles fallas. Este resultado se observa en la Figura 4.10.

CHIP "proyecto01" ASSIGNED TO AN: EPM7128AETC100-10

Pin Name/Usage	Location	Dir.	I/O Standard	Voltage	I/O Bank	User Assignment
RESERVED_INPUT	1					
RESERVED_INPUT	2					
VCCIO	3	power		3.3V		
TDI	4	input	LVTTL			N
RESERVED_INPUT	5					
RESERVED_INPUT	6					
RESERVED_INPUT	7					
s	8	output	LVTTL			Y
cout	9	output	LVTTL			Y
RESERVED_INPUT	10					
GND	11	gnd				
RESERVED_INPUT	12					
RESERVED_INPUT	13					
cin	14	input	LVTTL			Y
TMS	15	input	LVTTL			N
a	16	input	LVTTL			Y
b	17	input	LVTTL			Y
VCCIO	18	power		3.3V		
RESERVED_INPUT	19					
RESERVED_INPUT	20					
RESERVED_INPUT	21					
RESERVED_INPUT	22					
RESERVED_INPUT	23					
RESERVED_INPUT	24					
RESERVED_INPUT	25					
GND	26	gnd				
RESERVED_INPUT	27					
RESERVED_INPUT	28					
RESERVED_INPUT	29					
RESERVED_INPUT	30					
RESERVED_INPUT	31					
RESERVED_INPUT	32					
RESERVED_INPUT	33					
VCCIO	34	power		3.3V		
RESERVED_INPUT	35					
RESERVED_INPUT	36					
RESERVED_INPUT	37					
GND	38	gnd				
VCCINT	39	power		3.3V		
RESERVED_INPUT	40					
RESERVED_INPUT	41					
RESERVED_INPUT	42					
GND	43	gnd				

Figura 4.10- Reporte de distribución de pines para un proyecto predefinido.

Al compilar el proyecto “sumador1” (sin asignaciones previas), el entorno IDE Quartus asigna automáticamente los pines a las entradas y las salidas. Esto se observa el reporte dado por el archivo sumador1 (archivo de texto, reporte tipo PIN), donde a las entradas a, b y cin les corresponden los pines 27, 50 y 85 mientras que a las salidas sum y cout les corresponden los pines 1 y 2 respectivamente, como se presenta en la Figura 4.11. Estas asignaciones pueden no coincidir con las implementadas a nivel de interfase entre el PLD y los dispositivos de verificación.

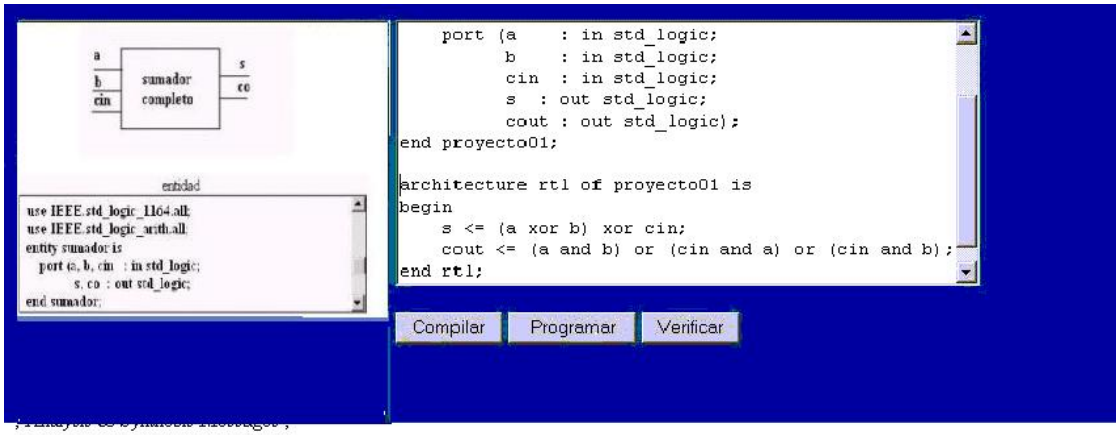
CHIP "sumador1" ASSIGNED TO AN: EPM7128AETC100-10

Pin Name/Usage	Location	Dir.	I/O Standard	Voltage	I/O Bank	Use
cout	1	output	LVTTL			N
sum	2	output	LVTTL			N
VCCIO	3	power		3.3V		
TDI	4	input	LVTTL			N
GND*	5					
GND*	6					
GND*	7					
GND*	8					
GND*	9					
GND*	10					
GND	11	gnd				
GND*	12					
GND*	13					
GND*	14					
TMS	15	input	LVTTL			N
GND*	16					
GND*	17					
VCCIO	18	power		3.3V		
GND*	19					
GND*	20					
GND*	21					
GND*	22					
GND*	23					
GND*	24					
GND*	25					
GND	26	gnd				
a	27	input	LVTTL			N
GND*	28					
GND*	29					
GND*	30					
GND*	31					
GND*	32					
GND*	33					
VCCIO	34	power		3.3V		
GND*	35					
GND*	36					
GND*	37					
GND	38	gnd				
VCCINT	39	power		3.3V		
GND*	40					
GND*	41					
GND*	42					
GND	43	gnd				
GND*	44					
GND*	45					
GND*	46					
GND*	47					
GND*	48					
GND*	49					
cin	50	input	LVTTL			N
VCCIO	51	power		3.3V		
GND*	52					
GND*	53					
GND*	54					
GND*	55					
GND*	56					
GND*	57					
GND*	58					
GND	59	gnd				
GND*	60					
GND*	61					
TCK	62	input	LVTTL			N
GND*	63					
GND*	64					
GND*	65					
VCCIO	66	power		3.3V		
GND*	67					
GND*	68					
GND*	69					
GND*	70					
GND*	71					
GND*	72					
TDO	73	output	LVTTL			N
GND	74	gnd				
GND*	75					
GND*	76					
GND*	77					
GND*	78					
GND*	79					
GND*	80					
GND*	81					
VCCIO	82	power		3.3V		
GND*	83					
GND*	84					
b	85	input	LVTTL			N
GND	86	gnd				
GND+	87					
GND+	88					
GND+	89					
GND+	90					
VCCINT	91	power		3.3V		
GND*	92					
GND*	93					
GND*	94					
GND	95	gnd				
GND*	96					
GND*	97					
GND*	98					
GND*	99					
GND*	100					

Figura 4.11 - Reporte de distribución de pines definidos automáticamente.

Por este motivo se genera previamente un proyecto con una asignación de pines predefinida. El usuario se debe ocupar de escribir el código VHDL que describe el circuito que se va a diseñar y ensayar. Este código se puede escribir directamente en el lugar dedicado en la página o copiar desde cualquier editor de textos. De este modo se verifica que el usuario solamente requiere instalados en su computadora un navegador y un simple editor de textos. Una vez que se ingresó el código se pueden usar los botones: Compilar, Programar y Verificar.

El botón Compilar llama a un programa PHP que permite ejecutar los comandos necesarios para que el entorno IDE instalado en el servidor genere un proyecto y lo compile. El usuario recibe un reporte de los resultados de este proceso remoto como se muestra en la Figura 4.12




```
---------------------------------
Info: *******************************************************************************
Info: Running Quartus II Analysis & Synthesis
Info: Version 3.0 Build 199 06/26/2003 SJ Full Version
Info: Processing started: Fri Sep 22 12:24:44 2006
Info: Command: quartus_map --lower_priority --import_settings_files=on --export_settings_files=off proyecto1 -c proyecto1
Warning: Feature SignalTap II is not available with your current license
Error: Node instance instantiates undefined entity proyecto1
Error: Quartus II Analysis & Synthesis was unsuccessful. 1 error, 1 warning
Error: Processing ended: Fri Sep 22 12:24:44 2006
Error: Elapsed time: 00:00:00
Info: Writing report file proyecto1.map.rpt
```

Figura 4.12 - Página dinámica con la recepción del reporte con los resultados de la compilación del código VHDL enviado.

El siguiente código es una porción del código de la página correspondiente a uno de los proyectos seleccionados. En estas líneas se presentan las acciones de cada botón (compilar, programar y verificar).

```
top.bottomFrame.document.form1.action = "http://localhost/compilar.php";
                top.bottomFrame.document.form1.submit();
        }
        else if (que == "programar"){
                top.bottomFrame.document.form1.action = "http://localhost/programar.php";
                top.bottomFrame.document.form1.submit();
        }
        else if (que == "verificar"){
                top.bottomFrame.location.href = "http://192.168.218.174/index04.htm";
```

Se observa que el botón “compilar” hace referencia a un script de PHP que permite procesar los comandos de compilación del entorno IDE Quartus II. Del mismo modo actúa el botón “programar”. Mientras que el botón “verificar” referencia a una dirección IP que es la que se le ha asignado al microcontrolador con Internet embebida y una página “index04.htm” que se encuentra embebida en el microcontrolador.

4.7 Compilación remota del código VHDL

Al presionar el botón Compilar se ejecuta el programa compilar.php que envía el texto vhdl al servidor. La aplicación se encarga de almacenarlo en una carpeta del disco rígido c:\prueba con el nombre proyecto1.vhd. El servidor corre en modo residente el siguiente código:

```
IF EXIST c:\prueba\proyecto1.vhd (goto hacer) ELSE GOTO FIN

:hacer

copy c:\prueba\proyecto1.vhd  c:\quartus\bin

del c:\prueba\*.vhd

cd c:\quartus\bin

quartus_map proyecto1.vhd

quartus_sh --flow compile proyecto1

copy proyecto1.* c:\altera

:FIN
```

Estos comandos constatan la presencia del archivo *.vhd que envía el cliente. Si este archivo está presente se ejecutan los comandos del Quartus II correspondientes a la generación y compilación del proyecto en cuestión. Todos los archivos resultantes de estos procesos se copian en una carpeta del disco servidor c:\altera. Algunos de estos archivos son los reportes que indican lo sucedido en el proceso de compilación del proyecto.

El código PHP compilar.php es el siguiente:

```
<?php
// recibe datos para compilar
$lineas = array('');
$dato = $_GET['texto'];
$archivo = fopen("c:\\prueba\\proyecto1.vhd", "w");
fwrite($archivo, $dato);
fclose($archivo);
// espera que termine de compilar
while ( !file_exists("c:\\altera\\proyecto1.map.rpt") ) sleep(10);
$a = fopen("c:\\altera\\proyecto1.map.rpt", "r");
```

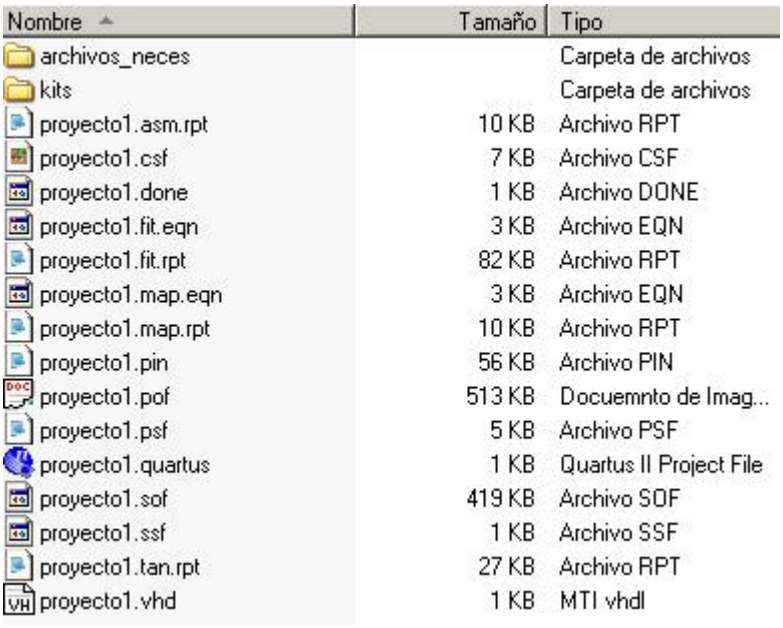
```
while ( !feof($a) ) {
        $c = fgetc($a);
        if ( $c == "\r" ) {
                echo "<br>";
        }
        else {
                echo $c;
        }
}
fclose($a);
system("del c:\altera\proyecto1.map.rpt");
?>
```

Este código recibe el archivo *.vhd del cliente y lo almacena en la carpeta c:\prueba, espera que el Quartus II lo procese y toma el archivo "proyecto1.map.rpt" que contiene la información que resulta de la compilación del archivo vhd recibido y se lo envía al cliente. De esta manera el alumno ha logrado el acceso remoto al entorno de desarrollo del Kit, mediante la utilización de un web browser y un editor de textos.

4.8 Programación remota del PLD

Todos los archivos resultantes del proceso de compilación se almacenan en la carpeta c:\altera. En el caso de que la compilación no de errores, en este conjunto de archivos se encuentran los necesarios para la programación del kit de PLD, situación que se presenta en la Figura 4.13.

El código asociado al botón programar es similar al del botón de compilación. Si en la carpeta c:\prueba se encuentra un archivo con extensión *.pof, se ejecutan los comandos de programación del Quartus.



Nombre	Tamaño	Tipo
archivos_neces		Carpeta de archivos
kits		Carpeta de archivos
proyecto1.asm.rpt	10 KB	Archivo RPT
proyecto1.csf	7 KB	Archivo CSF
proyecto1.done	1 KB	Archivo DONE
proyecto1.fit.eqn	3 KB	Archivo EQN
proyecto1.fit.rpt	82 KB	Archivo RPT
proyecto1.map.eqn	3 KB	Archivo EQN
proyecto1.map.rpt	10 KB	Archivo RPT
proyecto1.pin	56 KB	Archivo PIN
proyecto1.pof	513 KB	Docuemnto de Imag...
proyecto1.psf	5 KB	Archivo PSF
proyecto1.quartus	1 KB	Quartus II Project File
proyecto1.sof	419 KB	Archivo SOF
proyecto1.ssf	1 KB	Archivo SSF
proyecto1.tan.rpt	27 KB	Archivo RPT
proyecto1.vhd	1 KB	MTI vhdl

Figura 4.13. Archivo generados por la compilación exitosa de un proyecto en el entorno Quartus II

El entorno Quartus sólo genera los archivos de programación si la compilación no da errores. Esta característica asegura que el PLD no se pueda programar en el caso de errores de compilación, salvaguardando el hardware del kit.

```
<?php
if ( file_exists("c:\\altera\\proyecto1.pof") ) {
        $archivo = fopen("c:\\prueba\\programar.flg", "w");
        fwrite($archivo, 0);
        fclose($archivo);
}
else {
        echo "¡ERROR! no puede programar a menos que la compilación sea exitosa";
}
?>
```

4.9 Verificación

Una parte central de las experiencias de laboratorio, ya sean presenciales o remotas, es la verificación de los procesos realizados, para lo cual se deben observar los resultados de la experiencia. Esta tarea que resulta básica en un entorno presencial es quizás una de las que presenta mayores dificultades en un entorno remoto que se pueden solucionar mediante cámaras web que capturan información y la presentan al usuario .

Una alternativa para alcanzar este fin, se basa en generar vectores de prueba que se emplean como señales de entrada al PLD, y recuperar las salidas del mismo como respuesta a las señales de entrada. La respuesta del PLD a los vectores de prueba se expresan mediante los valores ceros o unos que toman las salidas físicas. De esta manera se puede comprobar con relativa facilidad que el código programado para el PLD cumpla con las condiciones de funcionamiento esperadas. También se puede fijar una secuencia de valores en los vectores de entrada, de manera de tomar muestras de las salidas en periodos de tiempo regulares durante un intervalo adecuado para la experiencia que se esté practicando.

El botón de verificación es en realidad un enlace a un servidor web embebido en un microcontrolador cuyo servicio es llamar a una función CGI que es la encargada de llevar a cabo el proceso de generar los vectores de prueba y obtener la respuesta del PLD.

Esta página, que se muestra en la Figura 4.14, presenta la tabla de la verdad del circuito que se está diseñando y para cada combinación de las variables de entrada se muestra la salida que se espera debe tener el circuito descripto en el código VHDL que se envió, compiló y programó en modo remoto mediante la teleoperación del entorno Quartus II.

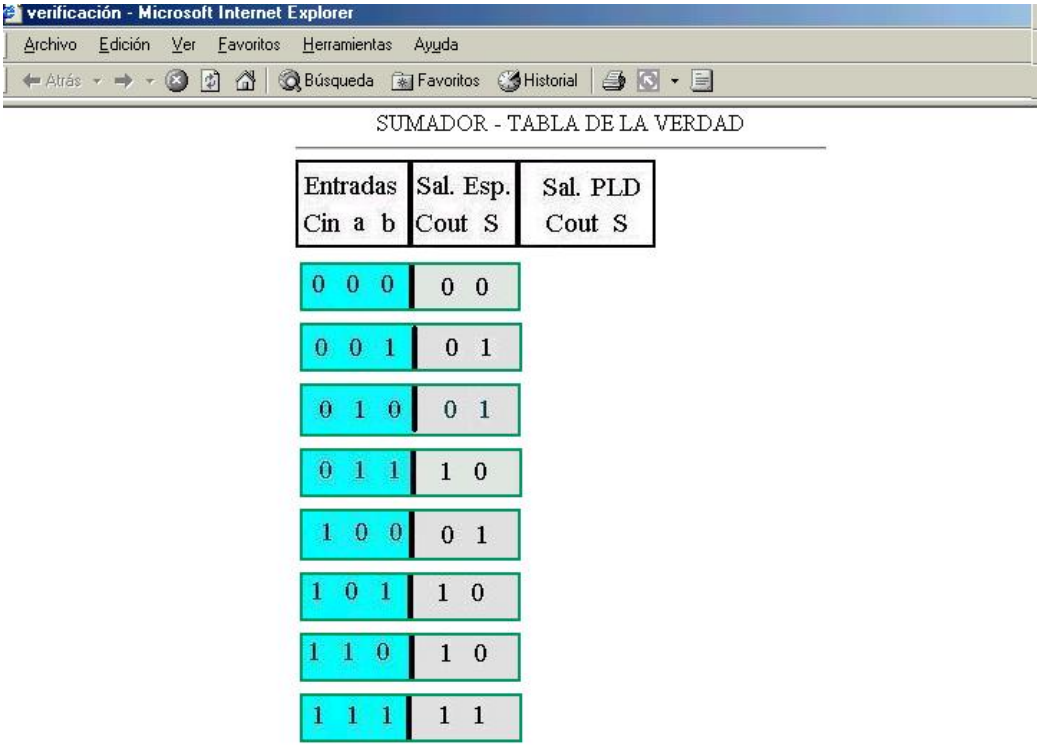


Figura 4.14 - Imagen de la página de verificación

Cada entrada de la tabla tiene asociado un link a una función CGI que se ejecuta en el microcontrolador Rabbit. Esta función escribe en uno de los puertos del microcontrolador configurado como salida, los valores de las entradas Cin, a y b del circuito que se ha implantado en el PLD. Y lee los valores de las salidas del PLD Cout y S para esa combinación de las entradas. El resultado de este proceso se presenta en la página verificación como Sal. PLD, para que pueda comparar con las salidas esperadas. De este modo, el alumno verifica en modo remoto el comportamiento del circuito que diseñó y describió mediante el lenguaje VHDL.

Una porción del código HTML [29] embebido de la página correspondiente a la verificación del trabajo práctico de un circuito sumador es el siguiente, donde muestra como se llaman a las funciones CGI para cada combinación de la tabla

de la verdad del sumador y se obtine el valor de las salidas del PLD “suma” y “acarreo”:

```
<TABLE BORDER="0" CELLSPACING="2" CELLPADDING="1">
        <TR>
        <TR>
                <TD> <A HREF="/vector0.cgi"> <img SRC="cero.jpg"> </A> </TD>
                <TD> <!--#echo var="acarreo"--> </TD>
                <TD> <!--#echo var="suma"--> </TD>
        </TR>
        <TR>
                <TD> <A HREF="/vector1.cgi"> <img SRC="uno.jpg"> </A> </TD>
                <TD> <!--#echo var="acarreo"--> </TD>
                <TD> <!--#echo var="suma"--> </TD>
        </TR>
```

Códigos html similares implementan la verificación de las otras actividades o trabajos prácticos previstos (decodificador y contador binario) con llamadas a las funciones CGI que corresponden a la verificación de dichas experiencias.

El microcontrolador que permite materializar la verificación debe cumplir con algunos requisitos tales como una cantidad suficiente de puertos de entrada – salida, capacidad de memoria flash para el código de generación de distintos vectores y soporte de funciones CGI.

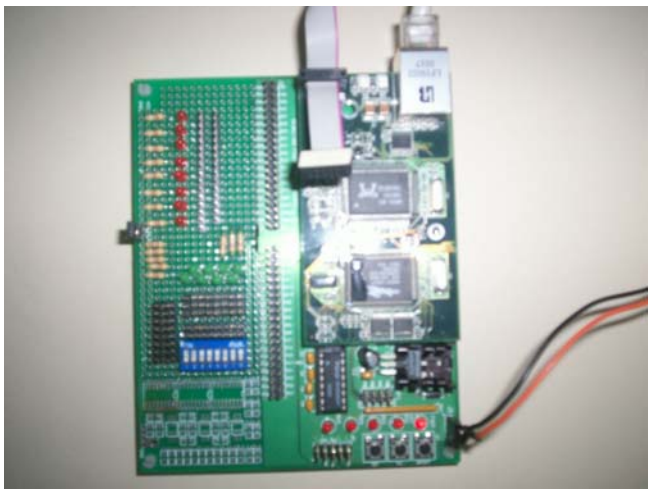


Figura 4.15 - Imagen del módulo RCM 2100

Con este fin se utilizó un módulo RCM 2100 de Zworld, el cual se muestra en la Figura 4.15. El mismo posee un microcontrolador Rabbit de 8 bits de la línea 2000, 512 KB de memoria flash y 512 KB de memoria RAM, , que sumados a su entorno de desarrollo (Dynamic C), y las prestaciones de internet embebida que posee, disminuyen notablemente los tiempos requeridos para el desarrollo. [30]

En el prototipo se configuró el microcontrolador con 8 salidas y 8 entradas, de modo de poder experimentar con circuitos que requieran hasta 8 señales de entrada y 8 salidas como máximo. Esta es una limitación respecto del número de entradas – salidas que tiene el PLD, pero es factible usar otros puertos del microcontrolador ya que cuenta con 5 puertos de 8 bits cada uno o incorporar circuitos multiplexores. El código correspondiente a la página de verificación y la generación de vectores de prueba y lectura de las salidas del PLD para cada una de las prácticas propuestas se encuentran embebidos en la flash del RCM2100. A este módulo se le asigna una dirección IP de modo que se comporta como un host más en una red [31], de manera que los servicios que requiere el laboratorio remoto interactúan sobre una red TCP/IP. La configuración de recursos descripta se presenta en la Figura 4.16.

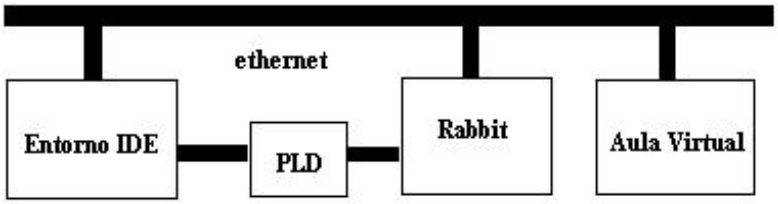


Figura 4.16 - Imagen de la red de recursos del laboratorio

Un ejemplo de código escrito en Dynamic C del entorno del Rabbit donde se le asigna la dirección IP y se configura como host de red al módulo RCM2100, además de cómo se embeben páginas html, se invoca a funciones CGI y se abre una conexión con la aplicación del servidor es el siguiente cuadro:

```
/*****************************************************
configuración TCP/IP
*****************************************************/
#define MY_IP_ADDRESS   "192.168.218.174"
#define MY_NETMASK      "255.255.255.224"
#define MY_GATEWAY      "192.168.218.221"
/*****************************************************
#class auto
/*****************************************************
archivos embebidos
*****************************************************/
#ximport "index.html"
#ximport "/foto1.jpg"    foto1_jpg
#ximport "/index0.htm"  index0_html
/*****************************************************
definiciones de la conexión
*****************************************************/
#define TCP_BUF_SIZE 2048
#define HTTP_MAXSERVERS 2
#define MAX_TCP_SOCKET_BUFFERS 2
#define REDIRECTHOST                "192.168.218.174"
#define REDIRECTTO                  "http://" REDIRECTHOST1 ""
#memmap xmem
#use "dcrtcp.lib"
#use "http.lib"
*****************************************************
definición de tipos y manejadores
*****************************************************/
const HttpType http_types[] =
{
   { ".shtml", "text/html", shtml_handler}, // ssi
   { ".htm", "text/htm", NULL},          // html
   { ".gif", "image/gif", NULL},
   { ".cgi", "", NULL},                  // cgi
   { ".jpg", "image/jpg", NULL}
};
/*****************************************************
```

```
Pograma ejemplo de control de entrada y salida ( función CGI)
*****************************************************/
int pruebaleds(HttpState* state)

{
  WrPortI(SPCR, &SPCRShadow, 0x84);        // setea el puerto A como salida
  WrPortI(PADR, &PADRShadow, 0xff);                  // apaga todos los leds


  while (1){                                        // endless loop
   costate {
                  BitWrPortI(PADR, &PADRShadow, 1, 0);
                  waitfor(DelayMs(1000));
                  BitWrPortI(PADR, &PADRShadow, 0, 0);
                  waitfor(DelayMs(1000));
                   //poner una variable que acumle delayms para salir cuando llegua a 16000
        }
   costate {
                  BitWrPortI(PADR, &PADRShadow, 1, 1);
                  waitfor(DelayMs(2000));
                  BitWrPortI(PADR, &PADRShadow, 0, 1);
                  waitfor(DelayMs(2000));
        }
   costate {
                  BitWrPortI(PADR, &PADRShadow, 1, 2);
                  waitfor(DelayMs(4000));
                  BitWrPortI(PADR, &PADRShadow, 0, 2);
                  waitfor(DelayMs(4000));
        }
    }
    cgi_redirectto(state,REDIRECTTO);
    return (0);
}  // fin pruebaleds
*****************************************************
Archivos que disponen los manejadores para trabajar
*****************************************************/

const static HttpSpec http_flashspec[] =

{
   { HTTPSPEC_FILE,  "/",             index0_html,    NULL, 0, NULL, NULL},
   { HTTPSPEC_FILE,  "/index0.htm", index0_html, NULL, 0, NULL, NULL},
   { HTTPSPEC_FILE,  "/foto1.jpg",   foto1_jpg,   NULL, 0, NULL, NULL},
```

```
   { HTTPSPEC_FUNCTION, "/pruebaleds.cgi", 0,  pruebaleds, 0, NULL, NULL}
};
*****************************************************
Programa principal
*****************************************************/
void main()
{
        sock_init();
        http_init();
        while(1){
        http_handler();
}
}
```

La conexión física entre el módulo RCM2100 de Rabbit y el kit de PLD se implementa mediante un cable plano, como se observa en la Figura 4.17.

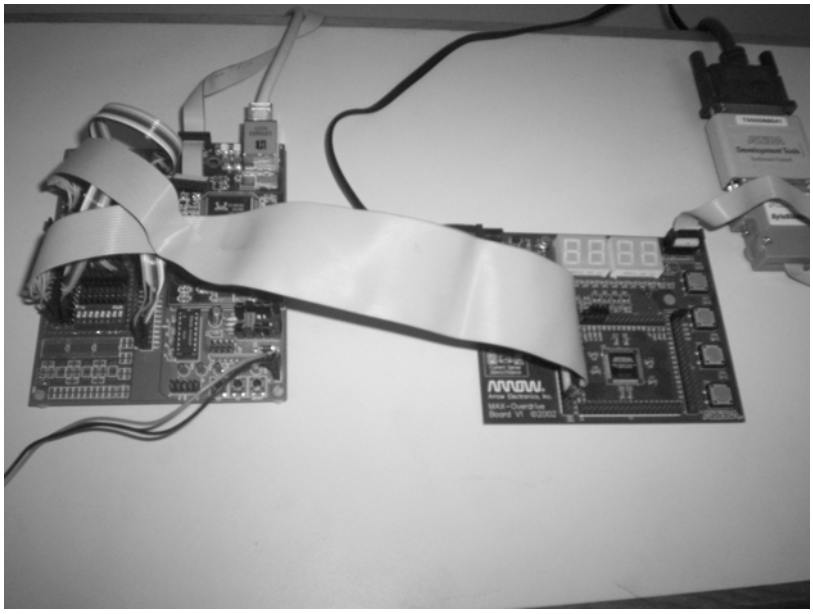


Figura 4.17 - Conexión del módulo RCM2100 con el Kit de PLD

Utilizando las herramientas mencionadas, se ha logrado desarrollar un prototipo funcional de laboratorio remoto para la programación de PLDs con lenguaje VHDL, permitiendo la realización de experiencias básicas de diseño, descripción y verificación de circuitos digitales. Este prototipo se ha implementado en base al modelo conceptual de estructura de capas propuesto

en el Capítulo 3 para el análisis y diseño de laboratorios remotos, modelo que se presenta en Figura 4.18.

Considerando un análisis de abajo hacia arriba, la capa 1 (Recursos físicos, equipamiento de laboratorio) está constituida por el kit de programación de PLD con sus pines de entradas y salidas. La capa 2 (Sistemas de interconexión e interfases) se materializa mediante los puertos del módulo del microcontrolador y los cables que conectan este módulo con las entradas – salidas del Kit de PLD. La programación de los puertos configuran las conexiones que requiere cada experimento. La capa 3 (Control del sistema de interconexión) es responsabilidad del microcontrolador y del software embebido en su memoria. En este caso particular se trata de código C que configura el microcontrolador como un host de la red local y los puertos del mismo en función de la experiencia que se va a realizar. Además de páginas html que invocan a las funciones CGI correspondientes al experimento. Las funciones de la capa 4 (Gestor de configuraciones) las lleva a cabo la aplicación PHP, que le asigna a cada proyecto, es decir a cada experiencia, un archivo de configuración. En esta asignación se referencia a las páginas html y funciones CGI que corresponden a la configuración que requiere la experiencia que se quiere desarrollar.

Los servicios de la capa 5 (Servidor de aplicaciones y acceso a los servicios del laboratorio) la implementan una aplicación PHP que permite el acceso remoto a las utilidades del entorno de desarrollo IDE del Kit de PLD, conjuntamente con un sistema de aula virtual que se encarga del control de acceso, apoyo y seguimiento.

Finalmente la capa 6 (alumno, cliente) satisface el principio de que sólo se necesita para su implementación un navegador en la computadora del cliente para acceder a los recursos del laboratorio.

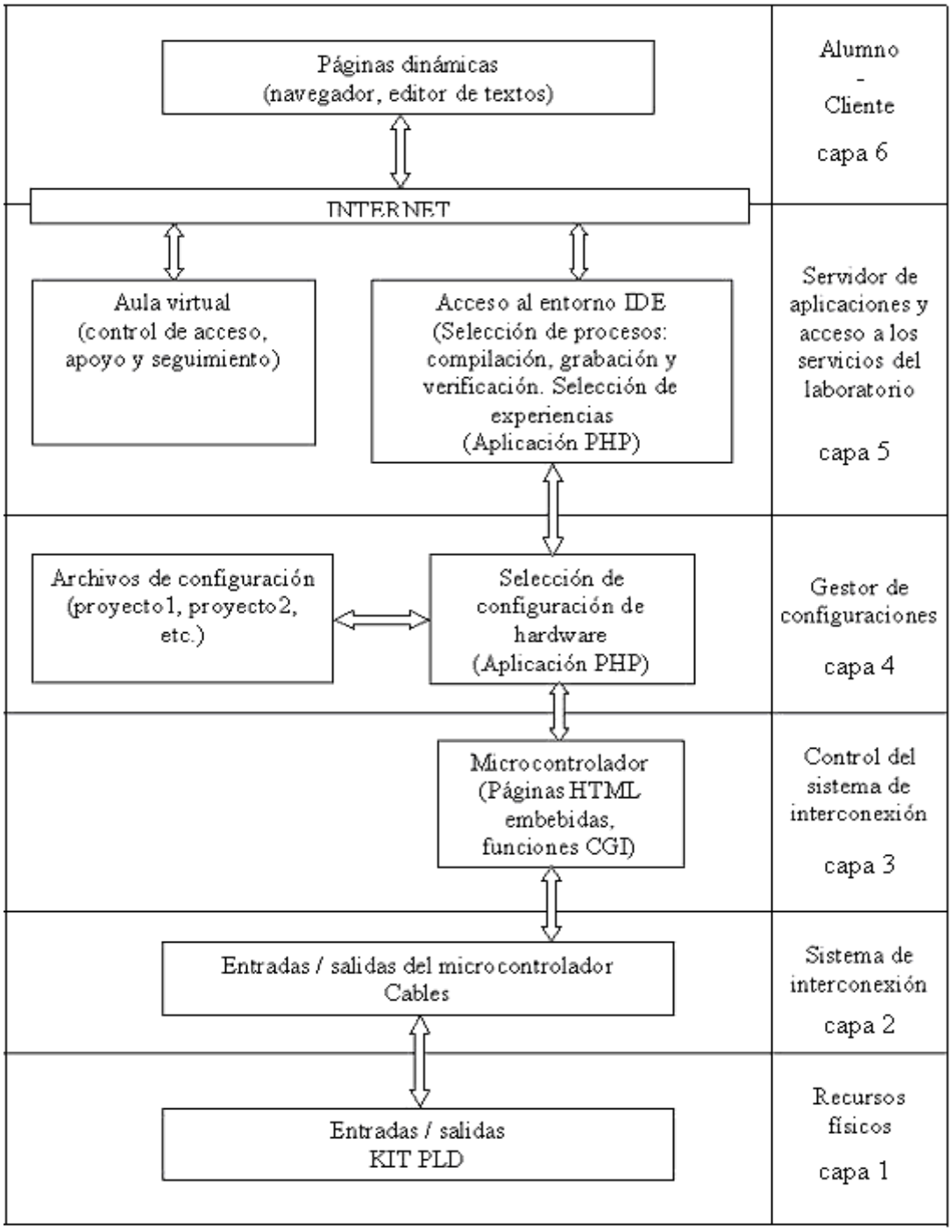


Figura 4.18 - Arquitectura del prototipo de laboratorio remoto.

CAPÍTULO 5

CONCLUSIONES Y TRABAJOS FUTUROS

A partir de la clasificación de los entornos para la realización de prácticas de laboratorio propuesta, se ha planteado un modelo conceptual para el diseño e implementación de laboratorios remotos, que presenta la posibilidad de separar los requerimientos de la arquitectura de estos sistemas en capas o niveles. Este esquema permite una posible generalización de las funciones correspondientes a cada nivel del modelo, de modo que facilita la toma de decisiones en cuanto a las herramientas y los recursos necesarios para la construcción de laboratorios remotos. Esta descripción de la arquitectura de los entornos remotos para las prácticas de laboratorio, es aproximación que permitiría la utilización de módulos preexistentes que implementen algunas de las funciones requeridas, además de facilitar la modificación de módulos y la incorporación de nuevos módulos funcionales. Se logra de esta manera una arquitectura flexible y escalable.

Comparando el laboratorio presencial para la práctica de programación de PLDs con el prototipo desarrollado, se observa que el laboratorio remoto requiere el cableado de la LAN y el microcontrolador, costos relativamente accesibles frente a la ventaja del acceso remoto. Por otra parte el microcontrolador es reutilizable y reconfigurable lo que permite la implementación de prácticas sobre otros kits de desarrollo de PLDs distintos al utilizado en este trabajo. Incluso es factible aplicar el microcontrolador para administrar otros recursos físicos diferentes a PLDs.

Las experiencias realizadas han permitido comprobar la funcionalidad del prototipo. De esta manera se posibilita el acceso remoto, a un costo relativamente bajo, a una herramienta de diseño y programación de PLDs. Finalmente la recuperación de la salida del PLD capturando el estado de las

mismas en un registro de un microcontrolador con Internet embebida constituye una ventaja importante para validar las experiencias remotas.

Trabajos Futuros

En cuanto a la funcionalidad del prototipo implementado, para su puesta en marcha y aprovechamiento integral como recurso didáctico se debe mejorar el acceso a los recursos con un sistema de reservas que complete las posibilidades que brinda el entorno de aula virtual utilizado.

Esta mejora también se relaciona con la incorporación de otros recursos físicos. Es decir que el laboratorio cuente con más de un KIT de desarrollo como recurso local. En el caso de contar con kits replicados (todos iguales) lo que se debe hacer es administrar los accesos a esos kits. La posible solución es generar tablas que contengan el estado de cada kit en cuanto a su utilización y asignación, de modo que la consulta a esas tablas permita la toma de decisiones respecto al acceso remoto a los kits. También se debe implementar un sistema de interconexión para las interfases entre los kits. En este proceso seguramente resulten de mucha utilidad las posibilidades que presentan los microcontroladores en cuanto a la configuración del sistema de interconexión.

Finalmente es necesario considerar también la probabilidad de que se cuente con kits de desarrollos distintos. Si los kits son distintos pero pertenecen al mismo fabricante la situación es similar a la descripta en el párrafo anterior ya que el entorno de desarrollo es el mismo. Pero en el caso de kits de distintos fabricantes el escenario es de mayor complejidad ya que los entornos de desarrollo dependerán de las tecnologías de cada kit de desarrollo. Se debe considerar que la aplicación del servidor tenga en cuenta esta circunstancia para poder acceder a los recursos físicos.

Una última consideración es que los recursos se encuentren distribuidos geográficamente. La arquitectura intrínseca de los sistemas de laboratorios remotos y los recursos utilizados hace posible que este tipo de implementación

se pueda llevar acabo a partir de experiencias como la desarrollada en esta tesis.

Bibliografía – Referencias

[1] Poindexter E., Heck B., “Using the Web in Your Courses: What Can You Do? What Should You Do?”, IEEE Control Systems Magazine, p. 83-92, vol. 19, n°1, Feb. 1999.

[2] Pascoe R., "Introducing WWW Technology into Tertiary Teaching: A Personal Perspective, " North American Web Conference ’97, New Brunswick, Canada, Oct. 1997.

[3] Kapur S., Stillman G., "Teaching and Learning Using the World Wide Web", Innovations in Education and Training International, p. 316-322, vol. 34, n° 4, 1997.

[4] Marianetti O. León O., “Laboratorios Remotos con Internet embebida”, Congreso Internacional Educación Superior y Nuevas Tecnologías, Universidad Nacional del Litoral, Santa Fe, 10 al 12 de agosto de 2005.

[5] Dormido Bencomo S., “Control learning: present and future”, Annual Reviews in Control, p. 115–136, vol. 28, n°1, 2004.

[6] Marianetti O. , León O. , García Garino C., “ Aplicación de Dispositivos con Internet Embebida en Laboratorios Remotos”, V CAEDI 2006, S. Rivera, S. y J. Núñez Mc Leod, (compiladores), Fac. de Ingeniería, UNCuyo y Fac. Regional Mza, UTN, Mendoza, ISBN 987-05-1360-3, p. 1019-1026, vol. 2, 2006.

[7] Grau A., “Laboratorio Remoto de Microprocesadores y Microcontroladores”, XXI Jornadas De Automática F.R. Rubio, p. 190-195, vol.1, 2000.

[8] González I., Gómez F., Martínez J., “LabomatWeb: Recursos Reconfigurables Remotos Vía Word Wide Web”, JCRA 2001, Alicante, p. 102-109, vol. 1, 2001.

[9] Olivares J., Merino A., Palomares J., Montijano M., **"**Laboratorio Virtual Para la Programación de FPGAs", In Proceedings of the VII Simpósio Internacional de Informática Educativa (SIIE05). Leira, Portugal, p. 16-18, vol. 1, Nov. 2005.

[10] Azad A., Otieno A., Ghrayeb O., Anand N., " Internet Based Experiments for Physical Laboratory Set-up", From Proceeding Web-based Education, 2005.
[11] http://www.ni.com/labview.
[12] Hurley W., Kwan Lee C., "Development, Implementation, and Assessment of a Web-Based Power Electronics Laboratory", IEEE Transactions On Education, p. 567-576, vol. 48, n° 4, Nov. 2005.
[13] Sivakumar S., Robertson W., Artimy M., Aslam N., "A Web-Based Remote Interactive Laboratory for Internetworking Education" IEEE Transactions On Education, p. 531-546, vol. 48, n° 4, Nov. 2005.
[14] Moreno J., Berenguel M., Rodríguez F., Sarabia J.F., Garrote R., Guzmán J.L., López O., "Proyecto De Aplicación De Telerobótica A Un Minirobot Móvil" XXIV Jornadas De Automática, León, Sep. 2003.
[15] Bagnasco E., Scapolla A., "A Grid of Remote Laboratory for Teaching Electronics", 2nd International LeGE-WG Workshop on e-Learning and Grid Technologies: A Fundamental Challenge for Europe, P. Ritrovano et al. (Eds), BCS, París, Francia, 3 y 4 de Marzo de 2003.
[16] Dormido S., "Laboratorios virtuales y remotos de control automático: Análisis, diseño y desarrollo", Proyecto DPI 2001-1012, Valencia, 2004.
[17] Coelho P., Oliveira R., "Arquitectura do laboratorio remoto e-CASSIOPEIA", DEEC, Portugal, 2003.
[18] Marianetti O., León O., García Garino C., "Laboratorio Remoto Para la Programación de Dispositivos Lógicos Programables", 7° Simposio Argentino de Tecnología de Computación 2006, Javier Orozco y Ricardo Cayssials, (compiladores), p. 234-243, vol. 1, n°. 21, ISSN 1850 2806, 35º Jornadas Argentinas de Informática e Investigación Operativa ISSN 1850-2776, Mendoza, 2006.

[19] http://www.claroline.net.
[20] Toral Marín S. L., Barrero García F. J., Martínez Torres R., Gallardo Vázquez S., Lillo Moreno A. J., “Implementation of a Web-Based Educational Tool for Digital Signal Processing Teaching Using the Technological Acceptance Model”, IEEE Transactions On Education, p. 467-479, vol. 48, n°. 4, Nov. 2005.
[21] Pérez López S., Soto Campos E., Fernández Gómez S. “Diseño de Sistemas Digitales con VHDL”, Thomson, ISBN 84-9732-081-6, 2002
[22] Nelson V. P., Troy Tagle H., Carroll H. D., Irwin J. D., “Análisis y Diseño de Circuitos Lógicos Digitales”, Prentice Hall, 1996
[23] E. Mandado, L. Alvarez, M. Valdés. “ Dispositivos Lógicos Programables”. Thomsom, ISBN 84-9732-054-9. 2002.
[24] Arrow Electronics Inc. “Development Board, User Guide Version 1.1”, Arrow Electronics Inc 2003.
[25] http://www.altera.com/literature/lit-inde.html.
[26] Altera Corporation, “Max Plus II, Programmable Logic Development System, Getting Started”, Altera Corporation, Sep. 1997.
[27] Altera Corporation, “Introduccion to Quartus II”, Altera Corporation, Jun. 2003.
[28] http://www.phpbuilder.com/manual2/es.
[29] Willard W., “Fundamentos de Programación en HTML”, McGrawHill, ISBN 958-41-0266-4, 2001.
[30] RabbitCore Modules, NetworkingSolutions TCP/ip, Documentation on line, Rabbit Z-World.
[31] Dynamic C Funtion Reference, Rabbit Z-World, http://www.zworld.com
[32]] C. Tavernier, “Circuitos Lógicos Programables”. Editorial Paraninfo, ISBN 2-10-001117-0, 1999.
[33] http://www. latticesemi.com.

[34] Advanced Micro Device, Inc, 1993.

[35] http://www.altera.com/literature.

[36] F. Pardo, J. Boluda, “VHDL Lenguaje para síntesis y modelado de circuitos”, Alfaomega, ISBN 970-15-0443-7, 2000.

APÉNDICE A

Microcontroladores

A.1.1 Introducción a los microcontroladores. Evolución histórica

La siguiente es una lista cronológica de los eventos tecnológicos que han tenido impacto sobre la aparición y el desarrollo del campo de los microcontroladores en la electrónica digital.

En 1971 Intel fabrica el primer microprocesador (el 4004) de tecnología PMOS. Este era un microprocesador de 4 bits y fue fabricado por Intel a petición de Datapoint Corporation con el objeto de sustituir la CPU de terminales inteligentes fabricadas en esa fecha por Datapoint mediante circuitería discreta. El dispositivo fabricado por Intel resultó 10 veces más lento de lo requerido y Datapoint no lo compró, de esta manera Intel comenzó a comercializarlo. El 4004 podía direccionar sólo 4096 (4kB) localidades de memoria de 4 bits, reconocía 45 instrucciones y podía ejecutar una instrucción en 20 μseg en promedio.

En 1972 las aplicaciones del 4004 estaban muy limitadas por su reducida capacidad y rápidamente Intel desarrolló una versión más poderosa (el 8008), el cual podía manipular bytes completos, por lo cual fue un microprocesador de 8 bits. La memoria que este podía manejar se incrementó a 16 kbytes, sin embargo, la velocidad de operación continuó igual.

Para 1973 Intel lanza al mercado el 8080 el primer microprocesador de tecnología NMOS, lo cual permite superar la velocidad de su predecesor (el 8008) por un factor de diez, además se incrementó la capacidad de direccionamiento de memoria a 64 kbytes. A partir del 8080 de Intel se produjo una revolución en el diseño de microcomputadoras y varias compañías fabricantes de circuitos integrados comenzaron a producir microprocesadores. Algunos ejemplos de los primeros microprocesadores son: el IMP-4 y el SC/MP

de National Semiconductors, el PPS-4 y PPS-8 de Rockwell International, el MC6800 de Motorola, el F-8 de Fairchild.

En 1975 Zilog lanza al mercado el Z80, uno de los microprocesadores de 8 bits más poderosos. En ese mismo año, Motorola baja dramáticamente los costos con sus microprocesadores 6501 y 6502 (este último adoptado por APPLE para su primera microcomputadora personal). Estos microprocesadores se comercializan en 20 y 25 dólares respectivamente. Esto provoca un auge en el mercado de microcomputadoras de uso doméstico y un caos en la proliferación de lenguajes, sistemas operativos y programas (ningún producto era compatible con el de otro fabricante).

En 1976 surgen las primeras microcomputadoras de un solo chip, que más tarde se denominarán microcontroladores. Dos de los primeros microcontroladores, son el 8048 de Intel y el 6805R2 de Motorola.

En la década de los 80's comienza la ruptura entre la evolución tecnológica de los microprocesadores y la de los microcontroladores, Ya que los primeros han ido incorporando cada vez más y mejores capacidades par las aplicaciones en donde se requiere el manejo de grandes volúmenes de información y por otro lado, los segundos han incorporado más capacidades que les permiten la interacción con el mundo físico en tiempo real, además de mejores desempeños en ambientes de tipo industrial. Esta ruptura se representa esquemáticamente en la figura A.1.

La diferencia entre el campo de aplicaciones de un microprocesador y un microcontrolador, se presenta en la figura A.2, donde se observa el tipo de aplicaciones como un continuo desde el extremo de los sistemas que manejan grandes volúmenes de información, hasta los que requieren una gran interacción con el mundo físico en tiempo real.

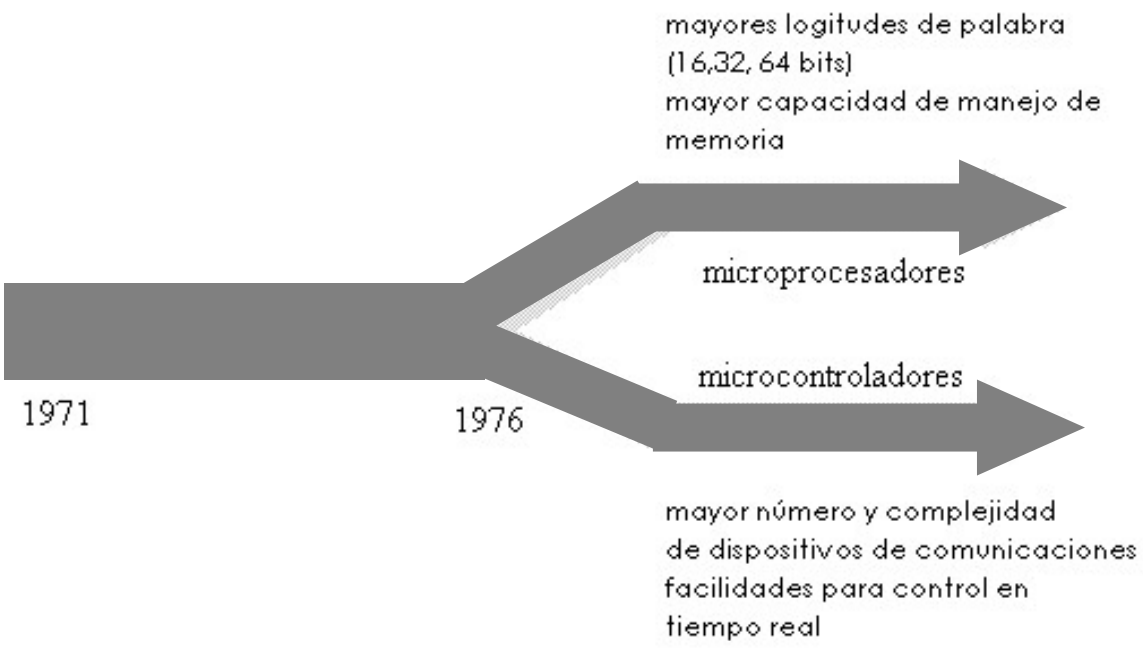


Figura A.1. Microprocesadores y microcontroladores. Evolución

Figura A.2. Microprocesadores y microcontroladores. Aplicaciones

A.1.2. Revisión de conceptos

Revisando algunos conceptos básicos se puede entender de manera más precisa las diferencias entre un microprocesador y un microcontrolador.

a) Unidad Central de Proceso (CPU): Es el "cerebro" de una computadora, de manera más precisa, es la parte de una computadora que se encarga de

ordenar y controlar el proceso y la transferencia de información. La CPU interpreta las instrucciones del programa y coordina su ejecución.

Microprocesador (µp): Es una CPU en un sólo circuito integrado.

Microcomputadora (µc): Es una computadora cuya CPU es un µp.

Microcontrolador (µcc): Es una microcomputadora en un solo circuito integrado.

Computadora: Una computadora es un sistema secuencial síncrono programable, el cual para desempeñar sus funciones debe poseer además de la CPU:

- Canales para el flujo de la información (buses)
- Dispositivos para almacenar información
- Dispositivos para comunicarse con el exterior.

En la figura A.3 se muestra en forma esquemática y muy general la estructura de una computadora.

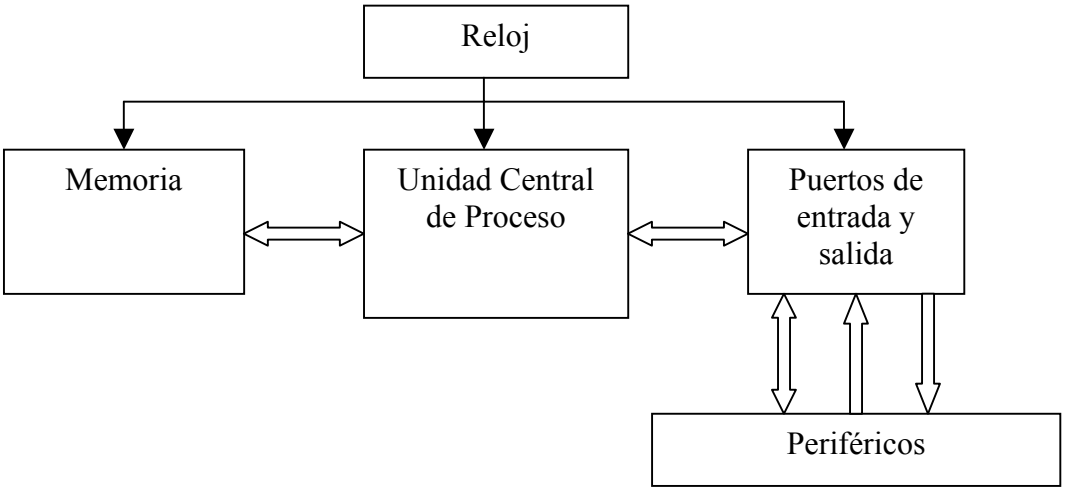


Figura A.3. Estructura básica de una computadora digital

La arquitectura de computadora más usada se denomina Arquitectura de Von Neumann y tiene la característica de poseer un bus común llamado bus de datos para la información sin hacer distinción entre datos e instrucciones.

Una alternativa a la arquitectura clásica de Von Neumann es la arquitectura Hardvard. En esta estructura la memoria de programa (pasiva) recibe un tratamiento diferente que la memoria de datos (activa), pudiéndose llegar a una

total diferenciación entre los buses de datos y de instrucciones. En una arquitectura Hardvard existen bloques de memoria físicamente separados para datos y programas. Cada uno de estos bloques de memoria se direcciona mediante buses separados (tanto de direcciones como de datos), e incluso es posible que la memoria de datos tenga distinto ancho de palabra que la memoria de programa. La Figura A.4. presenta ambas arquitecturas.

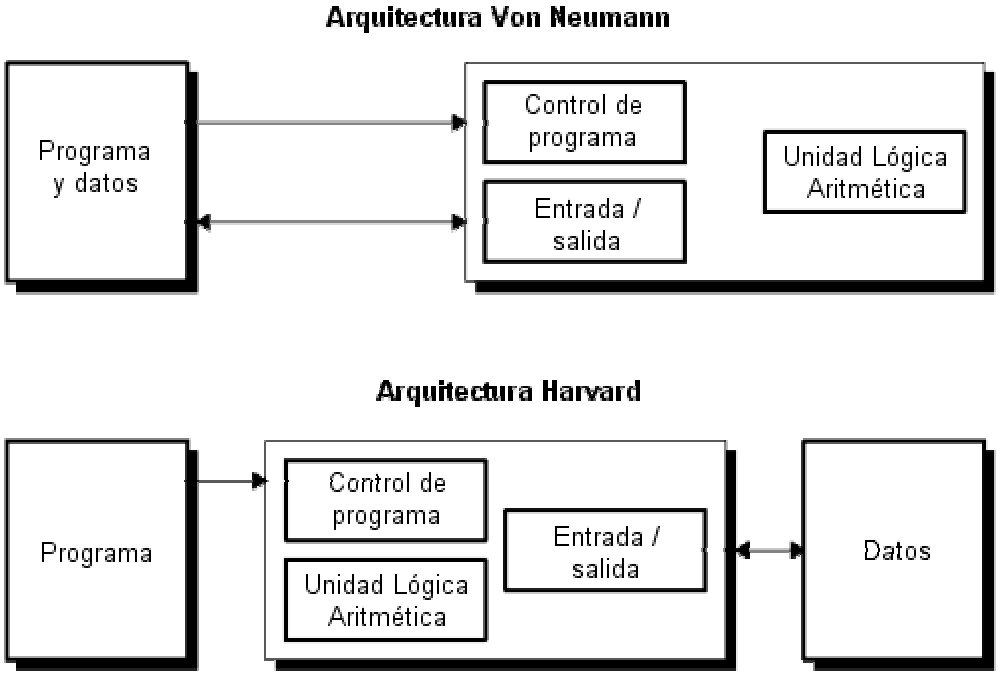


Figura A.4. Arquitecturas de computadoras

Con este diseño se consigue acelerar la ejecución de las instrucciones, ya que el sistema puede ejecutar simultáneamente la lectura de datos de la instrucción “n” y comenzar a decodificar la instrucción “n+1”, disminuyendo el tiempo total de ejecución de cada instrucción. Para ver esto más claro, pensemos en un microprocesador clásico, cuyo ciclo de trabajo es:

- leer la posición de memoria apuntada por el contador de programa
- decodificar la instrucción
- ejecutar la instrucción

En la ejecución de la instrucción se dan estos pasos:

- leer los datos de memoria
- operar con ellos
- dejarlos en la RAM

Por tanto, podemos ver que durante la lectura de los datos de una instrucción el bus está ocupado y no podría ser usado por otra unidad de decodificación, como aparece en muchos microprocesadores modernos, en los que se realizan simultáneamente la ejecución de la instrucción n y la decodificación de la instrucción n+1.

b) RISC y CISC: Los microprocesadores de conjunto de instrucciones reducidos se caracterizan por tener pocas instrucciones muy simples, pero muy eficientes. Tiene un núcleo pequeño y carecen de instrucciones complejas como la multiplicación, manejo de cadenas o loops. Para resolver este tipo de problema se requieren varias instrucciones simples. Además incorporan una gran cantidad de registros. Los microprocesadores de conjunto de instrucciones complejas poseen una gran cantidad de instrucciones, incluso muy complejas, por lo que su núcleo más grande que el de los RISC. También se caracterizan por tener un registro (acumulador) que interviene en la mayoría de las instrucciones aritméticas.

En la actualidad se puede afirmar que gran parte de los microprocesadores comerciales poseen características de ambas tecnologías.

c) Ciclos de espera (wait states): El microprocesador debe acceder a los circuitos integrados de memoria para obtener datos o instrucciones. Debido a los retardos que presentan los circuitos de selección de chip de memoria más el tiempo de acceso requerido por los mismos; puede suceder que el microprocesador, por ser una máquina sincrónica, deba esperar un determinado tiempo la respuesta de los dispositivos de memoria para realizar

un acceso. Ese tiempo se logra insertando ciclos de reloj, que se denominan ciclos de espera.

Para no insertan estos ciclos se puede disminuir la frecuencia del clock. O utilizar memoria internas muy rápida (cache), gran cantidad de registros o coprocesadores que eviten accesos a memorias, favoreciendo una operación a frecuencias elevadas.

d) Ortogonalidad: un microprocesador tiene asignaciones ortogonales cuando la mayoría de los registros se pueden utilizar con la mayoría de los modos de direccionamiento.

e) Ancho de la palabra de procesamiento: La cantidad de líneas del bus de datos se corresponde generalmente con el ancho de la palabra de procesamiento. Este ancho lo suele definir la ALU. Si la ALU es de 8 bits el microprocesador es 8 bits, si es de 16 el micro es de 16 y si 32 o 64 el micro será de 32 o 64 bits. Aunque en algunos casos donde existe diferencias entre el bus de datos interno y el externo debido a los requerimientos de los periféricos, por ejemplo ALUs de 16 bits y registros de controladores de periféricos de 8 bits.

f) Espacio de direccionamiento de memoria: Este espacio es la cantidad máxima de memoria que el microprocesador puede direccionar y en principio está definido por el número de líneas del bus de direcciones. Por ejemplo si el bus de direcciones es de 16 bits, el espacio de direccionamiento de memoria será: 2^{16} = 65536 direcciones = 64 Kbytes

Manejo de memoria: Si las aplicaciones se hacen muy complejas un espacio de memoria de, por ejemplo 64KB, puede resultar muy pequeño. Una solución es agrupar la memoria en bancos y conmutar de banco. Otra solución más elaborada es desarrollar una unidad de manejo de memoria (MMU) que permita acceder a un espacio lógico de memoria distinto del espacio físico de memoria.

g) Direccionamiento físico y direccionamiento lógico: La memoria puede direccionarse en forma:

Lineal

Segmentada

Paginada

Segmentada-paginada

Memoria lineal: En este caso la dirección lógica es igual a la dirección física. La dirección lógica es la dirección que está presente en la palabra instrucción.

Palabra instrucción

Código operativo	Dirección de memoria

Por ejemplo, si en la palabra instrucción, la dirección lógica es 0 2 A 7 (expresada en hexadecimal), (0000 0010 1010 0111, expresada en binario), la dirección física será 0 2 A 7. Este ejemplo se muestra en la Figura A.5.

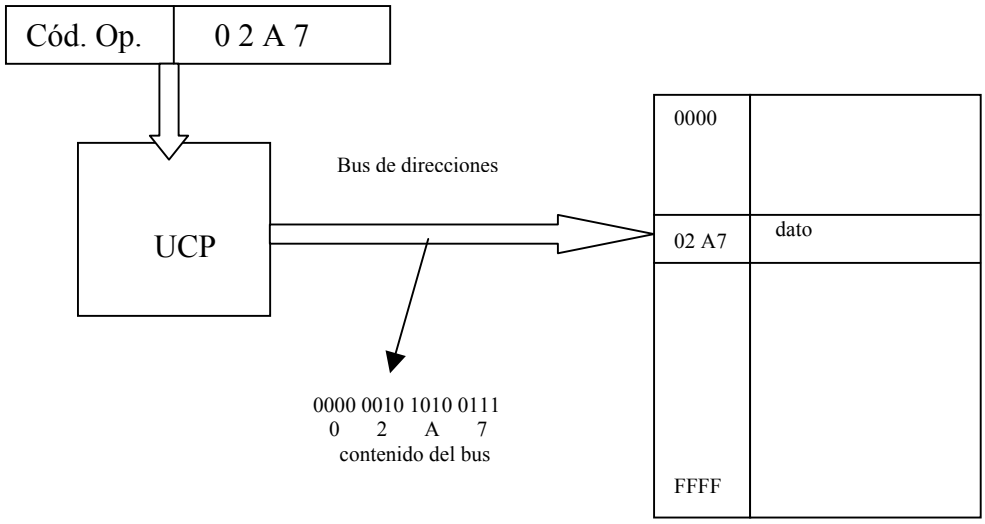


Figura A.5. Direccionamiento lineal

Memoria segmentada

En este caso en la memoria pueden almacenarse segmentos de distintas longitudes. Cada segmento está definido por un descriptor que indica la base del segmento, el límite y los atributos del segmento (si es de código, si de sólo lectura, etc.). Esto permite que algunos segmentos estén presentes en la memoria principal (memoria física) y otros segmentos se encuentren en una

memoria auxiliar (memoria virtual), por ejemplo disco rígido. La Figura A.6 presenta este tipo de direccionamineto.

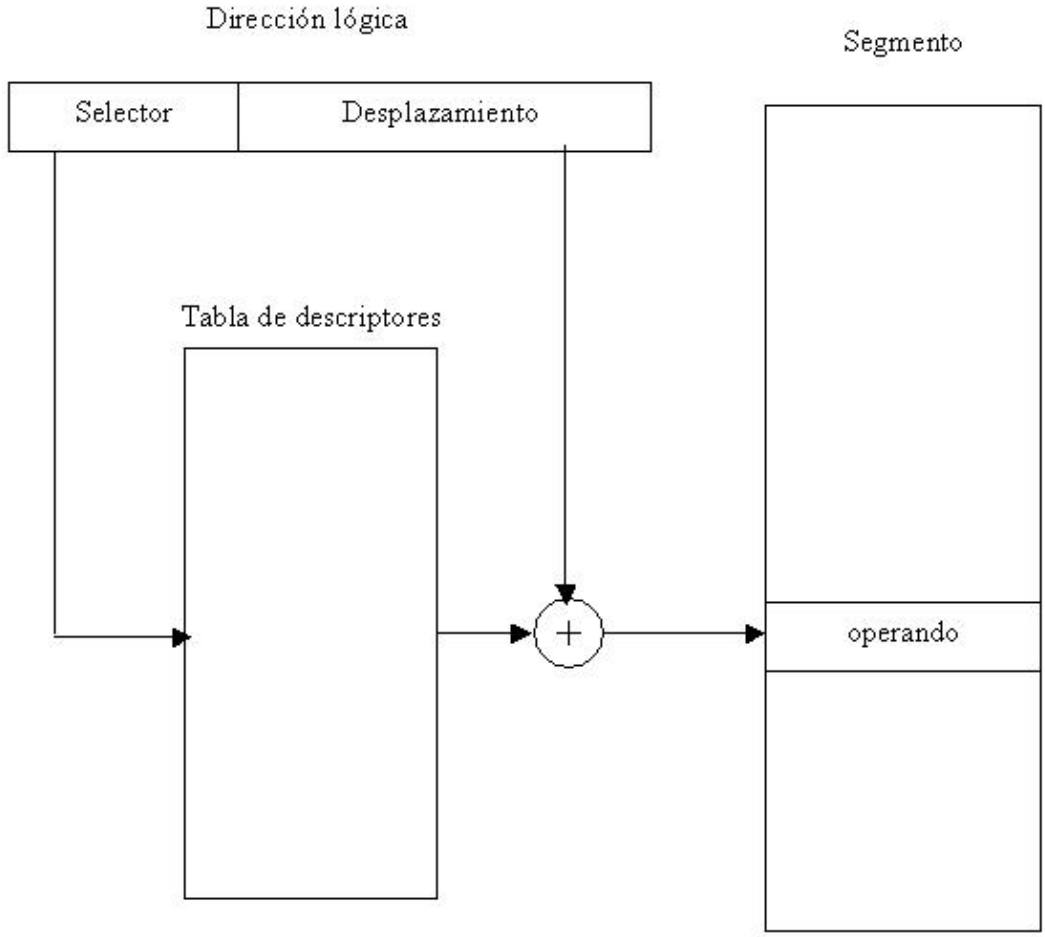


Figura A.6. Direccionamiento segmentado

Este sistema permite que se intercambien segmentos según los requerimientos de la aplicación.

En la Figura A.6 se observa que la dirección lógica no es igual a la dirección física. Se debe aplicar un mecanismo que traduce la dirección lógica a la dirección física. Este mecanismo se representa en la figura, ya que tomando la base del segmento que se encuentra en el descriptor y sumándole el desplazamiento se obtiene una dirección física de 32 bits. Esos 32 bits serán los que estarán presentes en el bus de direcciones.

Podemos observar también que el límite del direccionamiento físiso está dado por los 32 bits del bus de direcciones. (2^{32} = 4 Gbytes) , mientras que el límite del direccionamiento virtual está dado por los 48 bits de la dirección lógica (2^{48} = 64 Tbytes)

Memoria paginada:

Este caso es similar al anterior, salvo que las páginas tienen todas la misma cantidad de bytes.

Memoria segmentada-paginada:

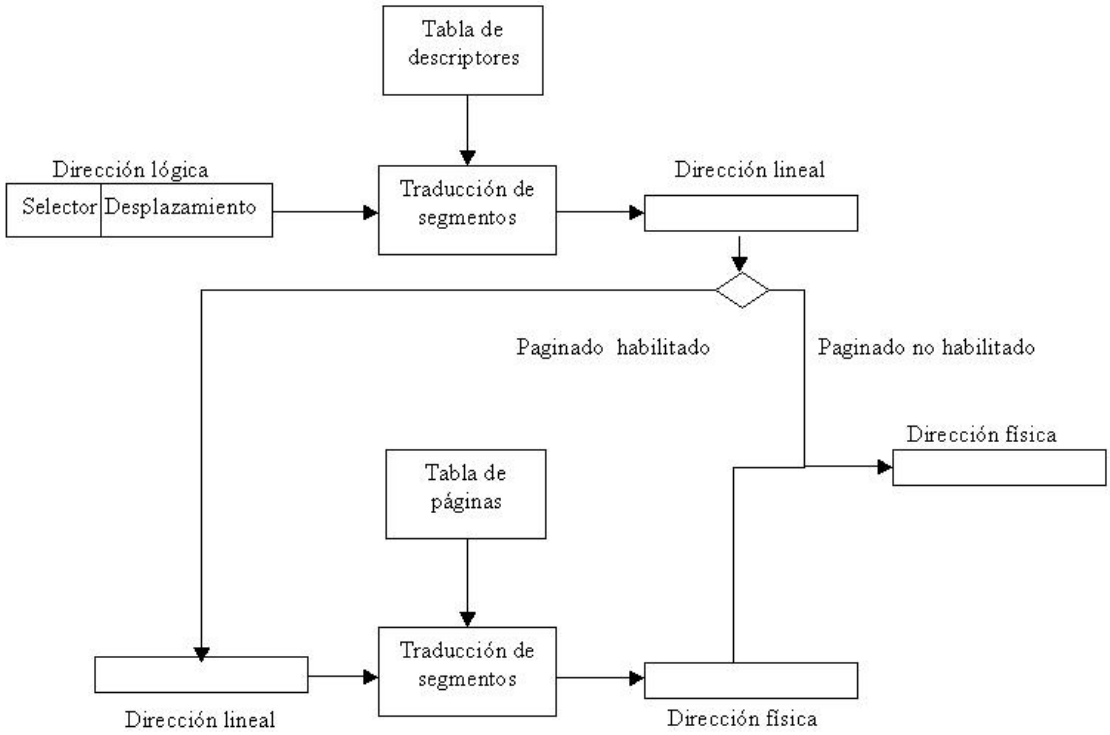


Figura A.7 Direccionamiento segmentado - paginado

Este caso es la combinación de los dos anteriores. En la Figura A.7 se representa los mecanismos de traducción de dirección lógica a dirección física que se implementan. Si no está habilitada la paginación, la dirección lineal de 32 bits es igual a la dirección física. Cuando se habilita la paginación, la dirección lineal de 32 bits se divide en tres campos. Un campo de 10 bits que selecciona una tabla de páginas dentro de un directorio de tablas de páginas.

Una vez seleccionada la tabla; los siguientes 10 bits del campo página se encargan de seleccionar una página. Por último, una vez seleccionada la página, el campo desplazamiento indica la dirección dentro de la página.

h) Interrupciones: Las interrupciones son un mecanismo que permite la conexión del microprocesador con el mundo exterior, sincronizando la ejecución de un programa con operaciones de entrada y salida. Las interrupciones permiten la ejecución de una subrutina cuando se activa una terminal de entrada de la CPU. Es decir que es un mecanismo de hardware. Aunque también puede aparecer una interrupción cuando un registro llega a un valor determinado; en este caso la interrupción es por software.

Vectores en un microprocesador: Se denomina vector a direcciones de memoria (generalmente ROM o EPROM), donde se encuentran almacenados bytes que unidos forman una dirección de memoria a la que el microprocesador salta por si solo y comienza la ejecución de una subrutina de atención asociada al vector. El salto al vector sólo se efectúa por un requerimiento de interrupción o por una inicialización (RESET). Las direcciones de los vectores son fijas para cada microprocesador. El programador puede definir el contenido de estas direcciones.

Interrupciones hardware: Cuando un microprocesador recibe un requerimiento de interrupción, realiza el siguiente proceso:

1- Detiene la ejecución del programa y guarda en la pila la dirección de retorno al programa principal y también puede guardar el valor de algunos registros

2-Carga el contador de programa con la dirección de la subrutina de atención.

3-Al final de la subrutina de atención el microprocesador encuentra una instrucción de retorno. Entonces recupera de la pila la dirección de retorna al programa principal y en algunos casos el valor de algunos registros.

Interrupciones enmascarables: Mediante una línea de entrada, que se puede identificar como INT o IRQ, se produce la solicitud de interrupción. Los microprocesadores suelen tener en el registro de estado un bit que permite

evitar la atención al requerimiento de interrupción. Es decir que , por ejemplo si ese bit, que se conoce como máscara de interrupción esta en 0, se procesa la solicitud de atención. En caso de que el bit de máscara valga 1, la solicitud no se procesa.

Interrupciones no enmascarables: La solicitud que se hace por esta línea de entrada (NMI) es de máxima prioridad, ya que no se puede enmascarar mediante el bit de máscara.

Interrupciones software: Muchos microprocesadores tienen una instrucción (SWI) que detiene la ejecución del programa principal como si fuera una interrupción y actúa de modo similar a la interrupción enmascarable.

Reset: La entrada de reset se emplea en todos los microprocesadores para su inicialización. La activación de esta línea da lugar a una secuencia de operaciones:

1- El microprocesador detiene inmediatamente la ejecución del programa en curso.

2- Seguidamente, pone a cero todos sus registros internos.

3- En el flanco de finalización de la orden de Reset el contador de programa se carga

automáticamente con los valores contenidos en el llamado "vector del Reset".

4- El microprocesador ejecuta la subrutina asociada a la orden de RESET.

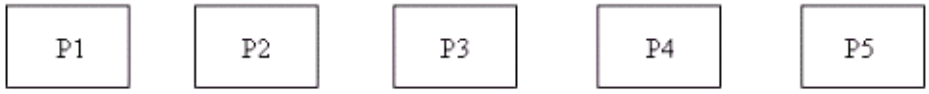
Es de destacar que, como ya dijimos, el vector de RESET es fijo para cada microprocesador, pero la subrutina asociada a él es fijada tanto en longitud como en contenido por el programador que diseña el sistema.

En los sistemas de desarrollo de los microprocesadores. la activación de la entrada RESET suele realizarse de forma automática en el momento de encendido del sistema, si bien el usuario puede inicializarlo en cualquier momento con un pulsador. La orden de reset permite iniciar la ejecución de un programa almacenado en memoria ROM, que se denomina programa de arranque o programa monitor, que contiene todos los datos, tablas y rutinas

básicas del sistema y le permiten al sistema estar en condiciones de ejecutar los programas que el usuario requiera.

i) Pipeline: Una variante de esta idea es la de separar en partes la ejecución de cada instrucción, como el armado de un auto en una línea de ensamble. En la Figura A.8 se observa una CPU formada por cinco unidades de procesamiento de Pl a P5. Durante el primer intervalo de tiempo, P1 extrae de memoria la primera instrucción; en el segundo intervalo, la primera instrucción pasa a P2 para su decodificación, mientras que P1 extrae la siguiente instrucción. En cada uno de los intervalos subsiguientes, P1 extrae una nueva instrucción y las anteriores pasan a la siguiente unidad, a lo largo de la trayectoria.

A la organización de la figura se le podría llamar procesamiento entubado o procesamiento en línea (pipeline). Si cada paso (intervalo de tiempo) dura n nanosegundos, se requiere de 5n nanosegundos para ejecutar una instrucción. Sin embargo, cada n nanosegundos P5 termina de ejecutar una instrucción, lo que incremento la velocidad en un factor de cinco.



P1: Unidad de búsqueda de instrucciones
P2: Unidad de decodificación de instrucciones
P3: Unidad de direccionamiento
P4: Unidad de búsqueda de datos
P5: Unidad de ejecución

P1	1	2	3	4	5	6	7	8	9
P2		1	2	3	4	5	6	7	8
P3			1	2	3	4	5	6	7
P4				1	2	3	4	5	6
P5					1	2	3	4	5

tiempo

Figura A.8. Pipeline

j) Sistemas mínimos basados en microprocesador: Un sistema mínimo basado en µp o µcc es una microcomputadora de propósito específico, equipada con el mínimo de componentes (memoria RAM, ROM, puertos, sensores, actuadores, etc.) para realizar sus funciones.

Los propósitos para los cuales puede diseñarse un sistema mínimo pueden caer en una infinidad de campos, tales como instrumentación, control, monitoreo, señalización, secuenciamiento, autorización, comunicaciones, procesamiento de señales, etc.

A.1.3 Aplicaciones de los microcontroladores.

Las aplicaciones específicas de los microcontroladores son tan enormemente variadas que no se exagera cuando se dice que éstas están limitadas solamente por la imaginación del diseñador.

En cualquier problema en el cual se requiera un instrumento digital compacto que sea capaz de realizar funciones como las siguientes, es posible pensar en sistema basado en un microcontrolador: secuenciamiento, codificación, decodificación, monitoreo, adquisición de datos, señalización, procesamiento de señales, control realimentado, temporización, cálculos aritméticos sencillos, comunicaciones, automatización, visualización digital, control on - off, etc.

A.2.1. Sistemas Embebidos. Concepto y aplicaciones

Un sistema embebido (SE) o empotrado es un sistema basado en microprocesador diseñado específicamente para ser montado como parte integrante en equipos industriales de instrumentación, automatización, producción, en vehículos para transporte terrestre, marítimo y aéreo y también en equipos dedicados al sector de consumo tales como electrodomésticos, equipos multimedia, juguetes, etc.

A diferencia de la típica computadora personal (PC), un sistema embebido, es un sistema que busca un costo mínimo mediante el diseño de sus módulos adecuándolos a las necesidades requeridas por el sistema principal donde va a ser colocado. Ello implica la eliminación de varios de los elementos que existen

en un PC, que no serán necesarios en el sistema principal de la CPU embebida. La diferencia más frecuente es la menor potencia de procesamiento que suelen requerir las aplicaciones de sistemas embebidos con respecto a las de un PC convencional, así como la de menor resolución gráfica y posibilidades de ampliación con nuevos módulos ya que el sistema principal será diseñado en su totalidad para unos requisitos específicos. Si el sistema principal queda obsoleto será no sólo por la CPU embebida sino también por el resto de los elementos que lo integran, con lo que la solución viene dada por el rediseño del sistema completo en la mayoría de los casos. Además, en muchas situaciones el tamaño requerido del sistema embebido es cada vez más reducido.

Con respecto a la eliminación de ciertos módulos o capacidades en un sistema embebido con respecto al PC convencional conviene señalar que para ciertas aplicaciones puede ser necesario, por el contrario, incluir alguna capacidad específica en el sistema embebido que no aparezca en un PC convencional.

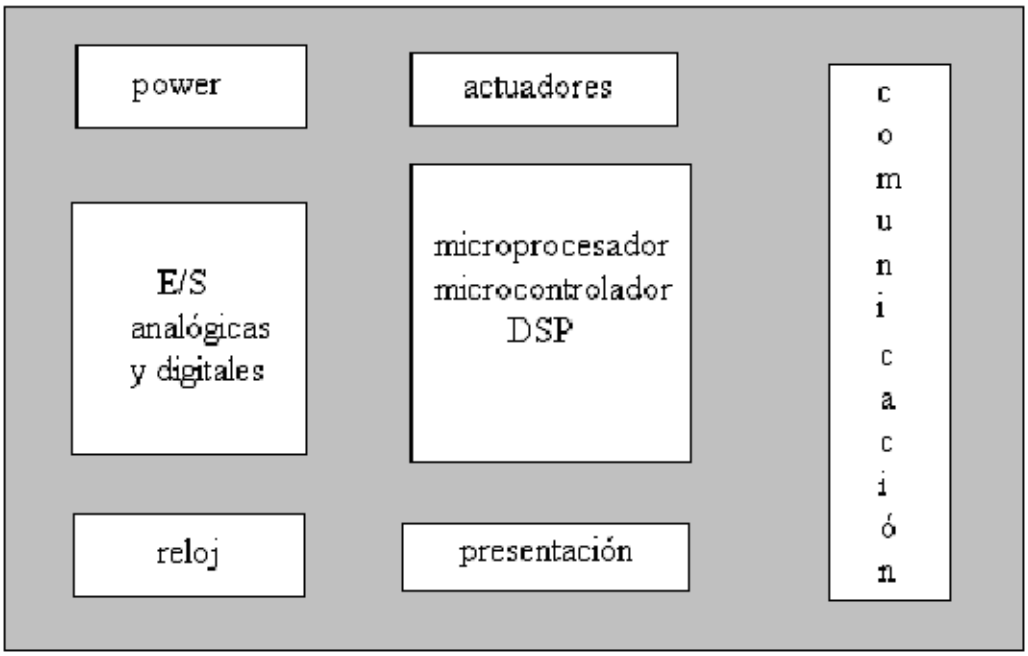


Figura A.9. Sistema embebido

En la Figura A.9 se muestra lo que puede ser un modelo general de un sistema embebido.

En la parte central se encuentra el microprocesador, microcontrolador, dsp, etc. Es decir la CPU o unidad que aporta inteligencia al sistema. Según el sistema puede incluir memoria interna o externa, un micro con arquitectura especifica según requisitos. La oferta es elevada y se requiere de una fase de estudio inicial para seleccionar el mas adecuado a cada aplicación.

Lo normal es que el SE pueda comunicarse mediante interfaces estándar de comunicaciones por cable o inalámbricas. Así un SE normalmente incorporará puertos de comunicaciones del tipo RS232, RS485, SPI, CAN, USB, IP, WiFi, GSM, GPRS, etc.

El subsistema de presentación típico suele ser una pantalla gráfica, táctil, LCD alfanumérico, etc.

Los actuadores son los posibles elementos electrónicos que supuestamente el sistema se encarga de controlar. Puede ser un motor eléctrico, un conmutador tipo relé, etc. El mas habitual puede ser una salida de señal PWM para control de velocidad de motores de corriente continua.

El módulo de E/S analógicas y digitales suele emplearse para digitalizar señales analógicas procedentes de sensores, activar diodos leds, reconocer el estado abierto cerrado de un conmutador o pulsador, etc.

El modulo de reloj es el encargado de generar las diferentes señales de reloj a partir de un único oscilador principal. El tipo de oscilador es importante por varios aspectos, por la frecuencia necesaria, por la estabilidad necesaria y por el consumo de corriente requerido. Los oscilador con mejores características en cuanto a estabilidad y costo son los basados en resonador de cristal de cuarzo, mientras que los que requieren menor consumo son los RC. Mediante sistemas PLL (phase loocked loop) se obtienen otras frecuencias con la misma estabilidad que el oscilador patrón.

El módulo de energía (power) se encarga de generar las diferentes tensiones y corrientes necesarias para alimentar los diferentes circuitos del SE. Mediante conversores ac/dc o dc/dc se obtienen las diferentes tensiones necesarias para alimentar los diversos componentes activos del circuito. Además pueden utilizarse otros módulos como filtros, circuitos integrados supervisores de alimentación, etc.

El consumo de energía puede ser determinante en el desarrollo de algunos SE, que necesariamente se alimentan con baterías.

A.2.2. Microprocesadores y sistemas embebidos

Frecuentemente nos referimos a un microprocesador como simplemente "CPU", y la parte de un sistema que contiene al microprocesador se denomina subsistema de CPU.

Los microprocesadores varían en consumo de potencia, complejidad y costo. Los hay de unos pocos miles de transistores y con costo inferior a 2 dólares (en producción masiva) hasta de más de once millones de transistores que cuestan más de 600 dólares.

Los subsistemas de entrada, salida y memoria pueden ser combinados con un subsistema de CPU para formar una computadora o sistema embebido completo. Estos subsistemas se interconectan mediante los buses de sistema (formados a su vez por el bus de control, el bus de direcciones y el bus de datos).

El subsistema de entrada acepta datos del exterior para ser procesados mientras que el subsistema de salida transfiere los resultados hacia el exterior. Lo más habitual es que haya varios subsistemas de entrada y varios de salida. A estos subsistemas se les reconoce habitualmente como periféricos de E/S.

El subsistema de memoria almacena las instrucciones que controlan el funcionamiento del sistema. Estas instrucciones comprenden el programa que ejecuta el sistema. La memoria también almacena varios tipos de datos: datos

de entrada que aún no han sido procesados, resultados intermedios del procesado y resultados finales en espera de salida al exterior.

Un microcontrolador (MCU) es un CI que incluye una CPU, memoria y circuitos de E/S. Entre los subsistemas de E/S que incluyen los microcontroladores se encuentran los temporizadores, los convertidores analógico a digital (ADC) y digital a analógico (DAC) y los canales de comunicaciones serie. Estos subsistemas de E/S se suelen optimizar para aplicaciones específicas (por ejemplo audio, video, procesos industriales, comunicaciones, etc.).

Hay que señalar que las líneas reales de distinción entre microprocesador, microcontrolador y microcomputadora en un solo chip están difusas y en ocasiones se denominan de manera indistinta unos y otros.

En general, un SE consiste en un sistema con microprocesador cuyo hardware y software están específicamente diseñados y optimizados para resolver un problema concreto eficientemente. Normalmente un SE interactúa continuamente con el entorno para vigilar o controlar algún proceso mediante una serie de sensores. Su hardware se diseña normalmente a nivel de chips, o de interconexión de PCBs, buscando la mínima circuitería y el menor tamaño para una aplicación particular. Otra alternativa consiste en el diseño a nivel de PCBs consistente en el ensamblado de placas con microprocesadores comerciales que responden normalmente a un estándar (placas de tamaño concreto que se interconectan entre sí “apilándolas” unas sobre otras, cada una de ellas con una funcionalidad específica dentro del objetivo global que tenga el SE). Esta última solución acelera el tiempo de diseño pero no optimiza ni el tamaño del sistema ni el número de componentes utilizados ni el costo unitario.

En general, un sistema embebido simple contará con un microprocesador, memoria, unos pocos periféricos de E/S y un programa dedicado a una aplicación concreta almacenado permanentemente en la memoria. El término embebido o empotrado hace referencia al hecho de que la microcomputadora está encerrada o instalada dentro de un sistema mayor y su existencia como

microcomputadora puede no ser aparente. Un usuario no técnico de un sistema embebido puede no ser consciente de que está usando un sistema que incluye una computadora. En algunos hogares las personas, que no tienen por qué ser usuarias de una computadora personal estándar (PC), utilizan del orden de diez o más sistemas embebidos cada día.

A.2.3. Aplicaciones de un sistema embebido.

Las microcomputadoras embebidas en estos sistemas controlan electrodomésticos tales como: televisores, videos, lavadoras, alarmas, teléfonos inalámbricos, etc. Incluso una PC tiene microcomputadores embebidas en el monitor, impresora, y periféricos en general, adicionales a la CPU de la propia PC. Un automóvil puede tener hasta un centenar de microprocesadores y microcontroladores que controlan cosas como la ignición, transmisión, dirección asistida, frenos antibloqueo (ABS), control de la tracción, etc. Los sistemas embebidos se caracterizan normalmente por la necesidad de dispositivos de E/S especiales. Cuando se opta por diseñar el sistema embebido partiendo de una placa con microcomputador también es necesario comprar o diseñar placas de E/S adicionales para cumplir con los requisitos de la aplicación concreta.

Muchos sistemas embebidos son sistemas de tiempo real. Un sistema de tiempo real debe responder, dentro de un intervalo restringido de tiempo, a eventos externos mediante la ejecución de la tarea asociada con cada evento. Los sistemas de tiempo real se pueden caracterizar como blandos o duros. Si un sistema de tiempo real blando no cumple con sus restricciones de tiempo, simplemente se degrada el rendimiento del sistema, pero si el sistema es de tiempo real duro y no cumple con sus restricciones de tiempo, el sistema fallará. Este fallo puede tener posiblemente consecuencias catastróficas.

Un sistema embebido complejo puede utilizar un sistema operativo como apoyo para la ejecución de sus programas, sobre todo cuando se requiere la ejecución simultánea de los mismos. Cuando se utiliza un sistema operativo lo

más probable es que se tenga que tratar de un sistema operativo en tiempo real (RTOS), que es un sistema operativo diseñado y optimizado para manejar fuertes restricciones de tiempo asociadas con eventos en aplicaciones de tiempo real. En una aplicación de tiempo real compleja la utilización de un RTOS multitarea puede simplificar el desarrollo del software.

Los lugares donde se pueden encontrar los sistemas embebidos son numerosos y de varias naturalezas. A continuación se exponen varios ejemplos para ilustrar las posibilidades de los mismos:

• En una fábrica, para controlar un proceso de montaje o producción. Una máquina que se encargue de una determinada tarea hoy en día contiene numerosos circuitos electrónicos y eléctricos para el control de motores, hornos, etc. que deben ser gobernados por un procesador, el cual ofrece un interfaz persona – máquina para ser dirigido por un operario e informarle al mismo de la marcha del proceso.

• Puntos de servicio o venta (POS, Point Of Service). Las cajas donde se paga la compra en un supermercado son cada vez más completas, integrando teclados numéricos, lectores de códigos de barras mediante láser, lectores de tarjetas bancarias de banda magnética o chip, pantalla alfanumérica de cristal líquido, etc. La PC embebida en este caso requiere numerosos conectores de entrada y salida y unas características robustas para la operación continuada.

• Puntos de información al ciudadano. En oficinas de turismo, grandes almacenes, bibliotecas, etc. existen equipos con una pantalla táctil donde se puede pulsar sobre la misma y elegir la consulta a realizar, obteniendo una respuesta personalizada en un entorno gráfico amigable.

• Decodificadores y set-top boxes para la recepción de televisión. Cada vez existe un mayor número de operadores de televisión que aprovechando las tecnologías vía satélite y de red de cable ofrecen un servicio de televisión de pago diferenciado del convencional. En primer lugar envían la señal en formato digital MPEG-2 con lo que es necesario un procesado para decodificarla y

mandarla al televisor. Además viaja cifrada para evitar que la reciban en claro usuarios sin contrato, lo que requiere descifrarla en casa del abonado. También ofrecen un servicio de televisión interactiva o web-TV que necesita de un software específico para mostrar páginas web y con ello un sistema basado en procesador con salida de señal de televisión.

• Sistemas radar de aviones. El procesado de la señal recibida o reflejada del sistema radar embarcado en un avión requiere alta potencia de cálculo además de ocupar poco espacio, pesar poco y soportar condiciones extremas de funcionamiento (temperatura, presión atmosférica, vibraciones, etc.).

• Equipos de medicina en hospitales y ambulancias UVI – móvil.

• Máquinas de revelado automático de fotos.

• Cajeros automáticos.

• Gateways Internet-LAN.

• Y un sin fin de posibilidades aún por descubrir o en estado embrionario como son las heladeras inteligentes que controlen su suministro vía Internet, PCs de bolsillo, etc.

Una especificación interesante que presentan algunos microcontroladores es disponer de funciones de Internet embebida. La tabla A.1 compara placas de microcontroladores con conexión a red de área local (LAN Ethernet) disponibles comercialmente.

	Ethrrnut 2.1	Picoweb	Rabbit	Snijder	UCsimm
Procesador	Atmega 103	AT90S8515	Rabbit core	ARM7TDM	68az328
Puertos	RS232	RS232	RS232 RS485	RS232 RS485 I^2C VGA LCD	RS232 I^2C SPI
Lan	10/100 base T	10 base T	10/100 base T	10/100 base T	10 base T
E/S digitales	24	16	32	16	21
Memoria	128K FLASH 4K EEPROM 32K RAM	8K FLASH 512K EEPROM 512K RAM	512K FLASH 256K RAM	8M FLASH 8M RAM	2M FLASH 8M RAM
Protocolos	TCP/IP HTTP	TCP/IP HTTP	TCP/IP HTTP SNMP FTP TELNET POP3	TCP/IP HTTP SNMP FTP TELNET POP3	TCP/IP HTTP
Programación	C	ASEMBLER	C	JAVA	C
Precio	232	226	300	250	240

Tabla A.1. Microcontroladores con Internet embebida

Estos sistemas utilizan diverso controladores Ethernet, de forma transparente para el usuario, ya que todo el proceso de configuración y operación se lleva a cabo mediante una serie de bibliotecas, de modo que no se deba actuar directamente con el hardware.

Dando un vistazo a esta tabla y visitando las páginas web de los fabricantes se puede ver una gran variedad de capacidades y precios de las placas microcontroladoras. Mientras muchos de estos dispositivos son, simplemente, servidores de páginas web embebidos, otros son un microcontrolador que tiene un gran potencial para una gran variedad de aplicaciones y además que se programa en un lenguaje de alto nivel.

APÉNDICE B

Dispositivos Lógicos Programables.

B.1 Introducción

El incremento de popularidad y de utilización de los dispositivos lógicos programables o PLDs está siguiendo un proceso solamente comparable al que hace algunos años acompañó a los microprocesadores. Los PLDs se utilizan en casi todos los nuevos equipos electrónicos de control, industriales, de consumo, de oficina, de comunicaciones, etc.

Desde finales de la década de los sesenta, los equipos electrónicos digitales se han construido utilizando circuitos integrados de función lógica fija, realizados en pequeña o mediana escala de integración. Para las realizaciones muy complejas que exigirían un número elevado de circuitos integrados (CI) de función fija, se utilizan circuitos diseñados a medida que sólo sirven para una aplicación. Son los llamados CI específicos a una aplicación o ASIC (Application Specific Integrated Circuit). Por regla general, los ASICs los producen los fabricantes de CI con las especificaciones proporcionadas por el usuario.

Los equipos realizados con ASICs ocupan menos espacio, son más fiables, consumen menos energía y en grandes series resultan más baratos que los equipos equivalentes realizados con CI de función fija. Por otro lado, estos circuitos son muy difíciles de copiar.

Diferentes modalidades de ASICs son; los Circuitos a Medida (Full Custom), las Matrices de Puertas (Gate Arrays), las Células Normalizadas (Standard Cell) y los FPICs (Field Programmable Integrated Circuits); estos últimos son circuitos programables por el usuario final.

B.1.2 Circuitos integrados a medida

Los Circuitos Integrados a Medida (Full Custom), se diseñan a petición de un cliente para que resuelvan una determinada aplicación. Conllevan un alto costo de desarrollo y su empleo sólo se justifica para volúmenes de producción muy elevados. El tiempo necesario para la construcción de un CI a medida es considerable ya que puede oscilar de unos meses a unos años.

B.1.3 Matrices de puertas

Las Matrices de puertas (Gate Arrays) son pequeños trozos de silicio que dependen de algún proceso de metalización que defina las conexiones entre un importante número de puertas o transistores que poseen en su interior. Las matrices de puertas proporcionan densidades superiores a las 100.000 puertas, con un aprovechamiento del 80 al 90 por 100 para los dispositivos pequeños y del 40 por 100 para los grandes.

Los fabricantes de silicio ponen a disposición de sus potenciales clientes abundante documentación sobre estos Gate Arrays, con una serie de macros que pueden utilizar de forma inmediata y otras que pueden construirse ellos mismos. Los macros son agrupaciones de un número de células básicas que realizan funciones comunes como; sumadores; puertas NOT, AND, NAND, NOR, XOR, etc; latches y flip-flops S-R, J-K, D; buffer; osciladores; registros, decodificadores, multiplexores, etc.

Junto a esta documentación, los fabricantes aportan un software que contabiliza el número de células básicas utilizadas por todas las macros, sugiere el Gate Array adecuado para la aplicación, calcula la potencia disipada por el Gate Array que alojará el diseño del cliente, proporciona información sobre los tiempos de propagación de las señales y permite verificar el funcionamiento del circuito.

Una vez superadas todas las etapas previas, el cliente envía la documentación generada al fabricante para que éste ultime los procesos de metalización y fabrique un primer prototipo. El diseño con Gate Arrays puede durar semanas o meses. Requiere un volumen alto de circuitos para justificar sus costos.

B.1.4 Células normalizadas

Las células normalizadas (Standard Cell) son, en cierta forma, similares a las matrices de puertas. Su principal ventaja sobre ellas es que en lugar de trabajar con simples puertas o transistores, se dispone de colecciones de diferentes partes de circuitos que han sido depurados (puertas lógicas, circuitos MSI, RAM estáticas, archivo de registro, etcétera). El usuario tiene que ensamblar estos circuitos, verificarlos y finalmente enviar documentación al fabricante de silicio para el desarrollo del primer prototipo. A pesar del concepto de célula normalizada, los períodos y los costes de desarrollo son superiores a los de las matrices de puertas.

En las matrices de puertas sólo hay que realizar la máscara final que define las conexiones entre las puertas, mientras que en las células normalizadas, hay que realizar máscaras para todos los procesos de producción de los CI. Una vez más, el volumen de fabricación deberá ser los suficientemente alto como para amortizar la inversión económica realizada en el desarrollo.

B.1.5 FPICs. (Field Programmable Integrated Circuits)

Son chips programables por el usuario mediante programadores comerciales. El término FPIC también incluye a los CI no destinados a las aplicaciones lógicas. Son las memorias, los microcontroladores, los PLD (Programmable Logic Device), las FPGA (Field Programmable Gate Array) y los ASPLD (Aplication Specific Programmable Logic Devices).

Los FPIC ofrecen soluciones de bajo costo, de tiempo de desarrollo corto y con menor riesgo que los circuitos a medida, las matrices de puertas y las células normalizadas.

Los PLDs (Programmable Logic Devices) son ASICs configurables por el usuario capaces de realizar una determinada función lógica. La mayoría de los PLD consisten en una matriz de puertas AND seguida de otra matriz de puertas OR. Mediante esta estructura, puede realizarse cualquier función como suma de términos productos.

Aunque las memorias PROM, EPROM y EEPROM son PLDs, muchas veces se las excluye de esta denominación debido a que su contenido se define utilizando elementos de desarrollo propios de microprocesadores, tales como; ensambladores, emuladores y lenguajes de programación de alto nivel. Otras veces, cuando estas memorias se usan para realizar una función lógica y no para guardar un programa de un microprocesador, se las incluye dentro del término PLD. Una clasificación de los FPICs se presenta en la Figura B.1.

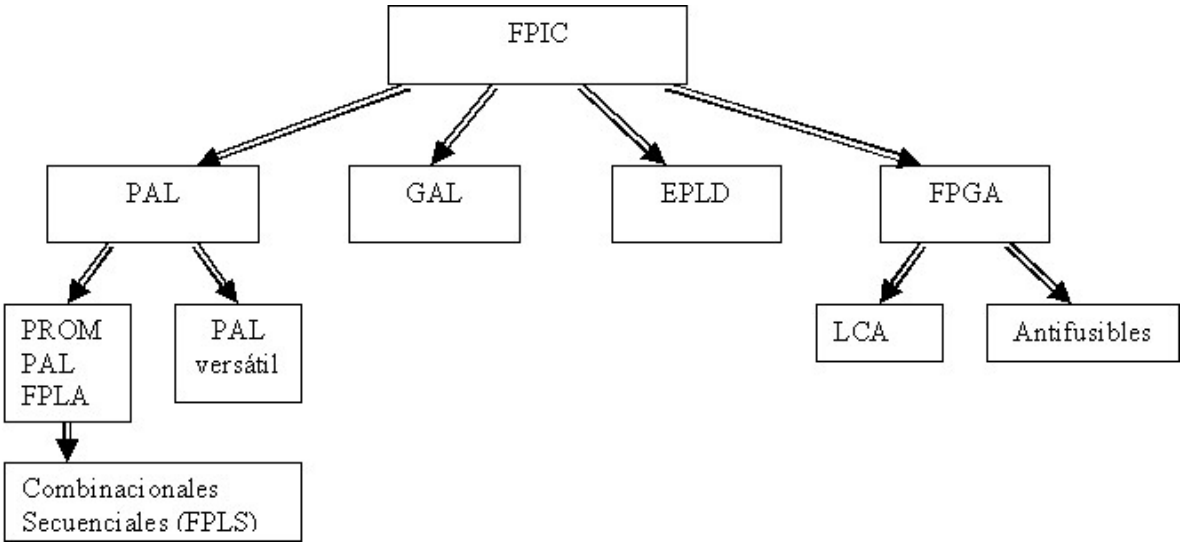


Figura B.1. Clasificación de los FPICs.

B.2 Estructura básica de un PAL

Un PAL es un bloque funcional que se utiliza para implementar multifunciones booleanas. Existe una gran relación entre su estructura interna y el conjunto de funciones que realiza.

Está constituido básicamente por dos submatrices o planos denominados plano AND y OR, respectivamente. Ambos planos están separados entre sí por una pequeña zona divisoria denominada zona de conexión. Tanto el plano AND como el plano OR disponen, a su vez, de dos zonas externas denominadas buffers o separadores de entrada y de salida. Las señales de entrada del PAL llegan a los buffers de entrada del plano AND y producen las señales invertidas Ambos tipos de señales penetran verticalmente en el plano AND y generan los términos productos. Estos últimos discurren horizontalmente por ambos planos, atravesando previamente la zona de conexión, y producen finalmente las salidas del PAL mediante la realización de sumas lógicas entre los términos producto anteriores.

Además de las zonas mencionadas, existen otras dos regiones especiales. Una de ellas está situada a la izquierda del plano AND y la otra en la parte superior del plano OR. Estas regiones están constituidas por transistores de "pull-up", que actúan como resistencia de carga, a través de los cuales se alimentan las líneas de los términos producto y las líneas de salida del PAL respectivamente. En la Figura B.2 se muestra un esquema global de su estructura.

La realización física de un PAL se lleva a cabo mediante la conexión de cada una de las celdas que pertenecen a las regiones anteriores (buffers de entrada, plano AND, transistores de pull-up del plano AND, conexión AND-OR, plano OR, transistores de pull-up del plano OR y buffers de salida).

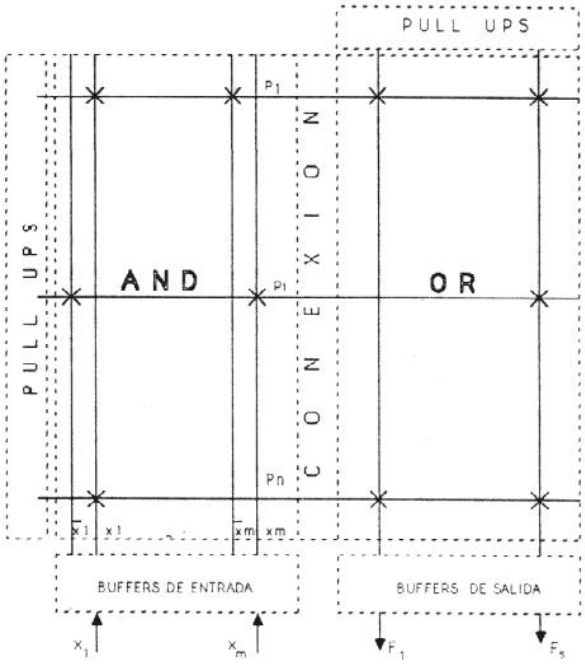


Figura B.2. Estructura de una PAL. (Fuente [32])

La estructura en bloques de las PALs se muestra en la Figura B.3, y un ejemplo específico se encuentra en la Figura B.4.

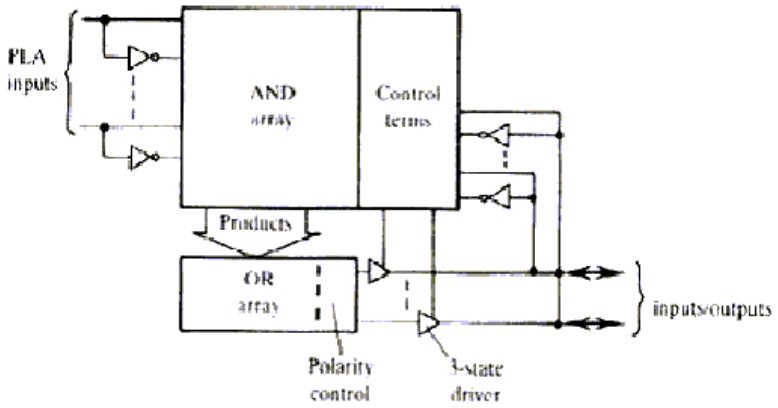


Figura B.3. Estructura en bloques de las PAL (Fuente [33])

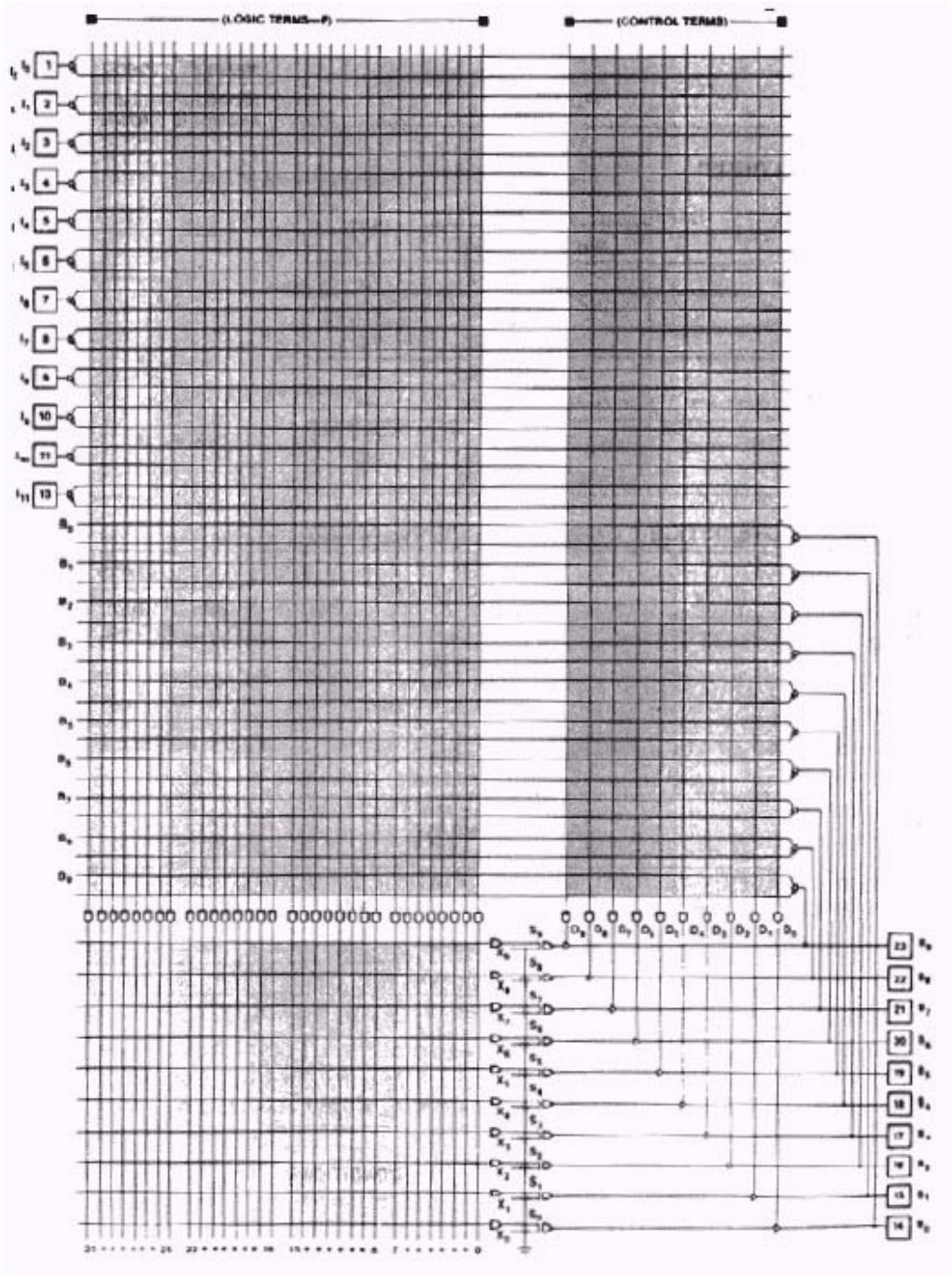


Figura B.4. Ejemplo de AL PAL. (Fuente [34])

B.3 Arquitecturas de los Dispositivos Lógicos Programables

Existen en la actualidad infinidad de arquitecturas diferentes de PLDs y su número se incrementa día a día. Dado que casi imposible hacer una referencia completa de todos los tipos de PLDs en el mercado, sólo se presentarán algunos de los más comunes.

Ya que generalmente los PLDs disponen de muchas entradas y resultaría muy complicado mostrarlas en un dibujo, se utiliza una representación simplificada, según la cual, para las puertas AND sólo se dibuja una línea de entrada llamada línea producto. Esta línea se cruza con dos líneas por cada entrada (entrada directa y entrada invertida), pudiendo existir un fusible en cada intersección. Aunque sólo se dibuja una línea de entrada por cada puerta AND, en realidad esta puerta tiene tantas entradas como intersecciones de la línea producto. Si en una intersección hay una X, significa que el fusible está intacto; sí no hay una X, el fusible esta fundido y no existe la conexión. En ocasiones, las puertas OR también se dibujan con una sola entrada.

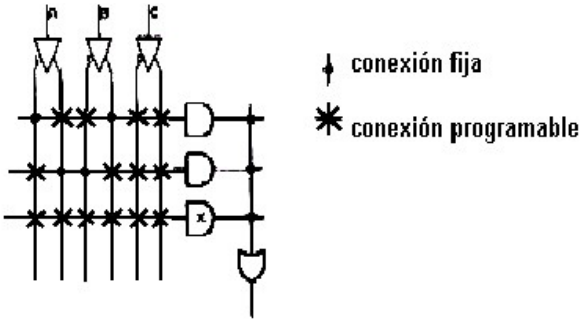


Figura B.5. Diagrama simplificado de un PLD.

En el diagrama simplificado de la Figura B.5 aparece una matriz de puertas AND de seis entradas, cuyas salidas están conectadas a una puerta OR. La intersección de las líneas producto con las líneas de entrada forman una matriz de puertas AND programable de 6 x 3 fusibles. El circuito está programado para

realizar la función OR –exclusiva entre las entradas A y B-. La puerta AND inferior está marcada con una X. Significa que todos sus fusibles están intactos y que su salida es 0. Cuando se funden todos los fusibles de una línea producto, la salida de la puerta AND asociada es 1.

A partir de esta configuración básica los PLDs se pueden distinguir los siguientes tipos de PLDs.

- **PAL (Programmable Array Logic).** También llamados PLAs, son un tipo de PLDs en las que se pueden programar las uniones en la matriz de puertas AND, siendo fijas las uniones en la matriz de puertas OR, como se observa en la Figura B.6.

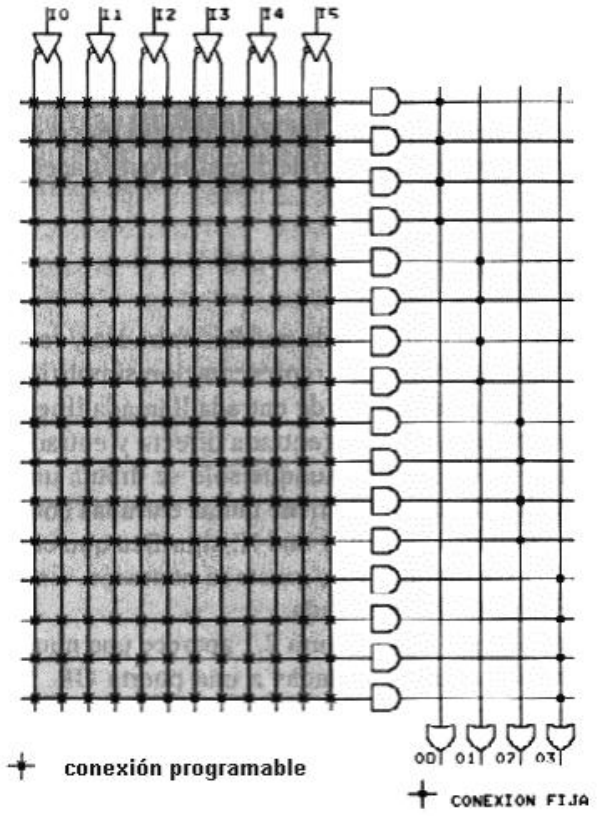


Figura B.6. Arquitectura PLA. (Fuente [32])

- **FPLA (Field Programmable Logic Array).** Es un PLD en el que se pueden programar las uniones en ambas matrices, como se ve en la Figura B.7. Son los dispositivos más flexibles, pero resultan penalizados en tamaño y en velocidad debido a los transistores adicionales en la matriz de puertas OR. Se utilizan fundamentalmente para construir máquinas de estados. Para otras aplicaciones, las PAL resultan más efectivas. Las PAL y las FPLA son sistemas combinacionales incompletos porque teniendo n entradas, disponen de menos de 2^n términos producto.

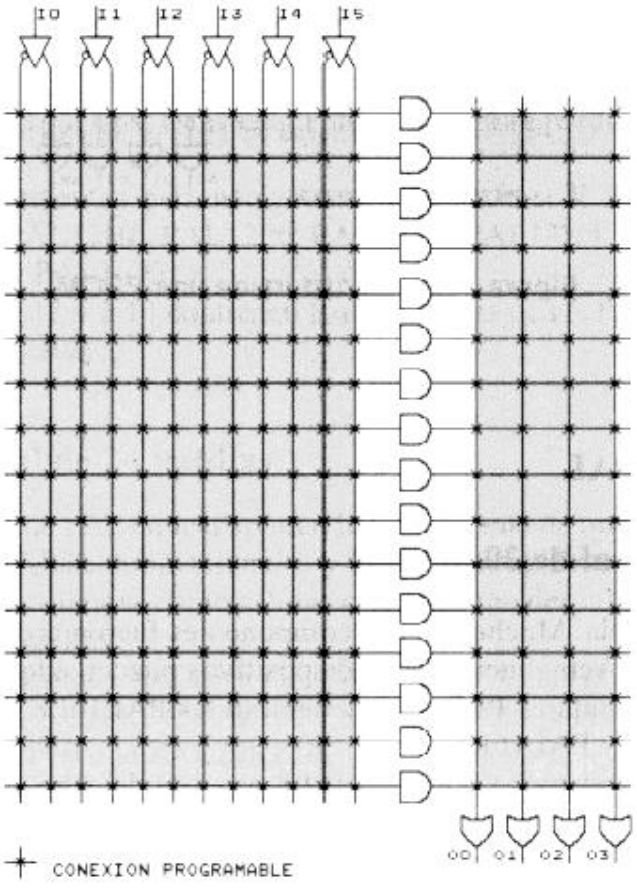


Figura B.7. Arquitectura FPLA. (Fuente [32])

- **PROM (Programmable Read Only Memory).** Es un PLD en el que las uniones en la matriz de puertas AND es fija, siendo programables las uniones en la matriz de puertas OR, como se muestra en la Figura B.8. Una PROM es un sistema combinacional completo que permite realizar cualquier función lógica con las n variables de entrada, ya que dispone de 2^n términos productos. Están muy bien adaptadas para aplicaciones tales como: tablas, generadores de caracteres, convertidores de códigos, etc. Generalmente las PROM tienen menos entradas que las PAL y FPLA.

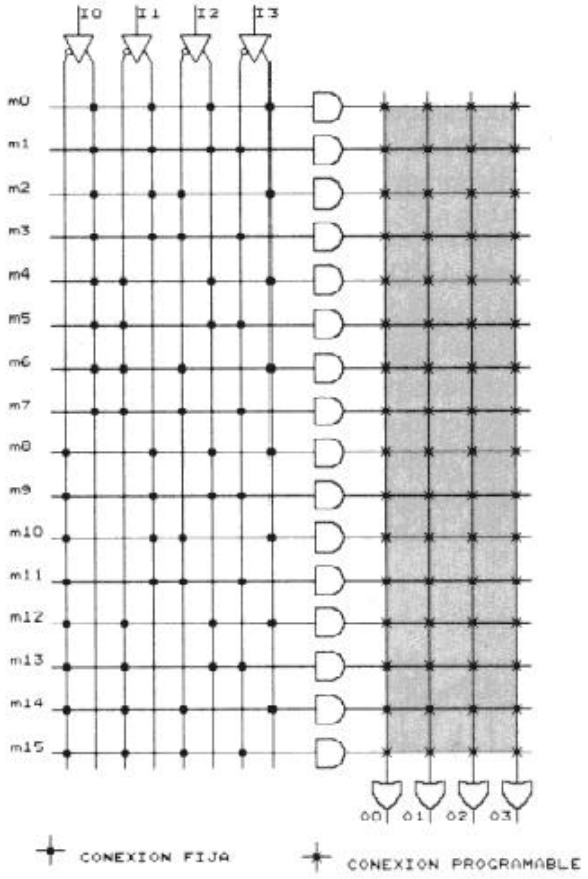


Figura B.8. Arquitectura PROM. (Fuente [32])

Se pueden encontrar PROM con capacidades potencia de 2, que van desde las 32 hasta las 8192 palabras de 4, 8 o 16 bit de ancho.

B.4 Dispositivos PAL combinacionales y sincrónicos

Los PALs pueden ser combinacionales o sincrónicos. La estructura de los dispositivos PALs combinacionales se muestra en la Figura B.9.

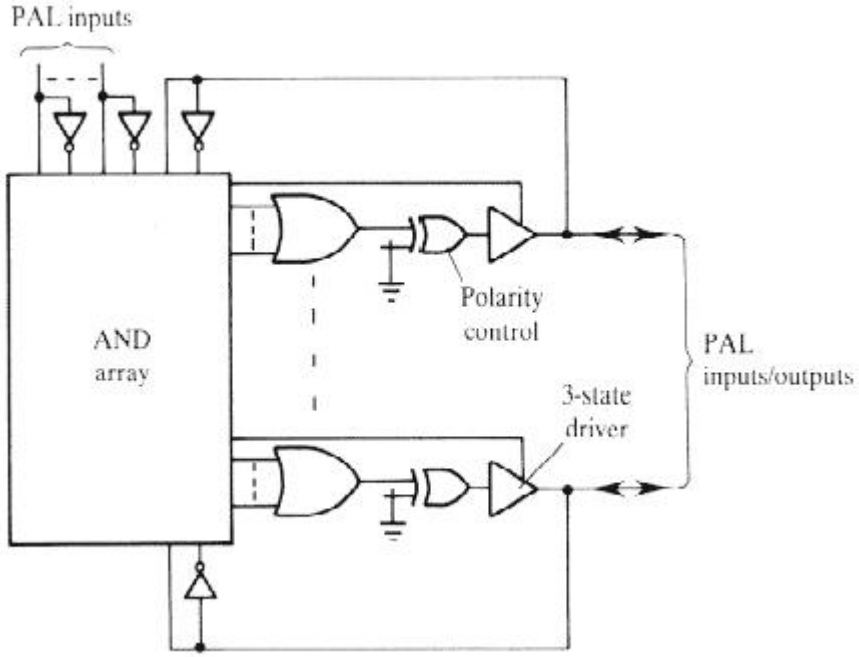


Figura B.9. PAL combinacional (Fuente [33])

Los SPAL (PAL sincrónica) tienen registros de salida alimentados desde el array lógico como el dispositivo mostrado en la Figura B.10. Todos estos dispositivos tienen un reloj externo común, siendo estos apropiados para diseños síncronos.

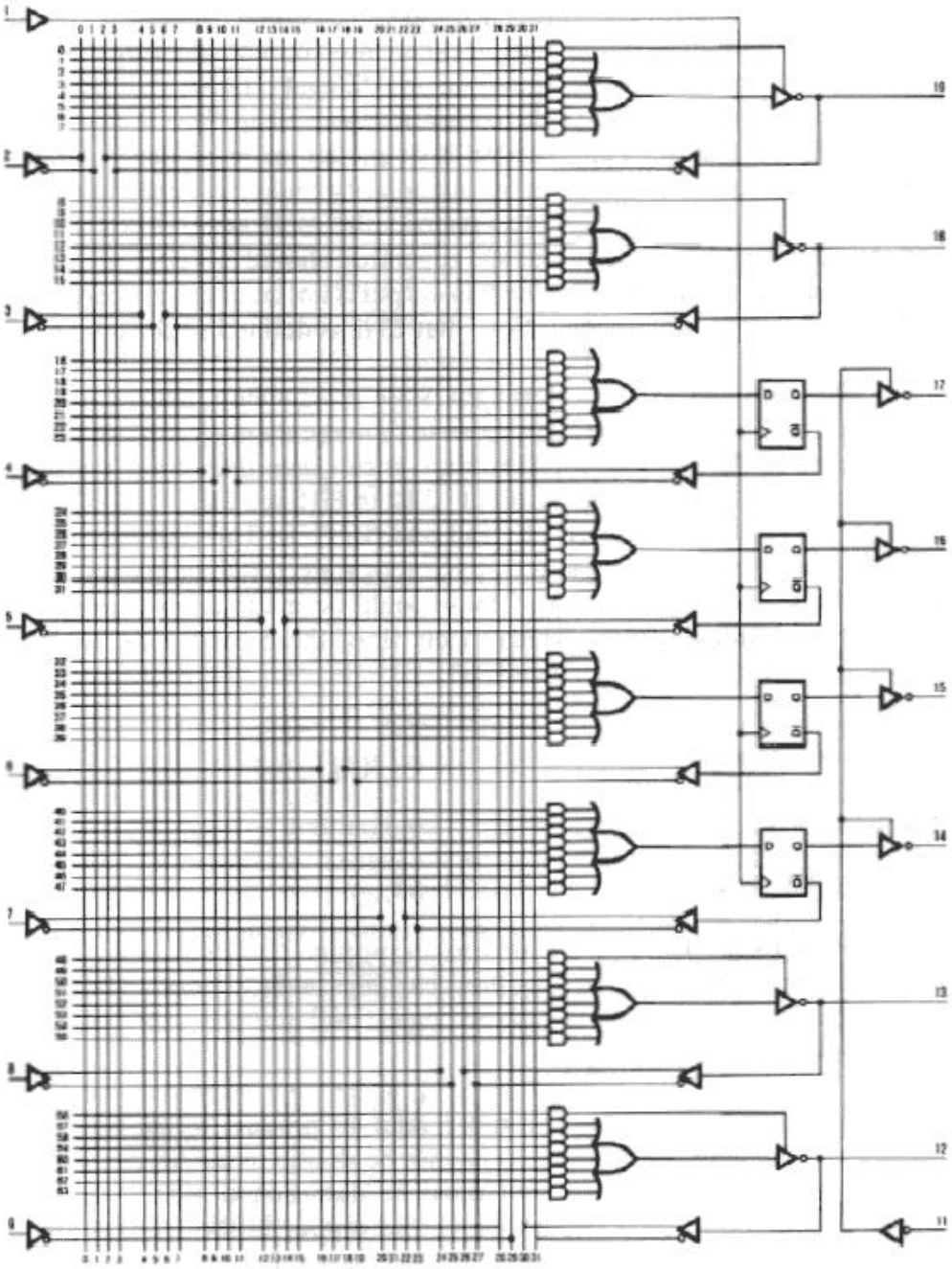


Figura B.10. PAL sincrónico. (Fuente [34])

B.5 VPAL (PAL versátil)

Estas estructuras representan el cambio de los PALs hacia los PLD de más alto nivel. En ellas se incluye una estructura de celda de salida más compleja llamada macrocelda. Un ejemplo de VPAL es la 22V10, cuya arquitectura se muestra en la Figura B.11. Incluye una macrocelda programable y una distribución de términos productos variable.

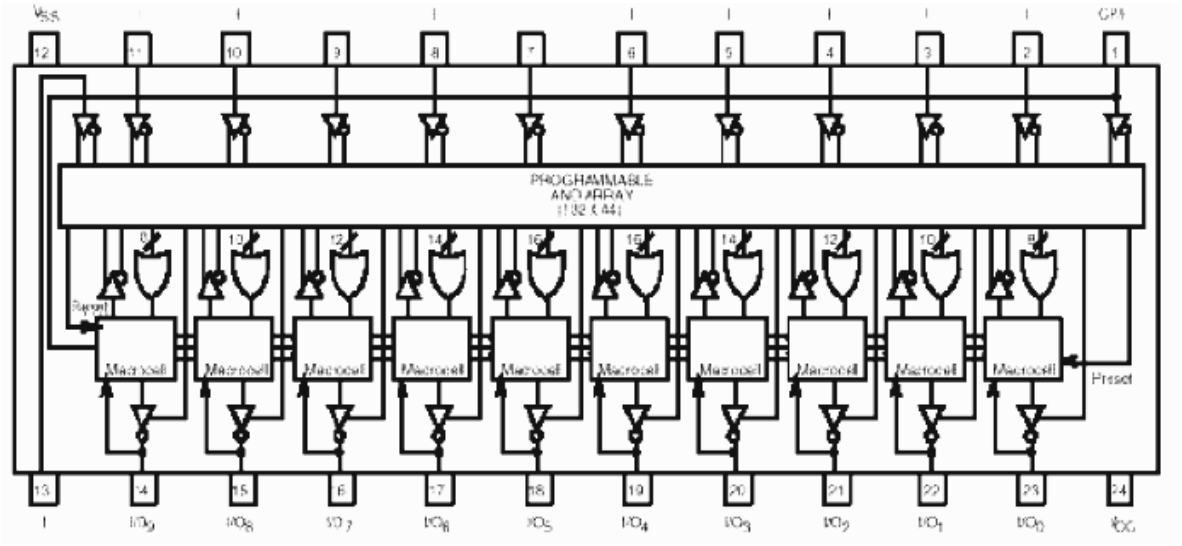


Figura B.11. Ejemplo de VPAL (Fuente [34])

Cada macrocelda se puede configurar de forma independiente. Los bits de configuración permiten poner a la salida el resultado del bloque de matrices en diferentes formas: directamente o a través de un registro. Existe un multiplexor que, en función de las líneas de control, selecciona un tipo de salida. Esto se observa en la Figura B.12. También se puede elegir el estado activo en la salida de la macrocelda (activo alto o activo bajo). De esta forma se puede reducir el número de términos necesarios para describir una función y se simplifica la implementación.

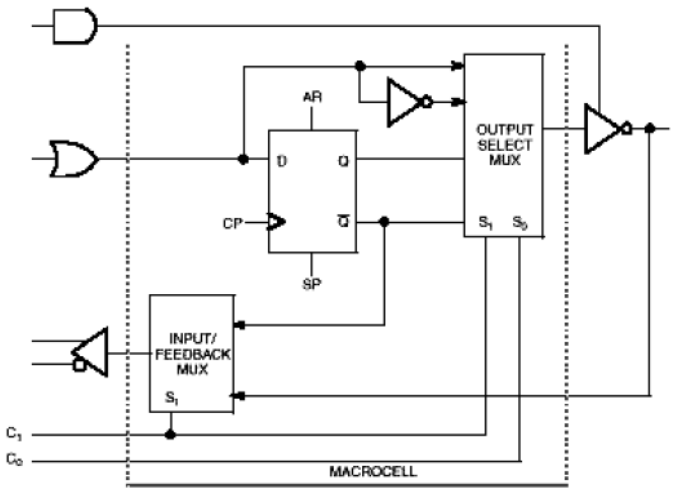


Figura B.12. Macrocelda de una VPAL (Fuente [34])

B.6 GAL (Generic Array Logic)

Se desarrollaron por la empresa Lattice Semiconductor. Son estructuras que presentan mayor flexibilidad de configuración de entrada – salida que las PAL. La celda de salida de una GAL es la de la Figura B.13.

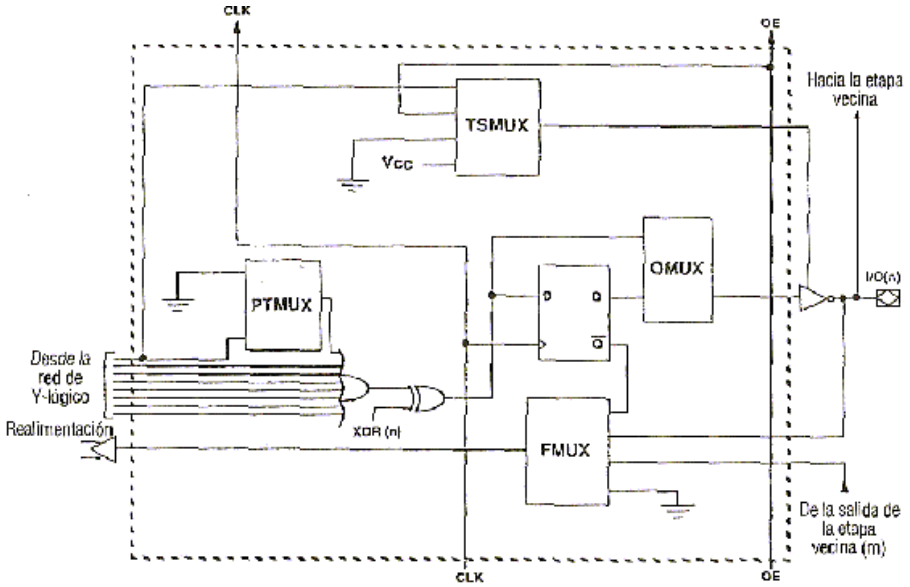


Figura B.13. Celda de salida de un GAL (Fuente [33])

Si bien las aplicaciones son similares a las de las PAL, estos dispositivos pueden soportar aplicaciones más complejas.

B.7 EPLD (Erasable Programmable Logic Device)

Estos dispositivos introducidos por Altera tienen una estructura interna, tipos de macrocelda y redes de interconexión muy diferentes a los descriptos anteriormente.

Pueden alojar en un solo encapsulado a veinticinco PALs clásicas o más de un centenar de funciones TTL comunes.. Además son programables por el usuario mediante un material y software sencillo que se pueden instalar en cualquier PC. Los objetivos de estos dispositivos son permitir una densidad de integración muy superior a los PALs y funcionar a una velocidad comparable a los PAL más rápidos.

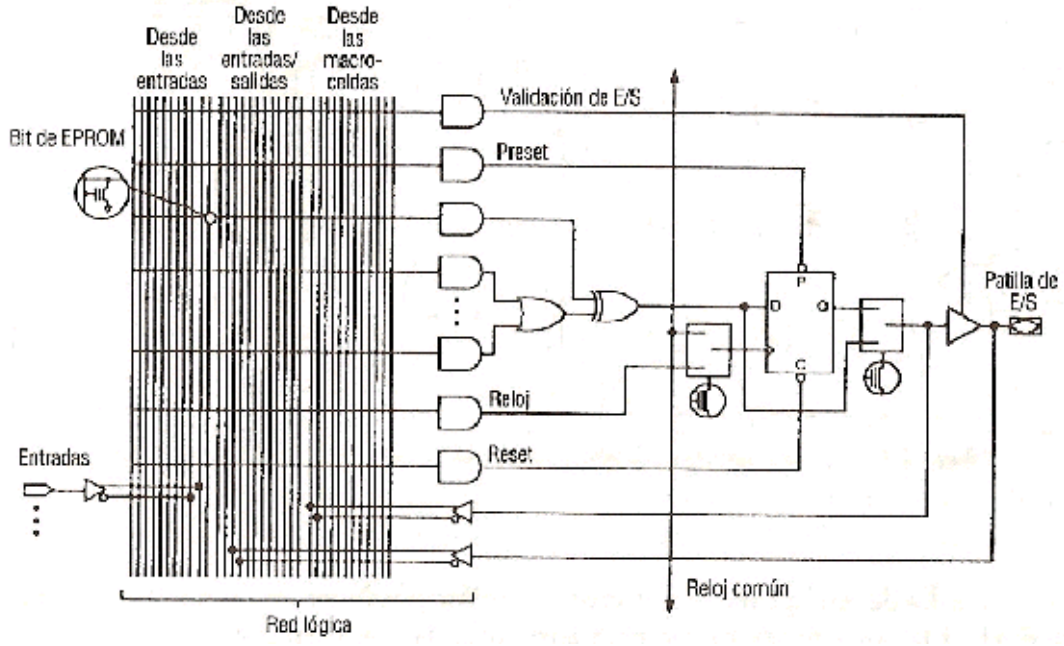


Figura B.14. Macrocelda EPLD. (Fuente [35])

Las macroceldas de los EPLDs presentan diferentes subconjuntos que pueden utilizarse conjuntamente o separadamente unos de otros de una macrocelda en particular. Esta característica se ve en la Figura B.14. Las posibilidades de conexión pueden ser de las señales de entrada al CI reservadas para este fin, de señales de entradas – salidas definidas por el usuario o de otras macroceldas. De esta manera se logra una gran flexibilidad en las conexiones.

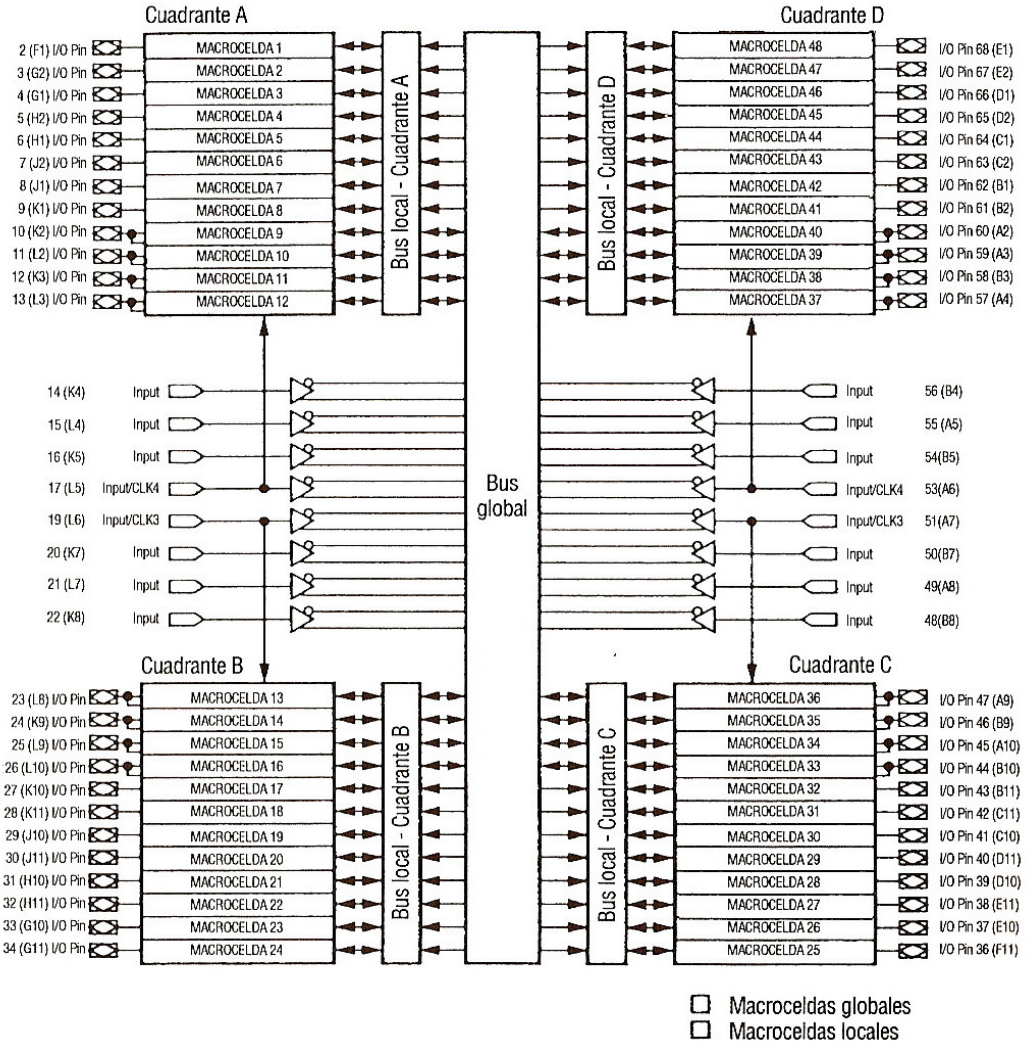


Figura B.15. Distribución de las macroceldas. (Fuente [35])

Por otra parte algunas macroceldas son locales y otras globales. En la Figura B.15 se observa que existe un bus local que pertenece a un grupo de macroceldas y un bus global para todas las macroceldas.

Esta distinción se debe a las posibilidades de conexión. Las locales tienen acceso a entradas dedicadas, a los retornos globales de los grupos de macroceldas y a los retornos de las macroceldas del grupo en que se encuentran. Por el contrario no pueden reinyectar sus señales de salida nada más que al bus local. Las macroceldas globales tienen acceso a los mismos recursos que las locales y además pueden reinyectar sus salidas sobre el bus local. La Figura B.16 presenta esta característica.

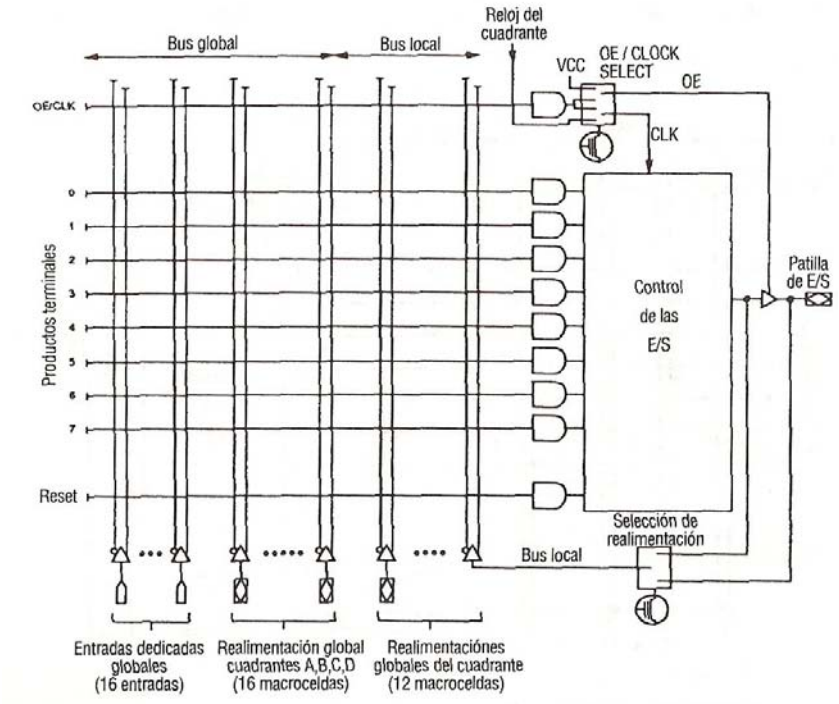


Figura B.16. Posibilidades de conexión de las macroceldas locales. (Fuente[35])

Las tablas B.1.1 y B.2.2 presentan la clasificación y características de los PLDs de las empresas Altera y Xilinix respectivamente.

	FAMILIA		CARACTERÍSTICAS	
			ARQUITECTURA	***MACROCELDA***
BPLD	Classic	EP220-EP224	• Una matriz de interconexión.	• Un biestable • Una realimentación
		EP610-EP910	• Una matriz de interconexión.	• Un biestable • Una realimentación
APLD	Classic	EP312-EP324	• Una matriz de interconexión. • Asignación variable de recursos: - Distribución simple de sumas de productos.	• Un biestable • Doble realimentación
		EP1810	• Segmentado.	• Un biestable • Una realimentación
	MAX	5000	• Segmentado. • Asignación variable de recursos: - Puertas NO-Y de expansión.	• Un biestable • Doble realimentación
		7000	• Segmentado. • Asignación variable de recursos: - Puertas NO-Y de expansión. - Distribución de sumas en cascada.	• Un biestable • Doble realimentación
		3000	• Segmentado. • Asignación variable de recursos: - Puertas NO-Y de expansión. - Distribución de sumas en cascada.	• Un biestable • Doble realimentación
CPLD	MAX9000		• Segmentado. • Recursos de interconexión distribuidos. • Asignación variable de recursos: - Puertas NO-Y de expansión. - Distribución de sumas en cascada.	• Un biestable • Doble realimentación
	FLASHlogic		• Segmentado. • Bloques lógicos mutifuncionales configurables como: - Memoria SRAM. - PLD avanzado. • Recursos lógicos operativos adicionales: - Comparador de 12 bits. • Asignación variable de recursos: - Distribución simple de sumas.	• Un biestable • Doble realimentación

Tabla B.1. Clasificación y características de PLDs de Altera (Fuente [23])

	FAMILIA	CARACTERÍSTICAS	
		ARQUITECTURA	*MACROCELDA*
APLD	9500	• Segmentado (Con dos matrices de interconexión). • Asignación variable de recursos: - Distribución de sumas de productos lógicos en cascada.	• Un biestable • Doble realimentación
	CoolRunner XPLA	• Segmentado (Con dos matrices de interconexión). • Asignación variable de recursos: - Compartición de algunos productos lógicos.	• Un biestable • Doble realimentación
	CoolRunner XPLA 3	• Segmentado (Con dos matrices de interconexión). • Asignación variable de recursos: - Puertas NO-Y de expansión. - Compartición de todos los productos lógicos.	• Un biestable • Doble realimentación
CPLD	CoolRunner XPLA 2	• Segmentado (Con tres matrices de interconexión). • Bloques lógicos complejos (Circuitos PLD avanzados segmentados). • Asignación variable de recursos: - Compartición de algunos productos lógicos.	• Un biestable • Doble realimentación

Tabla B.1. Clasificación y características de PLDs de Xilinix (Fuente[23])

B.8 ASPLDs. (Application Specific Programmable Logic Devices)

Los ASPLDs (Application Specific Programmable Logic Devices) son PLDs diseñados para realizar funciones especificas como, decodificadores de alta velocidad, secuenciadores, interfaces para buses particulares, periféricos programables para microprocesadores, etc.

Partes del ASPLD son programables permitiendo la adaptación del circuito a una aplicación determinada, pero manteniendo su función básica; así, por ejemplo, un decodificador lo personaliza el usuario, pero sigue siendo un decodificador. Estos circuitos están muy optimizados para la función para la que han sido diseñados.

Los decodificadores sólo tienen un término producto, carecen de puertas OR y resultan por consiguiente muy rápidos; por otro lado, los circuitos de interface para buses normalmente tienen un Fan-Out elevado.

Estos CI implementan una función específica en un amplio mercado, combinando una colección de funciones.

Un ejemplo es el CI CC770 de la Figura B.16 que se trata de un controlador de comunicaciones seriales que realiza un a comunicación serial de acuerdo al protocolo CAN, cumpliendo funciones como transmisión y recepción de mensajes, filtrado de mensajes, detección de interrupciones con una mínima interacción con el microcontrolador maestro.

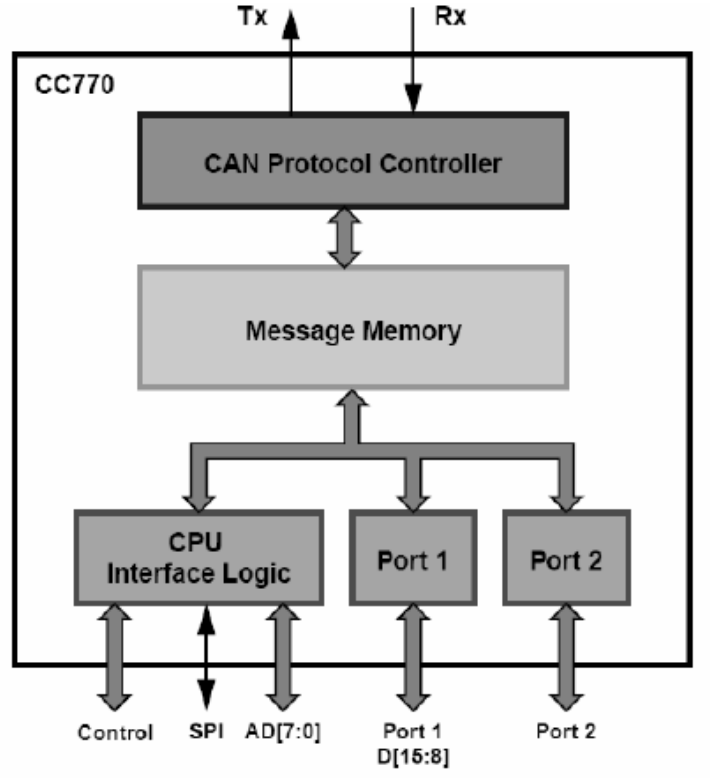


Figura B.16. Diagrama en bloques del CI CC770

Otro ejemplo de estos circuitos integrados puede ser el de la Figura b.17. Se trata de el CI PD6708 que es un controlador de periférico LSI para microcontroladores que utilizan el protocolo IEBus. El PD6708 provee un buffer de transmisión – recepción permitiendo que el microcontrolador se ocupe principalmente de procesar una aplicación, liberándolo de las tareas de transmisión y recepción.

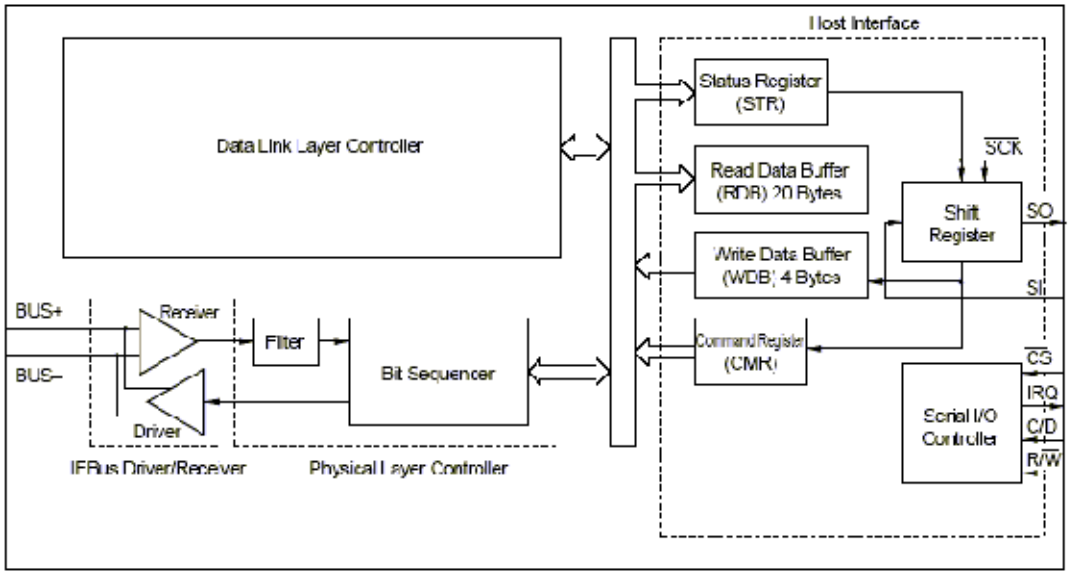


Figura B.17. Diagrama en bloques del CI PD6708

B.9 FPGAs. (Field Programmable Gate Arrays)

Las FPGAs (Field Programmable Gate Arrays) contienen bloques lógicos relativamente independientes entre sí, con una complejidad similar a un PLD de tamaño medio. Estos bloques lógicos pueden interconectarse, mediante conexiones programables, para formar circuitos mayores. Existen FPGAs que utilizan pocos bloques grandes (Pluslogic, Altera y AMD) y otras que utilizan muchos bloques pequeños (Xilinx, AT&T, Plessey, Actel).

A diferencia de los PLDs , no utilizan arquitectura de matriz de puertas AND seguida de la matriz de puertas OR y necesitan un proceso adicional de ruteado del que se encarga un software especializado.

La primera FPGA la introdujo Xilinx en el año 1985. La programación de las FPGAs de Xilinx basadas en RAM estática es diferente a la programación de los PLDs. Cada vez que se aplica la tensión de alimentación, se reprograma con la información que lee desde una PROM de configuración externa a la FPGA. Una FPGA basada en SRAM (RAM estática) admite un número ilimitado de reprogramaciones sin necesidad de borrados previos.

En general la complejidad de una FPGA es muy superior a la de un PLD. Los PLD tienen entre 100 y 2000 puertas, las FPGAs tienen desde 1200 a 100.000 puertas y la tendencia es hacia un rápido incremento en la densidad de puertas. El número de flip-flops de las FPGA generalmente supera al de los PLD. Sin embargo, la capacidad de la FPGA para realizar lógica con las entradas suele ser inferior a la de los PLD. Por ello, se puede decir que los diseños que precisan lógica realizada con muchas patillas de entrada y con pocos flip-flops, pueden realizarse fácilmente en unos pocos PLDs, mientras que en los diseños en los que intervienen muchos registros y no se necesita generar combinaciones con un elevado número de entradas, las FPGAs pueden ser la solución óptima.

Los conjuntos configurables de puertas FPGA están formados por tres elementos básicos, como se puede ver en la Figura B.18.

- Los bloques lógicos internos

Son recursos lógicos que permiten realizar diferentes funciones lógicas. Su complejidad puede variar desde un simple par de transistores hasta un conjunto de memorias de acceso aleatorio llamadas tablas de consulta (en inglés "look-up tables"), combinadas o no con elementos de memorización, de inversión de

variables, etc. En el primer caso, se obtiene una FPGA de granularidad fina que, tal como se indica anteriormente, posee gran cantidad de bloques lógicos muy sencillos con capacidad para realizar un número muy limitado de funciones lógicas cada uno. En el segundo caso, la FPGA es de granularidad gruesa y posee un menor número de bloques lógicos pero cada uno de ellos tiene capacidad para realizar funciones de cierta complejidad.

- Los bloques lógicos de entrada y salida

Son recursos lógicos que establecen el enlace entre los bloques lógicos internos y los terminales de entrada/salida. Al igual que en el caso anterior, estos bloques pueden tener diferentes grados de complejidad.

- Los recursos de interconexión.

Son un conjunto de líneas e interruptores programables que permiten transmitir las señales entre los bloques lógicos internos y entre ellos y los bloques de entrada/salida. Reciben en inglés la denominación de "routing channels".

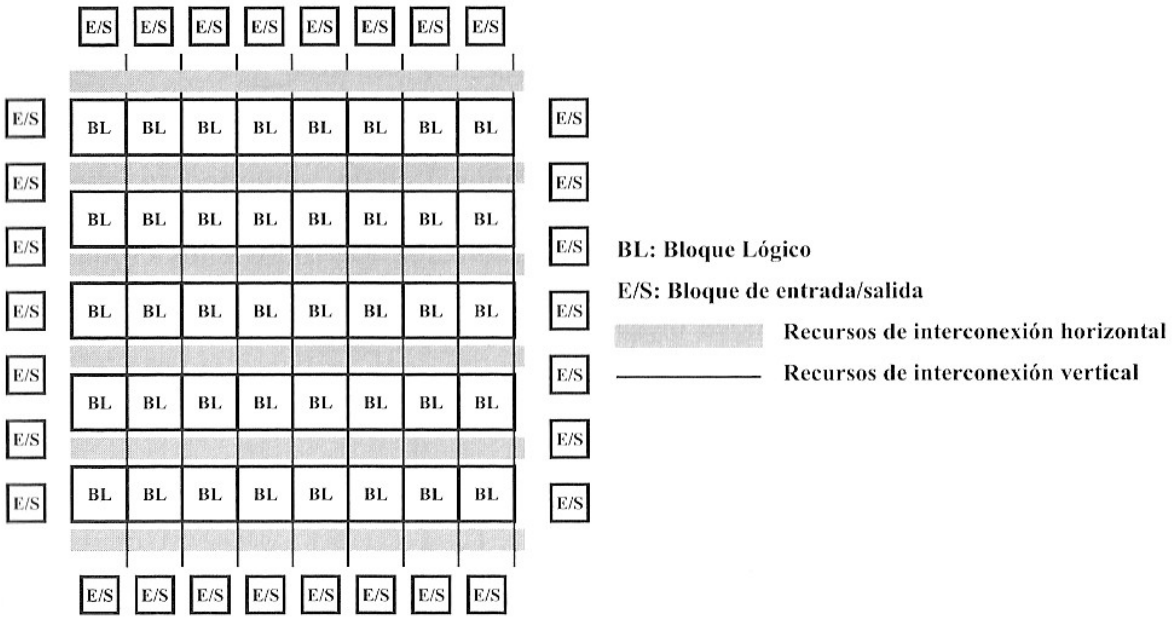


Figura B.18. Bloques de una FPGA.

Tal como se indica anteriormente, tanto la organización de los bloques lógicos internos como la forma de realizar los recursos de interconexión derivan de las arquitecturas desarrolladas en los circuitos integrados digitales semimedida, que presentan numerosas variantes. Por ello las FPGA son circuitos complejos en los que existe una gran cantidad de conceptos interrelacionados.

También se pueden distinguir dos tecnologías en las FPGAs. Una basada en los dispositivos introducidos por Xilinix que tienen bloques lógicos denominados LCA que es de tipo RAM y otra que se conoce como FPGA de antifusibles que es de tipo PROM de programación definitiva.

B.10 Características del diseño con PLDs

Los PLDs están situados en una zona intermedia entre los dispositivos a medida y la lógica de catálogo formada por los CI de función fija. Tienen casi todas las ventajas de los ASICs sin estar penalizados por un costo elevado para pequeñas series. Además el ciclo de diseño con PLDs es mucho más rápido que los de las matrices de puertas o las células normalizadas. En determinadas aplicaciones, un PLD puede sustituir desde unos pocos hasta unas decenas de CI de función fija, mientras que los grandes ASICs pueden sustituir a cientos e incluso miles de CI. En ocasiones, los PLD se utilizan para realizar prototipos que posteriormente se llevarán a un ASIC más complejo.

El trabajo con PLDs proporciona: facilidad de diseño, prestaciones, fiabilidad, economía y seguridad.

- **Facilidad de diseño**

Las herramientas de soporte al diseño con PLDs facilitan enormemente este proceso. Las hojas de codificación que se utilizaban en 1975 han dejado paso a los ensambladores y compiladores de lógica programable (PALASM, AMAZE, ABEL, CUPL, OrCAD/PLD, etc). Estas nuevas herramientas permiten expresar la

lógica de los circuitos utilizando formas variadas de entrada tales como; ecuaciones, tablas de verdad, procedimientos para máquinas de estados, esquemas, etc. La simulación digital posibilita la depuración de los diseños antes de la programación de los dispositivos.

- **Prestaciones.**

Los PLDs TTL que hay en el mercado tienen tiempos de conmutación tan rápidos como los circuitos integrados de función fija más veloces. Los PLDs ECL son todavía más rápidos. Sin embargo, el incremento de velocidad obtenido con los dispositivos CMOS, que ya han igualado o superado en prestaciones a los dispositivos TTL, está provocando el abandono de la tecnología bipolar por parte de los fabricantes. En cuanto al consumo de potencia, los PLDs generalmente consumen menos que el conjunto de chips a los que reemplazan.

- **Fiabilidad.**

Cuanto más complejo es un circuito, más probabilidades hay de que alguna de sus partes falle. Puesto que los PLDs reducen el número de chips en los sistemas, la probabilidad de un fallo disminuye. Los circuitos impresos con menor densidad de CI son más fáciles de construir y más fiables. Las fuentes de ruido también se reducen.

- **Economía.**

En este apartado, hay aspectos que resultan difíciles de cuantificar. Por ejemplo, los costos de pérdida de mercado por un introducción tardía de un producto. Otros son más claros, por ejemplo, la reducción del área de las placas de circuito impreso obtenida gracias a que cada PLD sustituye a varios circuitos integrados de función fija. Muchas veces se consigue reducir el número de placas de circuito impreso economizándose en conectores.

De la misma manera que para altos volúmenes de producción las memorias ROM resultan de menor costo que las EPROM, las HAL (Hard Array Logic) o PLDs programados por el fabricante proporcionan ahorros adicionales en grandes cantidades.

- **Seguridad.**

Los PLDs tienen fusibles de seguridad que impiden la lectura de los dispositivos programados, protegiendo los diseños frente a copias.

Además de los puntos mencionados, podemos añadir que los PLDs facilitan el ruteado de las placas de circulo impreso debido a la libertad de asignación de patillas que proporcionan. Permiten realizar modificaciones posteriores del diseño y en ocasiones hacen posible la reutilización de circuitos impresos con algunos fallos, mediante una reasignación de los PLDs.

B.11 Consumo de corriente en los PLDs.

En la fabricación de PLDs se utiliza tecnología bipolar TTL o ECL y tecnología CMOS. Los dispositivos bipolares son más rápidos y consumen más que los dispositivos CMOS. Actualmente los PLDs bipolares presentan retardos de propagación inferiores a 7 nsg y los consumos típicos rondan los 100-200 mA para un chip con 20-24 patillas.

Mientras los PLDs bipolares sólo pueden programarse una vez, la mayoría de los PLDs CMOS son reprogramables y permiten una fácil verificación por parte del usuario. A los PLDs CMOS borrables por radiación ultravioleta se les denomina EPLD y a los borrables eléctricamente se les conoce por EEPLD. Los EEPLD con encapsulados de plástico son más baratos que los EPLD provistos de ventanas de cuarzo que obligan a utilizar encapsulados cerámicos.

También existen las PALCE16V8Q (Quarter Power Icc = 55 mA) y las PALCE16V8Z (Zero Power) con un bajísimo consumo estático de potencia.

Acostumbrados a trabajar con dispositivos CMOS con un consumo prácticamente nulo a frecuencia cero, resulta sorprendente una PAL CMOS con un consumo de 90 mA a la máxima frecuencia de operación (15 Mhz), pero que todavía tendrá un consumo apreciable a frecuencia cero. En la actualidad, solamente una pequeña fracción de los PLDs del mercado se anuncian como Zero Power.

Si consultamos las hojas de datos de una PALCE16V8H-20, encontramos claves que permiten extraer valiosa información del nombre del dispositivo. La información incluida en el nombre nos indica:

PAL Programmable Array Logic.

CE C-MOS Electrically Erasable.

16V8 16 Entradas a la matriz de puertas AND y ocho salidas.

H Half Power (Iec = 90 mA).

20 Tiempo de propagación = 20 nsg.

APÉNDICE C

Lenguaje de descripción de Hardware - VHDL

C.1 VHDL. Breve reseña histórica

A mediados de los años setenta se produce una fuerte evolución en los procesos de fabricación de los circuitos integrados, y junto a las tecnologías bipolares, surgen la MOS, principalmente la NMOS, promoviendo el desarrollo de circuitos digitales hasta la primera mitad de los años ochenta. En aquellas épocas, el esfuerzo de diseño se concentraba en los niveles eléctricos de establecer características e interconexiones entre los componentes básicos a nivel de transistor. El proceso de diseño era altamente manual y tan solo se empleaban herramientas como el PSPICE, para simular esquemas eléctricos con modelos previamente personalizados, a las distintas tecnologías.

A medida que pasaban los años, los procesos tecnológicos se hacían más y más complejos. Los problemas de integración iban en aumento y los diseños eran cada vez más difíciles de depurar y de dar mantenimiento. Inicialmente los circuitos MSI y LSI se diseñaron mediante la realización de prototipos basados en módulos más sencillos. Cada uno de estos módulos estaba formado por puertas ya probadas, este método poco a poco, iba quedándose obsoleto. En ese momento (finales de los años setenta) se constata el enorme desfase que existe entre tecnología y diseño.

La considerable complejidad de los chips que se pueden fabricar, implica unos riesgos y costos de diseño desmesurados e imposibles de asumir por las empresas. Es entonces, cuando diversos grupos de investigadores empiezan a crear y desarrollar los llamados "lenguajes de descripción de hardware" con sus características diferentes. Empresas tales como IBM con su IDL, el TI - HDL de Texas Instruments, ZEUS de general Electric, etc., así como los primeros

prototipos empleados en las universidades, empezaron a desarrollarse buscando una solución a los problemas que presentaba el diseño de los complejos sistemas. Sin embargo, estos lenguajes nunca alcanzaron el nivel de difusión y consolidación necesarios por motivos distintos. Unos, los industriales, por ser propiedad de la empresa permanecieron encerrados en ellas y no estuvieron disponibles par su estandarización y mayor difusión, los otros, los universitarios, perecieron por no disponer de soporte ni mantenimiento adecuado.

Alrededor de 1981 el Departamento de Defensa de los Estados Unidos desarrolla un proyecto llamado VHSIC (*Very High Speed Integrated Circuit*), que posteriormente pasó a denominarse simplemente VHDL. Su objetivo era rentabilizar las inversiones en hardware haciendo más sencillo su mantenimiento. Se pretendía con ello resolver el problema de modificar el hardware diseñado en un proyecto para utilizarlo en otro, lo que no era posible hasta entonces porque no existía una herramienta adecuada que armonizase y normalizase dicha tarea.

En 1983, IBM, Intermetrics y Texas Instruments empezaron a trabajar en el desarrollo de un lenguaje de diseño que permitiera la estandarización, facilitando con ello, el mantenimiento de los diseños y la depuración de los algoritmos, para ello el IEEE propuso su estándar en 1984.

Tras varias versiones llevadas a cabo con la colaboración de la industria y de las universidades, que constituyeron a posteriori etapas intermedias en el desarrollo del lenguaje. El IEEE adoptó en diciembre de 1987 a VHDL como su lenguaje HDL, dando un impulso a éste frente a sus competidores. El estándar IEEE std 1076-1987 que constituyó el punto firme de partida de lo que después de cinco años sería ratificado como VHDL.

Esta doble influencia, tanto de la empresa como de la universidad, hizo que el estándar asumido fuera un compromiso intermedio entre los lenguajes que ya

habían desarrollado previamente los fabricantes, de manera que éste quedó como ensamblado y por consiguiente un tanto limitado en su facilidad de utilización haciendo dificultosa su total comprensión. Este hecho se ha visto incluso ahondado en su revisión de 1993.

Pero esta deficiencia se ve altamente recompensada por la disponibilidad pública, y la seguridad que le otorga el verse revisada y sometida a mantenimiento por el IEEE.

La independencia en la metodología de diseño, en el proceso de fabricación, su capacidad descriptiva en múltiples dominios y niveles de abstracción, su versatilidad para la fabricación de sistemas complejos, su posibilidad de reutilización y en definitiva la independencia de que goza con respecto de los fabricantes, han hecho que VHDL se convierta con el paso del tiempo en el lenguaje de descripción de hardware por excelencia.

C.2 Conociendo el lenguaje VHDL

El lenguaje VHDL está creado específicamente para el diseño de hardware, es decir, permite implementar con él una multitud de circuitos lógicos, tanto combinacionales como secuenciales. Además se pueden simular. Con VHDL sintetizable se llega a grabar un dispositivo lógico programable.

Un programa en VHDL consta de dos partes. La primera, la entidad, sirve para relacionar el diseño con el mundo exterior, es decir, analizar lo que se trata de crear como una "caja negra", de la que sólo se conoce sus entradas, sus salidas y la disposición de las mismas. La segunda parte, la arquitectura, describe que es lo el programa va a hacer con las entradas para transformarlas en salidas.

C.2.1 Concurrencias y secuenciales

Es importante comprender desde un principio la diferencia entre concurrente y secuencial.

El concepto de concurrencia, se ve claramente graficado en los circuitos electrónicos donde los componentes se encuentran siempre activos, existiendo una asociación intrínseca, entre todos los eventos del circuito; ello hace posible el hecho de que si se da algún cambio en una parte del mismo, se produce una variación (en algunos casos casi instantánea) de otras señales.

Esta posibilidad de los circuitos reales obliga a que VHDL soporte estructuras específicas para el modelado y diseño de este tipo de especificaciones de tiempos y concurrencias en el cambio de las distintas señales digitales de los diseños.

Por el contrario, las asignaciones secuenciales, son más bien propios de los SDL (soft design lenguage) en los que la programación tiene un flujo natural secuencializado, siendo propio de este tipo se eventos las sentencias case, if, while, loop, etc más propias de estas sintaxis.

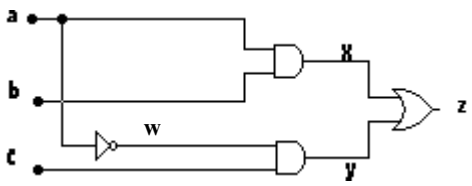
Las construcciones concurrentes del lenguaje son usadas dentro de estructuras concurrentes, por ejemplo una arquitectura tiene una naturaleza eminentemente concurrente (es decir que está activo todo el tiempo), mientras que el cuerpo de un process es en principio eminentemente secuencial. La asignación de eventos secuenciales dentro de una estructura concurrente se ejecutará de forma concurrente, es decir, al mismo tiempo que las demás sentencias.

VHDL soporta con este motivo, tres tipos de objetos: las variables, las constantes y las señales. Como las variables y las señales pueden variar su valor mientras ejecutamos un programa, serán éstas las encargadas de almacenar dichos datos, asimismo serán los portadores de la información. Únicamente las señales pueden tener la connotación de globalidad dentro de un programa, es decir, que pueden ser empleadas dentro de cualquier parte del programa a diferencia de las variables que solo tienen sentido en el interior de un proceso.

Los procesos (process), son estructuras concurrentes, constituidas por sentencias de ejecución secuencial. Por este motivo dentro de un proceso encontramos sentencias similares a las de los SDL (lenguajes de descripción de software) que nos llevan a emplear VHDL como si de otro lenguaje común se tratara. Dentro de un proceso podemos encontrar la declaración y la utilización de las variables como parámetros locales al proceso .

En ejecución secuencial, las variables evalúan su valor dentro del cuerpo del proceso de forma inmediata, sin consumir tiempo de ejecución, pero como están dentro de un process, que es una estructura concurrente, este valor no será asumido, sino hasta el final de la ejecución de dicho process. El siguiente ejemplo (Fuente:http://det.bp.ehu.es/vhdl) muestra la asignación concurrente para señales:

w<= **not** a;
x <= a **and** b;
y <= c **and** w;
z <= x **or** y;



Al producirse un cambio en la parte derecha de la estructura de asignación (<= NOT a;) de alguna de las sentencias, la expresión es evaluada de nuevo en su totalidad, y asignado el nuevo valor a la señal de la izquierda. Múltiples asignaciones en el cuerpo de una arquitectura se activan simultáneamente, por ejemplo, la estructura anterior se corresponde con el circuito de la derecha.

C. 3 Unidades Básicas de Diseño.

C. 3.1 Como se declara una Entidad

En la declaración de entidades, se definen las entradas y salidas del chip, diciendo cuántas son, de qué tamaño (de 0 a n bits), modo (entrada, salida, ...) y tipo (integer, bit,...). Las entidades pueden definir bien las entradas y salidas de un diseño más grande o las entradas y salidas de un chip directamente. La

declaración de entidades es análoga al símbolo esquemático de lo que se quiere implementar, el cual describe las conexiones de un componente al resto del proyecto, es decir, si hay una entrada o puerto de 8 bits, o dos salidas o puertos de 4 bits, etc. La declaración de entidades tiene la siguiente forma:

entity programa **is**	Cabecera del programa
port(	Se indica que a continuación viene los puertos (o grupos señales) de entrada y/o salida
-- puertos de entradas -- puertos de salidas -- puertos de I/O -- puertos de buffers	Aquí se declaran las entradas y/o salidas con la sintaxis que se verá a continuación. Las líneas empezadas por dos guiones son ignoradas por el compilador. El compilador no distingue las mayúsculas de las minúsculas
); **end** programa;	Se indica que se ha acabado la declaración de puertos de entrada y/o salida, y que se ha acabado la entidad

Cada señal en una declaración de entidad está referida a un puerto (o grupo de señales), el cual es análogo a un(os) pin(es) del símbolo esquemático. Un puerto es un objeto de información, el cual, puede ser usado en expresiones y al cual se le pueden asignar valores.

Seguido del nombre del puerto y separado de éste por dos puntos, viene el tipo de puerto que va a ser. El modo describe la dirección en la cual la información es transmitida a través del puerto. Éstos sólo pueden tener cuatro valores: in, out, buffer e inout. Si no se especifica nada, se asume que el puerto es del modo in.

VHDL sólo admite cuatro modos para los puertos, pero puede haber tantos tipos de señales como se necesiten, ya que las puede crear el usuario. VHDL incorpora varios tipos de forma estándar (por haber sido creado así), pudiendo usar otros mediante librerías normalizadas, y los creados por el usuario.

Ante la necesidad de ampliar la operatividad del tipo bit, la norma IEEE 1164, definió un nuevo tipo llamado **std_logic**, **std_ulogic**, y sus derivados tales como

std_logic_vector y **std_ulogic_vector**. Como su nombre pretende indicar, es el tipo de tipo lógico estándar.

Como ejemplo, a continuación se incluye la declaración de entidades de un multiplexor de 2x1 de cuatro bits, con entrada de habilitación o enable. El multiplexor necesita las entradas de información, la señal de selección, la de enable y las salidas de información.

```
entity multi is port (

enable: in bit;
selec:  in bit;
in1: in bit_vector(3 downto 0);
in2: in bit_vector(3 downto 0);
out1:out bit_vector(3 downto 0)

);
end multi;
```

Cabecera en la que **multi** es el nombre de la entidad

- **enable** es un bit de entrada (suficiente para habilitar o no)
- **selec** es otro bit de entrada, que selecciona la entrada **in1** o **in2**, ambas de 4 bits
- **out1** es de salida, que lógicamente, debe ser de la misma longitud que **in1** e **in2**

Notesé que el último puerto no lleva punto y coma al final de la línea. Si lo llevase estaría incorrecto

En el caso de querer dar un valor inicial a algunas de las señales se hace de la siguiente forma a: in bit:=´1´

Un aspecto interesante de VHDL es la parametrización. Mediante la cláusula generic se pueden crear entidades en las que los valores de algunos de sus parámetros se concretan en el momento de utilizarlas. Por ejemplo el tiempo de propagación de las señales desde la salida hasta la entrada.

```
entity multi is
Generic (t_delay:TIME:=5 ns);
port (
enable: in bit:=`0´;
selec:  in bit:=´0`;
in1: in bit_vector(3 downto 0);
in2: in bit_vector(3 downto 0);
out1:out bit_vector(3 downto 0)
```

VHDL no distingue las letras mayúsculas de las minúsculas, por lo que un puerto llamado por nosotros "EnTraDA" será equivalente a otro que se llame "ENTRADA" o "entrada".

El primer carácter de un puerto sólo puede ser una letra, nunca un número. Así mismo, no pueden contener caracteres especiales como $, %, ^, @, ... y dos caracteres de subrayado seguidos.

C.3.2 Como se declara una arquitectura

La arquitectura es la que expresa que deben hacer los grupos de señales de entrada que se han declarados previamente en la entidad para obtener las salidas, también declarados en la entidad. En la declaración de la arquitectura reside todo el funcionamiento de un programa, ya que es ahí donde se indica que hacer con cada entrada, para obtener la salida. Si la entidad es vista como una "caja negra", para la cual lo único importante son las entradas y las salidas, entonces, la arquitectura es el conjunto de detalles interiores de la caja negra.

La declaración de arquitecturas debe constar de las siguientes partes como mínimo, aunque suelen ser más:

architecture archpro **of** programa **is**	Cabecera de la arquitectura. En ésta, **archpro** es un nombre cualquiera (suele empezar por "arch", aunque no es necesario) y **programa** es el nombre de una entidad existente en el mismo archivo
-- declaración de señales y otros accesorios	Declaraciones de apoyo
begin	Se da comienzo al programa
-- núcleo del programa	Conjunto de sentencias, bucles, procesos, funciones,... que dan operatividad al programa.
end archpro;	Fin del programa

Se apreciar que es una estructura muy sencilla, y que guarda alguna relación con Turbo Pascal. Las sentencias entre begin y end son las que realmente "hacen algo". A continuación, está el código fuente de un programa en VHDL de un multiplexor (esta es una de las múltiples formas de implementar un multiplexor en VHDL), el cual debe ir unido a la entidad expuesta en el apartado de la declaración de entidades, ya que una parte sin la otra carecen de sentido.

architecture archimulti **of** multi **is**	Cabecera de la arquitectura. En esta ocasión el nombre de la arquitectura es **archimulti**, y el de la entidad es **multi**, la cual está definida anteriormente.
-- señales	En este programa no se necesitan señales
Begín	Se da comienzo al programa

process(enable,in1,in2)
begin
if enable='0' **then** out1<="1111";
elsif enable='1' **then**
if(selec = '0') **then** out1<=in1;
elsif(selec = '1') **then**
out1<=in2;
end if;
end if;
end process;
	Sentencias que hacen que la entidad definida como multiplexor hagan la función propia de su nombre.

end archimulti;	Fin del programa

Para describir una arquitectura podremos usar cuatro estilos, teniendo cada uno, su propio nivel de abstracción. Los estilos son:

- Estilo **behavioral** o **comportamiento**: Este estilo se caracteriza por incluir las sentencias y órdenes típicas de un lenguaje de programación (if, then, case,...), sin importar como quedará la distribución de puertas lógicas dentro del PLD. Es necesario un proceso al ser una estructura secuencial.

- Estilo **dataflow** o **flujo de datos**: Este estilo puede encontrarse de dos formas similares, pero ambas implican cómo la información será transferida de señal a señal y de la entrada a la salida sin el uso de asignaciones secuenciales. Es decir, en este estilo no se pueden usar los procesos.
- Estilo **structural** o **estructural**: En él, se describe un "netlist" de VHDL, en los cuales los componentes son conectados y evaluados instantáneamente con señales. No se suele usar este estilo únicamente en una arquitectura ya que resulta muy difícil de modificar, siendo de verdadera utilidad cuando se tiene que crear una estructura grande y se desea descomponerla en partes para manejarla mejor, y para hacer la simulación de cada parte más sencilla. Se suele requerir el uso de señales auxiliares, y además paquetes y librerías accesorios, lo cual debe estar declarado al comienzo de la entidad.
- Estilo **mixto**: Es el estilo que está compuesto en mayor o menor medida de dos o más de los estilos descritos anteriormente.

Se tiene que tener en cuenta que el código VHDL que se escriba no siempre va a describir una función optimizada, es decir que no será reducida por la herramienta de compilación, lo que se traduce en un pobre aprovechamiento de los recursos de los PLDs. Por lo tanto, diferentes diseños producen diferentes, aunque equivalentes, ecuaciones de diseño, pudiéndose dar, sin embargo, disposiciones diferentes de los recursos.

Es habitual el usar el estilo estructural para descomponer un diseño en unidades manejables, para ser creado por partes por equipos de trabajo distintos. El estilo estructural se usa además para tener un grado de control alto sobre la síntesis.

Tanto la entidad y la arquitectura deben ir unidas en el mismo archivo, ya que una parte carece de sentido sin la otra.

C.4. Paquetes

Un diseñador de hardware que utilice frecuentemente la misma tecnología de diseño puede desarrollar con el paso del tiempo, una reseña amplia de procedures, funciones, puertas y, en general, de componentes que emplea con frecuencia. Los **packages** permiten agrupar un conjunto de declaraciones para que puedan ser usadas por varios dispositivos sin ser repetidas en la declaración de cada uno.

La estructura básica en la declaración de un paquete está dividida en dos partes claramente diferenciadas:

La declaración del paquete va precedida por la palabra reservada **package** y finaliza con **end**. En el cuerpo de la declaración se encuentran los procedures, funciones, componentes, etc. tal y como aparecerían en la parte de la declaración de una entidad.

```
package nombre_del_package is

-- declaración de procedures
-- declaración de funciones
-- declaración de tipos, etc...

end nombre_del_package;
```

Esta parte es a una entidad, lo mismo que un paquete es a un programa normal en VHDL.

C. 5. Librerías

En una librería llamada se ofrecen múltiples módulos, ya creados, que pueden facilitar enormemente el trabajo, ya que incluye desde el generador de constantes más sencillo hasta contadores y multiplicadores con todas las características opcionales posibles.

Para usar cualquiera de éstos módulos, se incluye en el archivo de código la siguiente línea, encima de la declaración de entidades y de arquitecturas:

```
use work.lpmpkg.all;
```

C.6 Objetos

En un lenguaje de descripción de software (SDL) una variable contiene un valor y puede aceptar un nuevo valor a través de una asignación secuencial. Por otro lado, las constantes tienen valores prefijados a lo largo de toda la ejecución del programa. Sin embargo, en VHDL se hace necesaria la utilización de un nuevo tipo de objeto que puede emular las asignaciones concurrentes propias de los circuitos eléctricos reales; este nuevo tipo de objeto son las señales.

Un objeto en VHDL es un elemento que tiene asignado un valor de un tipo determinado. Según sea el tipo de dato, el objeto poseerá un conjunto de operaciones que se le podrán aplicar. En general, no será posible realizar operaciones entre dos objetos de distinto tipo, a menos que definamos previamente un programa de conversión de tipos.

C. 7 Identificadores

Los identificadores son un conjunto de caracteres dispuestos de una forma adecuada y siguiendo unas normas propias del lenguaje, para dar un nombre a los elementos en VHDL, por lo que es aconsejable elegir un nombre que sea representativo y que facilite la comprensión del código. Las reglas a tener en cuenta a la hora de elegir un identificador son:

- Los identificadores deben empezar con un carácter alfabético, no pudiendo terminar con un carácter subrayado, ni tener dos o más de estos caracteres subrayados seguidos.
- VHDL identifica indistintamente tanto las mayúsculas como las minúsculas, pudiéndose emplear por igual el identificador "sumador" o "SUMADOR".
- El tamaño o extensión del identificador no está fijado por VHDL, siendo recomendable que el usuario elija un tamaño que confiera sentido y

significado al identificador, sin llegar a alcanzar longitudes excesivamente largas.

- Los identificadores pueden contener caracteres numéricos del '0' al '9', sin que éstos puedan aparecer al principio.
- No puede usarse como identificador una palabra reservada por VHDL.

C.7.1 Palabras reservadas

Las palabras reservadas son un conjunto de identificadores que tienen un significado específico en VHDL. Estas palabras son empleadas dentro del lenguaje a la hora de realizar un diseño. Por esta razón y buscando obtener claridad en el lenguaje, las palabras reservadas no pueden ser empleadas como identificadores definidos por el usuario.

Las palabras reservadas por VHDL son:

Abs	else	nand	return
Access	elsif	new	select
After	end	next	severity
Alias	entity	nor	signal
All	exit	not	subtype
And	file	null	then
architecture	for	of	to
Array	function	on	transoprt
Asser	generate	open	type
attribute	generic	or	units
Begin	guarded	others	until
Block	if	out	use
Body	in	package	variable
Buffer	inout	port	wait
Bus	is	procedure	when

Case	label	process	while
component	library	range	with
configuration	linkage	record	xor
constant	loop	register	
disconnect	map	rem	
downto	mod	report	

C.7.2 Símbolos especiales

Además de las palabras reservadas empleadas como identificadores predefinidos, VHDL utiliza algunos símbolos especiales con funciones diferentes y específicas, tales como el símbolo "+" se utiliza para representar la operación suma y, en este caso, es un operador. El símbolo "- -" es empleado para los comentarios realizados por el usuario, de tal forma que el programa al encontrar una instrucción precedida por "- -" la saltará ignorando su contenido. De esta forma, el programador puede hacer más comprensible el código del programa.

Los símbolos especiales en VHDL son:

+ - / ( ) . , : ; & ' < > = | # <= => := --

El símbolo ";" debe finalizar todas y cada una de las líneas del código dando por terminada dicha sentencia en el programa.

C. 7. 3 Tipos de datos

El tipo de datos es un elemento básico en VHDL, ya que delimita que valores puede tener un objeto y que operaciones podemos realizar con él. Aparte de los tipos ya creados, se puede crear nuevos tipos y subconjuntos de tipos.

La declaración de un tipo de datos es la sentencia VHDL utilizada para introducir un nuevo tipo. Esta declaración está formada por un identificador que permitirá usar el nuevo tipo al llamarlo y la descripción del conjunto de valores que forman

el tipo de datos. Para ello se usa la palabra reservada type. La declaración puede tener varios formatos como por ejemplo:

type longitud_maxima **is range** 2 **to** 50
type estados **is** (estado_a, estado_b, estado_c);

Una vez declarado el nuevo tipo se podrá usar para declarar objetos de este tipo.

C. 7. 4. Tipos de datos predefinidos:

Tipo	Rango	Descripción
Integer	-MAXINT ... MAXINT	Números enteros
Natural	0 ... MAXINT	Números naturales
Positivo	1 ... MAXINT	Números positivos
Real	-MAXREAL ... MAXREAL	Números reales (FP)
Bolean	TRUE, FALSE	Valores boléanos
Bit	O,1	Números binarios
Bit_vector		Cadenas de bits
Character	Paquete standard	Caracteres
String		Cadena de caracteres
Time	0 a MAXTIME fs	Unidades de tiempo (ps,..,hr)

C.7.5 Operadores

El VHDL define un conjunto de operadores que se agrupan en tres categorías fundamentales: aritméticos, relacionales y lógicos.

Operadores aritméticos:

operación	operador	Tipo de datos
Suma	+	Cualquier tipo numérico
Resta	-	Cualquier tipo numérico
Producto	*	Integer, real
División	/	Integer, real
Exponenciación	**	Integer, real (exponente solo integer)
Módulo	mod	integer
Resto	rem	integer
Valor absoluto	abs	Cualquier tipo numérico

Operadores relacionales

Operación	Operador
igual	=
Diferente	/=
Mayor	>
Mayor o igual	>=
Menor	<
Menor o igual	<=

Operadores lógicos

Operación	Operador
Y	and
Y negado	nand
O	or
O negado	nor
O – exclusiva	xor
O – exclusiva negada	xnor
Negación	not

Operadores de desplazamiento

Operación	Operador
Desplazamiento lógico a la izquierda	sll
Desplazamiento lógico a la derecha	srl
Desplazamiento aritmético a la izquierda	sla
Desplazamiento aritmético a la derecha	sra
Desplazamiento circular a la izquierda	rol
Desplazamiento circular a la derecha	ror

C.7.6 Objetos de datos

Son aquellos elementos del lenguaje que permiten almacenar información. Hay tres tipos: constantes, variables y señales.

- Constantes:

Almacenan valores que no cambian a lo largo de una descripción. Las constantes de deben inicializar en el momento de su definición y en el caso de no hacerlo, el único lugar donde se pueden inicializar es en el cuerpo de un paquete. Ejemplo

```
Constant AND_DELAY: time:= 2 ns;
Constant E: real := 2.7172;
Constant  V37 : bit vector ( 7 downto 0) :=”0100110”;
.....
package body memorias is
        constant MEMORY_WORDS : ineteger := 32;
end package body memorias;
```

- Variables:

Almacenan valores que pueden cambiar a lo largo del funcionamiento. Por ejemplo:

```
Variable contador: natural;
Variable resultado: real:= 1,0;
Variable buffer: bit vector (7 downto 0);
```

Las variables solo se pueden definir dentro de un proceso y quedan restringidas a él. Se les puede asignar un valor inicial y si no toman un valor por defecto. Cuando se le asigna un valor a una variable esta cambia en forma instantánea.

```
Ej:     Temp := 7+2; - - valor de la variable Temp Igual a 9
        Temp := 7*3; - - valor de la variable Temp Igual a 21
```

- Señales:

Las señales pueden almacenar información como las variables, pero su significado está intrínsecamente ligado al lenguaje VHDL. Las señales representan las conexiones o terminales físicos de un sistema. Se pueden utilizar en todas las estructuras del lenguaje, concurrentes o secuenciales.

Signal flags: integer **range** 0 **to** 255;
Signal CLK: bit := ´0`;
Signal data_bus : bit_vector (31 **downto** 0);

Las señales no cambian de valor instantáneamente como las variables, sino que lo pueden hacer según un periodo de tiempo definido por el usuario, por ejemplo

IMPULSO <= `0´, `1´**after** 10 ns, `0´ **after** 40 ns;

La señal IMPULSO toma el valor inicial 0, luego de 10 ns toma le valor 1 y después de 40 ns vuelve a 0. El tiempo es siempre absoluto.

C. 7. 7 Atributos

Un atributo nos proporciona información sobre ciertos elementos como las entidades, arquitecturas, tipos y señales. Hay varios atributos de señales que son muy útiles en síntesis, y especialmente en el VHDL simulable. Los más usados son cuatro de ellos:

Atributo **'left**: se usa para manejar al elemento más a la izquierda de un vector.

if entrada'**left**='0' **then** ...	Si el elemento más a la izquierda de **entrada** es '0', entonces se ejecuta lo que sigue al **then**

Atributo **'right**: se usa para manejar al elemento más a la derecha de un vector.

if entrada'**right**='1' **then** ...	Si el elemento más a la derecha de **entrada** es '1', entonces se ejecuta lo que sigue al **then**

Atributo **'length**: se usa para manejar la longitud de un vector.

if entrada'**lenght**=5 **then** ...	Si la longitud de **entrada** es 5, entonces se ejecuta lo que sigue al **then**.

Atributo **'event**: se usa para conocer si una variable ha cambiado o no, soliéndose usar como variable booleana:

if entrada'**event**=5 **then** ...	Si hay un cambio en el nivel lógico de la señal **entrada**, entonces se ejecuta lo que sigue al **then**.

C. 8 Simulación

Mediante la simulación se puede verificar el comportamiento de un circuito antes de su implementación final. Para esto VHDL permite definir bancos de prueba. Estos no son más que entidades que no tienen conexiones externas sino que contienen otra entidad a la que se aplica un conjunto de estímulos o vectores de prueba para verificar su comportamiento. Un banco de pruebas debe generar los estímulos y aplicarlos a la entidad que se quiere verificar.

El siguiente es un ejemplo del código VHDL de un registro de desplazamiento sincrónico de 4 bits. [36]

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.all;

PACKAGE registros IS
  COMPONENT desplazamiento PORT(clk, rst, ed, ei : IN STD_LOGIC;
                                m : IN STD_LOGIC_VECTOR(1 DOWNTO 0);
                                e : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
                                s : BUFFER STD_LOGIC_VECTOR(3 DOWNTO 0));
  END COMPONENT;
END registros;


LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.all;
ENTITY desplazamiento IS PORT(clk, rst, ed, ei : IN STD_LOGIC;
                                m : IN STD_LOGIC_VECTOR(1 DOWNTO 0);
                                e : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
                                s : BUFFER STD_LOGIC_VECTOR(3 DOWNTO 0));
END desplazamiento;

ARCHITECTURE comportamiento OF desplazamiento IS
BEGIN
  PROCESS (clk,rst)
  BEGIN
        IF rst='1' THEN s<="0000";
        ELSIF (clk'EVENT AND clk='1') THEN
                CASE   m IS
```

```
                WHEN        "01"   => s(3)<=ei; s(2)<=s(3);   --Desplazamiento a
                           s(1)<=s(2); s(0)<=s(1); --la derecha
                WHEN        "10"   => s(3)<=s(2); s(2)<=s(1); --Desplazamiento a
                           s(1)<=s(0); s(0)<=ed;   --la izquierda
                WHEN        "11"   => s<=e;                  --Carga en paralelo
                WHEN        OTHERS => s<=s;                        --La salida no cambia
              END CASE;
        END IF;
  END PROCESS;
END comportamiento;
```

Para verificar el correcto funcionamiento del registro hay que elaborar una tabla con entradas y la respuesta prevista por el circuito. Esta formará parte del código VHDL que referencia el componente desde su biblioteca, que aplica los estímulos y verifica si las respuestas del circuito coinciden con las previstas. Un ejemplo de banco de pruebas es el siguiente:

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.all;
ENTITY test_shift is
END test_shift;


USE WORK.registros.desplazamiento;
ARCHITECTURE test OF test_shift IS
  SIGNAL clk, rst, ed, ei : STD_LOGIC;
  SIGNAL m : STD_LOGIC_VECTOR(1 DOWNTO 0);
  SIGNAL e : STD_LOGIC_VECTOR(3 DOWNTO 0);
  SIGNAL s : STD_LOGIC_VECTOR(3 DOWNTO 0);
  TYPE tipo_vtest IS RECORD
        clk, rst, ed, ei : STD_LOGIC;
        m : STD_LOGIC_VECTOR(1 DOWNTO 0);
        e : STD_LOGIC_VECTOR(3 DOWNTO 0);
        s : STD_LOGIC_VECTOR(3 DOWNTO 0);
  END RECORD;
```

```
TYPE lista_vtest IS ARRAY(0 TO 10) OF tipo_vtest;
  CONSTANT varios_v : lista_vtest :=(
        --Reset del registro
        (clk=>'0', rst=>'1', ed=>'-', ei=>'-', m=>"--", e=>"----", s=>"0000"),
        (clk=>'1', rst=>'1', ed=>'-', ei=>'-', m=>"--", e=>"----", s=>"0000"),
        (clk=>'0', rst=>'0', ed=>'-', ei=>'-', m=>"00", e=>"----", s=>"0000"),
        --Se comprueba que no se carga nada en el modo 0
        (clk=>'1', rst=>'0', ed=>'-', ei=>'-', m=>"00", e=>"1011", s=>"0000"),
        (clk=>'0', rst=>'0', ed=>'-', ei=>'-', m=>"00", e=>"1011", s=>"0000"),
        --Se carga el código 1010 en paralelo
        (clk=>'1', rst=>'0', ed=>'-', ei=>'-', m=>"11", e=>"1011", s=>"1011"),
        (clk=>'0', rst=>'0', ed=>'-', ei=>'-', m=>"11", e=>"1011", s=>"1011"),
        --Se desplaza hacia la derecha 0->101
        (clk=>'1', rst=>'0', ed=>'-', ei=>'0', m=>"01", e=>"----", s=>"0101"),
        (clk=>'0', rst=>'0', ed=>'-', ei=>'0', m=>"01", e=>"----", s=>"0101"),
        --Se desplaza hacia la izquierda 101<-1
        (clk=>'1', rst=>'0', ed=>'1', ei=>'-', m=>"10", e=>"----", s=>"1011"),
        (clk=>'0', rst=>'0', ed=>'1', ei=>'-', m=>"10", e=>"----", s=>"1011"));
  BEGIN
   --Se instancia un componente para testearlo
   registro1 : desplazamiento PORT MAP(clk=>clk, rst=>rst, ed=>ed,
                                       ei=>ei,  m=>m, e=>e, s=>s);
   --Se aplican los vectores de test y se verifica el resultado
   PROCESS
    VARIABLE vector : tipo_vtest;
    VARIABLE errores : BOOLEAN := FALSE;
    BEGIN
        FOR i IN 0 TO 10 LOOP
         vector:=varios_v(i);
         clk<=vector.clk; rst<=vector.rst; ed<=vector.ed; ei<=vector.ei;
         m<=vector.m; e<=vector.e;
         WAIT FOR 20 ns;
         IF s /= vector.s THEN  ASSERT FALSE REPORT "Salida incorrecta. ";
                                errores:=TRUE;
         END IF;
        END LOOP;
        ASSERT NOT errores  REPORT "Error en el test. ";
        ASSERT errores              REPORT "Test superado. ";
        WAIT;
  END PROCESS;
END;
```