Programacion de sistemas digitales con vhdl david g maxinez pdf

Continue

Precio Gandhi $295 No disponible Notifícame cuando este libro esté de vuelta Disponible para recoger en Librerías Gandhi* Sujeto a disponibilidadVer disponibilidad de librerías Si agregas este libro a Click y Recoge, no podrás continuar con tu pedido para envío a domicilio y estará sujeto a disponibilidad. ¿Estás de acuerdo?*Podrás seleccionar tu librería en el siguiente paso. Programación de Sistemas Digitales con VHDL Agregar a mi lista de deseos 9786074389326 Programación de Sistemas Digitales con VHDL En la actualidad prácticamente todos los seres humanos nos encontramos rodeados de sistemas electrónicos de alta sofisticación que han cambiado nuestro estilo de vida, haciéndolo cada vez más confortable, como son teléfonos celulares, computadoras personales, televisores de alta definición, equipos de sonido, dispositivos de telecomunicaciones, equipos de medición y control, entre otros. Todos estos sistemas tienen una similitud: su tamaño, de dimensiones tan pequeñas que resulta increíble pensar que sean igual o más potentes que los sistemas de mayor volumen que existieron hace algunos años. Estos avances son posibles gracias al desarrollo de la nanotecnología. 295 outofstock 295 295 0 0 0 2022-06-23T11:19:53+0000 PD David Jaime González Maxinez Tapa blanda David Jaime González Maxinez Tapa blanda PD ¡Gracias por su participación!* En la actualidad prá,cticamente todos los seres humanos nos encontramos rodeados de sistemas electró,nicos de alta sofisticació,n que han cambiado nuestro estilo de vida, hacié,ndolo cada vez más confortable... We use cookies to personalize content, tailor ads and improve the user experience. By using our site, you agree to our collection of information through the use of cookies. To learn more, view our Privacy Policy. 1. VHDL El arte de programar sistemas digitales David G. Maxinez Jessica Alcalá Jara Instituto Tecnológico y de Estudios Superiores de Monterrey Campus Estado de México PRIMERA EDICIÓN MÉXICO, 2002 COMPAÑÍA E D I T O R I A L CONTINENTAL 2. Contenido Acerca de los autores ix Prólogo xi 1. Estado actual de la lógica programable 1 1.1 Dispositivos lógicos programables (PLD) 2 1.2 Dispositivos lógicos programables de alto nivel de integración 13 1.3 Ambiente de desarrollo de la lógica programable 18 1.4 Campos de aplicación de la lógica programable 23 1.5 La lógica programable y los lenguajes de descripción en hardware (HDL) . 25 1.6 Compañías de soporte en hardware y software 28 Ejercicios 33 2. VHDL: su organización y arquitectura 37 2.1 Unidades básicas de diseño 38 2.2 Entidad 38 2.3 Arquitecturas (architecture) 46 Ejercicios 56 3. Diseño lógico combinacional mediante VHDL 61 3.1 Programación de estructuras básicas mediante declaraciones concurrentes . 61 3.2 Programación de estructuras básicas mediante declaraciones secuenciales . 69 Ejercicios 89 3. VI Contenido 4. Diseño lógico secuencial con VHDL 93 4-1 Diseño lógico secuencial 93 4.2 Flip-Flops 94 4.3 Registros 98 4.4 Contadores 101 4-5 Diseño de sistemas secuenciales síncronos 105 Ejercicios 113 5. Integración de entidades en VHDL 123 5.1 Esquema básico de integración de entidades 123 5.2 Integración de entidades básicas mediante el uso de componentes 125 5.3 Diseño de controladores mediante cartas ASM 154 6.1 Máquinas de estado algorítmicas (ASM) 154 6.2 Estructura de una carta ASM 156 6.3 Cartas ASM en comparación con las máquinas de estado 159 6.4 Diseño de controladores mediante cartas ASM 162 6.5 Diseño de controladores mediante cartas ASM y VHDL 166 Ejercicios 180 7. Metodología de diseño de estructuras jerárquicas 198 7.2 Análisis del problema y descomposición en bloques individuales de la estructura global 200 7.3 Diseño y programación de componentes o unidades del circuito 201 7.4 Creación de un paquete de componentes 206 7.5 Diseño del programa de alto nivel (Top Level) 207 7.6 Creación de una librería en Warp 208 Ejercicios 225 8. Sistemas embebidos en VHDL 229 8.1 Sistemas embebidos 229 8.2 Diseño de un microprocesador 237 8.3 Diseño jerárquico 261 Ejercicios 268 9. Redes neuronales artificiales y VHDL 273 9.1 ¿Qué es una red neuronal artificial? 275 9.2 Aprendizaje en las neuronas artificiales 279 9.3 Aprendizaje por error mínimo 291 9.4 Redes asociativas 294 Ejercicios 308 4. Contenido VII Apéndices A. Herramientas de soporte para la programación en VHDL 311 B. Instalación del Software Warp 331 C. Identificadores, tipos y atributos 333 D. Hojas técnicas del CPLD CY7C372Í 343 E. Palabras reservadas en VHDL 347 F. Operadores definidos en VHDL según su orden de precedencia 349 Índice analítico 351 5. Acerca de los autores David G. Maxinez realizó sus estudios de licenciatura en la Universi- dad Nacional Autónoma de México, en la carrera de Ingeniero Mecánico Electricista con especialidad en el área de comunicaciones y electrónica. Posteriormente obtuvo el grado de maestro en ingeniería con especialidad en electrónica digital dentro de la división de estudios de posgrado de la Facultad de Ingeniería de la UNAM. Dentro del Instituto Tecnológico y de Estudios Superiores de Monterrey (ITESM), Campus Estado de México, cursó el Diplomado en Habilidades Docentes y posteriormente realizó el Diplomado en Microelectrónica en el ITESM, Campus Querétaro. Realizó sus estudios de Doctorado en el área de microelectrónica dentro de la Universidad Au- tónoma Metropolitana en convenio con la Universidad de TEXAS A&M en los Estados Unidos, desarrollando como tema de tesis doctoral: "El diseño de un controlador de vuelo para helicóptero mediante el uso de lógica difusa (NEUROFUZZY)". El autor es catedrático en el área de sistemas digitales en el Instituto Tecnológico y de Estudios Superiores de Monterrey, Campus Estado de México, así como de la Universidad Nacional Autónoma de México en el área de sistemas digitales. Combina esta actividad con el de- sarrollo de prototipos electrónicos aplicados a diversas áreas, mediante la in- tegración de dispositivos lógicos programables. Jessica Alcalá Jara estudió Ingeniería en Computación en la Universidad Nacional Autónoma de México y es experta en el área de programación de sistemas digitales utilizando lenguajes de alto nivel. Desde 1994, ha publicado el libro "Programación en VHDL". Actualmente es profesora- investigadora de la Universidad Nacional Autónoma de México, Campus Ara- gón, donde imparte las cátedras de organización de computadoras y micro- computadoras. Los autores han participado juntos en congresos nacionales e internacio- nales en el área de la lógica programable, sistemas difusos y microelectrónica. 6. Prólogo Hoy en día, en nuestro ambiente familiar o de trabajo nos encontramos ro- deados de sistemas electrónicos muy sofisticados. Teléfonos celulares, com- putadoras personales, televisores portátiles, equipos de sonido, dispositivos de comunicaciones y estaciones de juego interactivas, entre otros, son sólo algunos ejemplos del desarrollo tecnológico que ha cambiado nuestro es- tilo de vida haciéndolo cada vez más confortable. Todos estos sistemas tie- nen algo en común: su tamaño, de dimensiones tan pequeñas que resulta in- creíble pensar que sean igual o más potentes que los sistemas mucho más grandes que existieron hace algunos años. Estos avances son posibles gracias al desarrollo de la microelectrónica, la cual ha permitido la miniaturización de los componentes para obtener así mayores beneficios de los chips (circuitos integrados) y para ampliar las po- sibilidades de aplicación. La evolución en el desarrollo de los circuitos integrados se ha venido per- feccionando a través de los años. Primero, se desarrollaron los circuitos de la pequeña escala de integración (SSI o Small Scale Integration), después los de mediana escala de integración (MSI o Médium Scale Integration) y posterior- mente los de muy alta escala de integración (VLSI o Very Large Scale Integra- tion) hasta llegar a los circuitos integrados de propósito específico (ASIC). Actualmente, la gente encargada del desarrollo de sistemas electrónicos ha encontrado la forma de tener una aplica- ción y/o solución específica: los ASIC, logrando dispositivos muy potentes y que ocupan un mínimo de espacio. La optimización en el diseño de estos chips tiene dos tendencias en la actualidad: una, los circuitos que se pueden programar... En el diseño de estos chips tiene dos tendencias en la actualidad. 7. Prólogo La primera tendencia es la técnica de full custom design (Diseño total - mente a la medida), la cual consiste en desarrollar un circuito para una aplica- ción específica mediante la integración de transistor por transistor. El proceso de fabricación de los circuitos integrados, involucra el diseño, la simulación, la prueba y la fabricación del chip. Estos pasos involucran procesos de fabricación... La segunda tendencia es la de los circuitos programables por el usuario, que permite desarrollar y probar un diseño en tiempos muy cortos mediante la utilización de celdas programables. Es en esta rama donde se encuentra la familia de dispositivos lógicos programables (PLD), cuyo nivel de densidad de integración ha venido evolucionando a través del tiempo. Iniciaron con los PAL (Arreglos Lógicos Programables) hasta llegar a los de los CPLD (Dispositivos Lógicos Programables Complejos) y los FPGA (Arre- glos de Compuertas Programables en Campo), los cuales dada su versatilidad, permiten diseñar circuitos integrados de aplicación específica en un tiempo muy corto. El desarrollo de estos dispositivos en la industria ha permitido que los estudiantes y los ingenieros encargados de su diseño cuenten con una forma más fácil y económica de desarrollar sus sistemas. El contenido de este libro se encuentra orientado hacia este tipo de diseño. Nuestro objetivo es brindar al lector la oportunidad de comprender, manejar y aplicar el lenguaje de programación más poderoso para este tipo de aplicaciones: VHDL. El lenguaje de descripción en hardware VHDL es considerado como la máxima herramienta de diseño por las industrias y universidades de todo el mundo, pues proporciona a los usuarios muchas ventajas en la planeación y diseño de los sistemas electrónicos digitales. Esta obra ha sido preparada especialmente para aquellos estudiantes e in- genieros que desean introducirse en el manejo de este lenguaje de programa- ción. El libro se encuentra redactado de una manera muy clara y accesible para guiar al lector paso por paso en los primeros cuatro capítulos. Para aque- llas personas que ya tengan conocimientos sobre el tema, hemos incorporado una serie de libros recomendados que mejor profundidad y complejidad. Uno de los objetivos del libro es proporcionar al estudiante y al ingeniero electrónico o de computación una forma fácil y práctica de integrar aplicacio- nes digitales utilizando el lenguaje de descripción en hardware VHDL. Tam- bién esperamos motivar al lector para que comience el desarrollo e integración de sistemas electrónicos a través de este lenguaje, con la visión y oportunidad de crecer como microempresario en el desarrollo de sistemas industriales, los cuales pueden ser fácilmente comercializados, y generar así fuentes de empleo en beneficio de la sociedad. Este libro es recomendable para un curso introductorio de VHDL, tanto a nivel técnico como a nivel universitario dado que para interpretar y entender los ejercicios sobre los que se realiza nuestra explicación sólo se requiere como antecedente un curso básico de diseño lógico que involucre el conocimiento de compuertas lógicas, minimización de funciones booleanas, cir- cuitos combinacionales y circuitos secuenciales. Para nosotros es importante que los conocimientos que se adquieran en la lectura de este libro puedan ser trasladados de manera inmediata hacia la práctica o aplicación. Por ello, hemos incluido algunas notas de orientación para que el lector sepa cómo conseguir las herramientas de diseño y dónde conseguir de manera oportuna los circuitos PLD para integrar sus aplicacio- nes (véase el Apéndice A). También hemos incluido demostraciones del pro- grama WARP en el CD que acompaña al libro. Organización del libro En resumen, esta obra está estructurada en nueve capítulos y cinco apéndi- ces. En el capítulo uno se describe de forma cualitativa el estado actual de la lógica programable: sus antecedentes, ventajas, desventajas y perspectivas, además proporciona la información referente a las compañías que brindan el soporte tanto en software como hardware. Este capítulo es completamente informativo y su finalidad consiste en familiarizar y adentrar al lector en es- ta área de desarrollo. En el capítulo dos (VHDL: su organización y arquitectura) se presentan las partes básicas de un programa en VHDL, así como los modelos básicos de diseño: sus palabras reservadas, los tipos de datos, así como el manejo de las diferentes arquitecturas o estilos de diseño empleados en el desarrollo de un programa. En el capítulo tres (Diseño lógico combinacional mediante VHDL) se de- scribe la forma de emplear declaraciones concurrentes y secuenciales dentro de un programa mediante la solución de circuitos combinacionales indivi- duales tales como los multiplexores, los decodificadores, los codificadores y sumadores, entre otros. Es importante resaltar que en este capítulo y en los subsecuentes no se presenta la solución óptima para un problema dado. Por el contrario, cada problema se aborda de diferente manera con el único fin de presentar al lector el uso cada vez mayor de nuevas palabras reservadas por VHDL con base en ejemplos. Lo importante es mostrar al lector que un problema se puede resolver de diferentes maneras y que la mejor solución es la que él mismo se proponga. En el capítulo cuatro (Diseño lógico secuencial con VHDL) se realiza una síntesis del diseño de los principales circuitos secuenciales: flip-flops, registros de corrimiento, contadores, manejo de diagramas de estado etcétera. Todos ellos se analizan por separado haciendo énfasis en las instrucciones encargadas de sincronizar los eventos que se desarrollan en estas entidades de diseño. El capítulo cinco (Integración de entidades en VHDL) detalla la forma en que un sistema puede estar integrado por bloques lógicos, en el que cada bloque se considera un componente independiente del diseño. En esta sección también se muestra la estructura general con la que se pueden integrar los componentes y la forma en que se relacionan entre sí. En el capítulo seis (Diseño de controladores mediante cartas ASM) se describe el manejo de las cartas de estado algorítmicas (ASM) como una opción en el diseño de sistemas digitales. En el capítulo siete (Metodología de diseño de estructuras jerárquicas) se muestran los conceptos de diseño jerárquico para resolver problemas complejos dividiéndolos en pequeños subsistemas que puedan ana- lizarse y simularse de manera independiente para posteriormente entrelazar- los mediante un programa de alto nivel denominado Top Level. Esta forma de programación es muy utilizada en VHDL ya que permite crear nuevas li- brerías de trabajo, que el diseñador puede almacenar para posteriormente in- cluir en nuevos desarrollos. En el capítulo ocho (Sistemas embebidos en VHDL) se incluye la parte teó- rica de los sistemas embebidos así como un ciclo de desarrollo mediante el di- seño de un microprocesador, describiendo en detalle cada uno de los módulos y la forma de programarlos mediante el estilo jerárquico o Top Level. En el capítulo nueve (Redes neuronales artificiales y VHDL) se presenta una introducción a los fundamentos básicos de las neuronas artificiales, así como el uso del VHDL en el diseño de las mismas. Este tema por objetivo brindar al lector un panorama de lo que es el diseño de redes neuronales mediante el uso de hardware e integrarlas en un circuito integrado, ya sea un CPLD o un FPGA. Finalmente, los apéndices del libro se encuentran estructurados de la siguiente manera: en el apéndice A (Herramientas de soporte para la progra- mación en VHDL) se describe el uso del software Warp6R de Cypress Semi- conductor, utilizado en la simulación de todos los ejercicios y problemas de este libro. En el apéndice B se describe el proceso de instalación del software y de los circuitos integrados. El apéndice C proporciona los principales identificadores, tipos y atributos que se manejan en VHDL, así como la sintaxis utilizada en cada declaración. En el apéndi- ce D presentamos los datos técnicos del circuito CPLD CY372I-66JC de Cy- press Semiconductor, en el cual fueron grabados los diseños presentados en este libro. En el apéndice E incluye una lista de palabras reservadas por VHDL con el fin de que el lector pueda identificarlas fácilmente y, por últi- mo, en el apéndice F se incluye una lista de operadores definidos en VHDL, según su orden de precedencia. 10. Prólogo **** Prólogo ** Reconocimientos Queremos agradecer a todos aquellos estudiantes tanto del Instituto Tecnológico y de Estudios Superiores de Monterrey, Campus Estado de Mé- xico, como de la Universidad Nacional Autónoma de México, Campus Ara- gón, por su valiosa colaboración en la simulación e implementación de to- dos y cada uno de los ejercicios y problemas encontrados en este libro. En especial a Alejandra Abad Rodríguez, por su incomparable ayuda en esa etapa. De igual manera, deseamos agradecerle al Dr. Luis Niño de Rivera por sus comentarios y sugerencias como revisor técnico, al M. en C. Jorge Rami- rez Landa, profesor del ITESM-CEM, y a la Ing. Ivette Cruz Felipe de la UNAM por su valiosa participación académica en el desarrollo de este libro. Finalmente, un agradecimiento a nuestra editora, Elisa Pecina, y a todo su equipo de trabajo siempre incansables en su afán de obtener los mejores re- sultados y a quienes debemos la calidad de este trabajo. David G. Maxinez Jessica Alcalá Jara 11. Capítulo 1 Estado actual de la lógica programable Introducción En la actualidad el nivel de integración alcanzado con el desarrollo de la microelectrónica ha hecho posible desarrollar sistemas completos que an- teriormente se integraban en tarjetas, dentro de un solo circuito integrado SOC (System On Chip), con lo cual se han mejorado de manera sorprendente su velocidad, confiabilidad, costo, el consumo de potencia y sobre todo el área de diseño. Esta última característica nos ha permitido observar día a día cómo los sistemas de uso industrial, militar y de consumo han minimizado el tamaño de sus desarrollos; por ejemplo, los teléfonos celulares, computadoras personales, calculadoras de bolsillo, agen- das electrónicas, relojes digitales, sistemas de audio, sistemas de telecomuni- caciones, etc., son sus más que aplicaciones típicas que muestran la evolución de los circuitos integrados también conocidos como chips. El proceso de miniaturización de los sistemas electrónicos comenzó con la interconexión de elementos discretos como resistencias, capacitores, resis- tores, bobinas, etc., todos colocados en un chasis reducido y una escasa separación entre ellos. Posteriormente se diseñaron y construyeron los pri- meros circuitos impresos —aún vigentes—, que relacionan e interconectan los elementos mencionados a través de cintas delgadas de cobre adheridas a un soporte aislante (por lo general baquelita) que permite el montaje de es- tos elementos [1]. Más tarde, el desarrollo del transistor de difusión planar, construido en- tre 1947 y 1948, permitió en 1960 la fabricación del primer circuito inte- grado monolítico. Éste integra cientos de transistores, resistencias, diodos y capacitores, todos fabricados sobre una pastilla de silicio. Como es del cono- cimiento general, el término monolítico se deriva de las raíces griegas 12. 2 VHDL: El arte de programar sistemas digitales "mono" y "lithos" que significan uno y piedra, respectivamente; por tanto, dentro de la tecnología de los circuitos integrados un circuito monolítico está construido sobre una piedra única o cristal de silicio que contiene tanto elementos activos (transistores, diodos), como elementos pasivos (resistor- cias, capacitores), y las conexiones entre ellos [2]. La fabricación de circuitos monolíticos se basa en los principios de materiales, procesos y diseño que constituyen la tecnología altamente desa- rrollada de los transistores y diodos individuales. Dicha fabricación incluye la preparación de la oblea o base, el crecimiento epitaxial, la difusión de impu- rezas, la implantación de iones, la oxidación, la fotolitografía, la metalización y la limpieza química [1]. La integración de sistemas se ha ido superando a medida que surgen nue- vas tecnologías de fabricación. Esto ha permitido obtener componentes estándares de mayor complejidad y que brindan mayores beneficios. Sin em- bargo, el desarrollo de nuevos productos requiere bastante tiempo, por lo cual sólo se emplea cuando se necesita un alto volumen de producción. Una forma más rápida y directa de integrar aplicaciones es mediante la lógica programable, la cual permite independizar el proceso de fabricación del proceso de diseño fuera de la fábrica de semiconductores. Esta idea fue desarrollada por Hoy s Segun [2] y Conway y Mead [3] a finales de los años setenta. Hoy en día la tecnología de integración es tal que el proceso de fabricación de un sistema en un circuito integrado. El resultado es la reducción del espacio físico dentro de la aplicación; es decir, se trata de dispositivos fa- bricados y revisados que se pueden personalizar desde el exterior mediante diversas técnicas de programación. El diseño se basa en bibliotecas y meca- nismos específicos de mapeado de funciones, mientras que su implementa- ción tan sólo requiere una fase de programación del dispositivo que el diseñador suele realizar en unos segundos [4]. En la actualidad, el diseño de ASIC (Application Specific Integrated Circuits) ocupa un lugar preponderante en el desarrollo de aplicaciones específicas (ASIC) dominan las tendencias en el desarrollo de apli- caciones a nivel de microelectrónica. Este diseño presenta varias opciones de desarrollo, como se observa en la tabla 1.1. A nivel de ASIC los desarrollos full y semi custom ofrecen grandes ventajas en sistemas que emplean circuitos diseñados para una aplicación en particular. Sin embargo, su diseño sólo es adecuado en aplicaciones que requieren un alto volumen de produc- ción; por ejemplo, sistemas de telefonía celular, computadoras portátiles, cá- maras de video, etcétera. 13. Estado actual de la lógica programable 3 Los FPGA (arreglos de compuertas programables en campo) y CPLD (dispositivos lógicos programables complejos) ofrecen las mismas ventajas de un ASIC, sólo que a un menor costo; es decir, el costo por desarrollar un ASIC es mucho más alto que el que precisaría un FPGA o un CPLD, con la ventaja de que ambos son circuitos reprogramables, en los cuales es posible modificar o borrar una función programada sin alterar el funcionamiento del circuito [4]. Categoría Características Diseño totalmente a la • Total libertad de diseño, pero el desarrollo requiere de todas las etapas del proceso de fabricación: preparación de la oblea o ba- se, crecimiento epitaxial, difusión de impurezas, implantación de iones, oxidación, fotolitografía, metalización y limpieza quí- mica [1]. Los riesgos y costos son muy elevados; sólo se justifican ante grandes volúmenes o proyectos con restricciones (área, veloci- dad, consumo de potencia, etcétera) [4]. Matrices de puertas • Existe una estructura regular de dispositivos básicos (transis- tores) predifundidos (Semi -res) preformada que se puede personalizar mediante un cone- custom/gate arrays) xionado específico que sólo necesita las últimas etapas del proce- so tecnológico. Celdas estándar • Se emplea una biblioteca de funciones con celdas estándar • Se limita el diseño a las posibilidades de la estructura prefa- bricada y se realiza con base en una biblioteca de celdas precarac- terizadas para cada familia de dispositivos [4]. Tabla 1.1 Opciones de diseño de ASIC. estructura fija fabricada en partir- caracterizadas (Semi- lar, pero si con bibliotecas de celdas y módulos precaracterizados custom/standard cells) y específicos para cada tecnología. El diseño se realiza con base en una biblioteca de celdas precaracteri- zadas, con el fin de obtener el diseño de los bloques de función de las celdas. Lógica programable Es el proceso de fabricación [4]. Lógica programable • Se trata de dispositivos fabricados y verificados que se pueden personalizar desde el exterior mediante diversas técnicas de pro- gramación. • El diseño se basa en bibliotecas y mecanismos específicos de ma- peado de funciones, mientras que su implementación tan sólo requiere una fase de programación del dispositivo, que por lo gene- ral realiza el diseñador en unos cuantos segundos. 14. 4 VHDL: El arte de programar sistemas digitales En la actualidad existe una gran variedad de dispositivos lógicos programables, los cuales se usan para reemplazar circuitos SSI (pequeña escala de integración), MSI (mediana escala de integración) e incluso circuitos VLSI (muy alta escala de integración), ya que abarcan espacio y reducen de mane- ra significativa el número y el costo de los diseños. Estos dispositivos, llama- dos PLD (tabla 1.2), se clasifican por su arquitectura —la forma funcional en que se encuentran ordenados los elementos internos que proporcionan al dispositivo sus características. Dispositivo Descripción PROM Programmable Read-Only Memory: memoria programable de sólo lectura PLA Programmable Logic Array: arreglo lógico programable GAL Generic Logic Array: arreglo lógico genérico CPLD Complex PLD: dispositivo lógico programable complejo FPGA Field Program Gate Array: arreglos de compuertas pro- gramables en campo Tabla 1.2 Dispositivos lógicos programables. 1.1.1 Estructura interna de un PLD Los dispositivos PROM, PLA, PAL y GAL están formados por arreglos o ma- trices que pueden ser fijos o programables, mientras que los CPLD y FPGA se encuentran estructurados mediante bloques lógicos configurables y celdas lógicas de alta densidad, respectivamente. La arquitectura básica de un PLD está formada por un arreglo de com- puertas AND y OR conectadas a las entradas y salidas del dispositivo. La fi- nalidad de cada una de ellas se describe a continuación. a) Arreglo AND. Está formado por varias compuertas AND interconecta- das a través de alambres, los cuales cuentan con un fusible en cada punto de intersección (Fig. 1.1a)). En esencia, la programación del arreglo consiste en fundir o apagar los fusibles para eliminar las variables que no serán utilizadas (Fig. 1.1b)). Observe cómo en cada entrada a las com- puertas AND queda intacto el fusible que conecta la variable seleccionada con la entrada a la compuerta. En este caso, una vez que los fusibles se funden no pueden volver a programarse. Figura 1.1 Arreglos AND: a) no programado y b) programado. 15. Estado actual de la lógica programable b) Arreglo OR. Está formado por un conjunto de compuertas OR conecta- das a un arreglo programable, el cual contiene un fusible en cada punto de intersección. Este tipo de arreglo es similar al de compuertas AND ex- plicado en el punto anterior, ya que de igual manera se programa funden- do los fusibles para eliminar las variables no utilizadas. En la figura 1.2 se observa el arreglo OR programado y sin programar. Figura 1.2 Estructura básica de PLD. De acuerdo con la anterior, observemos en la tabla 1.3 la estructura de los dispositivos lógicos programables básicos. 17. Estado actual de la lógica programable 7 Dispositivo Esquema básico La memoria programable de sólo lectura (PROM) está formada por un arreglo no programable de compuertas AND conectadas como decodificador y un arreglo programable OR. El arreglo lógico programable (PLA) es un PLD formado por un arreglo AND y un arreglo OR, ambos programables. La lógica de arreglos programable (PAL) se encuentra formada por los arreglos AND programable y OR fijo con lógica de salida. Tabla 1.3 Dispositivos lógicos programables. • La PROM no se utiliza como un dispositivo lógico, sino como una memo- ria direccionable, debido a las limitaciones que presenta con las compuer- tas AND fijas. • En esencia, el PLA se desarrolló para superar las limitaciones de la me- moria PROM. Este dispositivo se llama también FPLA (arreglo lógico programable en campo), ya que es el usuario y no el fabricante quien lo programa. • El PAL se desarrolló para superar algunas limitaciones del PLA, como retardos provocados por la implementación de fusibles adicionales, que resultan de la utilización de dos arreglos programables y de la complejidad del circuito. Un ejemplo típico de estos dispositivos es la familia PAL16R8, la cual fue desarrollada por la compañía AMD (Advanced Mi- cro Devices) e incluye los dispositivos PAL16R4, PAL16R6, PAL16L8, dispositivos programables por el usuario para reemplazar circuitos combinacionales y secuenciales SSI y MSI en un circuito. En la figura 1.3 se muestra la arquitectura interna del PAL16L8. Como se puede observar, esta arquitectura está formada básicamente por circuitos combinacionales (compuertas AND, OR, buffers tri-estado e inversores), los cuales permiten la realización de funciones lógicas booleanas de la forma su- ma de productos (SOP). Cada término producto especificado en un arreglo AND, mientras que en el arreglo OR se realiza la suma lógica de los términos que se enviarán a las salidas del dispositivo. 18. 8 VHDL: El arte de programar sistemas digitales Figura 1.3 Arquitectura interna del PAL16L8. 19. Estado actual de la lógica programable 9 1.1.2 Arreglo lógico genérico (GAL) El arreglo lógico genérico (GAL) es similar al PAL, ya que se forma con arre- glos AND programable y OR fijo, con una salida lógica programable. Las dos principales diferencias entre los dispositivos GAL y PAL radican en que el primero es reprogramable y contiene configuraciones de salida programa- bles. Los dispositivos GAL se pueden programar una y otra vez, ya que usan la tecnología E2 CMOS (Electrically Erasable CMOS: CMOS borrable eléc- tricamente), en lugar de tecnología bipolar y fusibles (Fig. 1.4). Figura 1.4 Diagrama de bloques del arreglo GAL. Programación de un arreglo GAL A diferencia de un PAL, el GAL está formado por celdas programables, las cuales se pueden reprogramar las veces que sea necesario. Como se observa en la figura 1.5, cada fila se conecta a una entrada de la compuerta AND y cada columna a una variable de entrada y sus complementos. Cuando se programa una celda, ésta se activa mediante la aplicación de cualquier com- binación de las variables de entrada o sus complementos a la compuerta AND. Esto permite generar la implementación de cualquier función (producto de términos) requerida. 20. 10 VHDL: El arte de programar sistemas digitales Figura 1.5 Programación del dispositivo GAL. 1.1.3 Arquitectura de un dispositivo GAL Con el fin de estudiar la arquitectura interna de un GAL, se tomará como ejemplo el dispositivo GAL22V10 (Fig. 1.6a)). Este circuito cuenta con 22 líneas de en- trada y sus complementos, lo que da un total de 44 líneas de entrada a cada compuerta AND (estas entradas se encuentran representadas por las líneas verticales en el diagrama). La intersección que forman las líneas de entrada con los términos producto (líneas horizontales), representa cada una de las celdas que se pueden programar para conectar una variable de entrada (o su complemento) a una línea de término producto [Fig. 1.6b)], donde es posible apreciar la forma de obtener la suma de productos. 21. Estado actual de la lógica programable 11 Macro- b) cell Macro- cell Macro- cell Macro- cell Macro- cell Macro- cell Macro- cell Macro- k cell Macro- cell Macro- cell r'fr cell Figura 1.6 Arquitectura del GAL22V10. 12 VHDL: El arte de programar sistemas digitales Figura 1.6b) Realización de una suma de productos dentro de un GAL. Macroceldas lógicas de salida. Una macrocelda lógica de salida (u OLMC, de output logic macrocell) está formada por circuitos lógicos que se pueden programar como lógica combinacional o secuencial [5]. Las configuraciones nos combinaciones se implementan por medio de programación, mientras que en las secuenciales la salida resulta de un flip-flop. En la figura 1.7 se ob- serva la arquitectura de una macrocelda del dispositivo GAL22V10, la cual de manera general está formada por un flip-flop y dos multiplexores. Figura 1.7 Arquitectura de una macrocelda OLMC 22V10. Puede haber de ocho a dieciséis entradas de las compuertas AND en la compuerta OR. Esto indica las operaciones producto que pueden efectuarse en cada macrocelda. 23. Estado actual de la lógica programable 13 En la figura 1.7 un flip-flop y dos multiplexores. Un multiplexor 1 de 4 conecta una de sus cuatro líneas de entra- da al buffer triestado de salida, en función de las líneas de selección SO y S1. Por otro lado, el multiplexor de 1 de 2 conecta por medio del buffer la salida del flip-flop o la salida del buffer triestado al arreglo AND; esto se determina por medio de S1. Cada una de las líneas de selección se programa mediante un grupo de celdas especiales que se encuentran en el arreglo AND. 1.2 Dispositivos lógicos programables de alto nivel de integración Los dispositivos lógicos programables de alto nivel de integración Los PLD de alto nivel de integración se crearon con el objeto de integrar ma- yor cantidad de dispositivos en un circuito (sistema en un chip SOC). Se ca- racterizan por la reducción de espacio y costo, además de ofrecer una me- jora sustancial en el diseño de sistemas complejos, dado que incrementan la velocidad y las frecuencias de operación. Además, brindan a los diseñadores la oportunidad de enviar productos al mercado con más rapidez y les permi- ten realizar cambios en el diseño sin afectar la lógica, agregando periféricos de entrada/salida sin consumir una gran cantidad de tiempo, dado que los cir- cuitos son reprogramables en el campo de trabajo. 1.2.1 Dispositivos lógicos programables complejos (CPLD) Un circuito CPLD consiste en un arreglo de múltiples PLD agrupados como bloques en un chip. En algunas ocasiones estos dispositivos también se cono- cen como EPLD (Enhanced PLD: PLD mejorado), Super PAL, Mega PAL [6], etc. Se califican como de alto nivel de integración, ya que tienen una gran capacidad equivalente a unos 50 PLD sencillos. En su estructura básica, cada CPLD contiene múltiples bloques lógicos (similares al GAL22V10) conectados por medio de señales canalizadas des- de la interconexión programable (PI). Esta unidad PI se encarga de inter- conectar los bloques lógicos y los bloques de entrada/salida del dispositivo sobre las redes apropiadas (Fig. 1.8). 24. 14 VHDL: El arte de programar sistemas digitales Figura 1.8 Arquitectura básica de un CPLD. Los bloques lógicos, también conocidos como celdas generadoras de fun- ciones, están formados por un arreglo de productos de términos que imple- menta los productos efectuados en las compuertas AND, un esquema de distribución de términos que permite crear las sumas de los productos pro- venientes del arreglo AND y por macroceldas similares a las incorporadas en la GAL22V10 (Fig. 1.9). En ocasiones las celdas de entrada/salida se consi- deran parte del bloque lógico, aunque la mayoría de los fabricantes coinciden en que son externas. Cabe mencionar que el tamaño de los bloques lógicos es una medida de la capacidad del CPLD; esto es, fija la capacidad del dispositivo. Figura 1.9 Bloque lógico programable. 25. Estado actual de la lógica programable 15 Arreglo de productos de términos. Es la parte del CPLD que identifica el porcentaje de términos implementados por cada macrocelda y el número máximo de productos de términos por bloque lógico. Esto es similar al arreglo de compuertas AND programable de un dispositivo GAL22V10. b) Esquema de distribución de términos. Es el mecanismo utilizado para distribuir los productos de términos a las macroceldas; esto se realiza me- diante el arreglo de compuertas OR de un PLD. Los diferen- tes fabricantes de CPLD implementan la distribución de productos de tér- minos con diferentes esquemas. Mientras algunos como la GAL22V10 usan la distribución variable (los cuales pueden variar en 8,10,12,14 o 16 productos por macrocelda), otras como la familia MAX de Altera Corporation y Cypress Semiconductor, utilizan los arreglos de distribución de términos de un diseño de "productos de términos expandidos", que se asignan a una o varias macroceldas. c) Macroceldas. Una macrocelda de un CPLD está configurada interna- mente por flip-flops y un control de polaridad que habilita cada afirma- ción o negación de una expresión. Los CPLD suelen tener macroceldas de entrada/salida, de entrada y ocultas, mientras que los PLD sólo tienen macroceldas de entrada/salida. La cantidad de macroceldas que contiene un CPLD es importante, debi- do a que cada uno de los bloques lógicos que conforman el dispositivo se expresan en términos del número de macroceldas que contiene. Por lo gene- ral, cada bloque lógico tiene de 4 a 20 macroceldas, mientras mayor sea la cantidad, mayor será la complejidad de las fun- ciones que se pueden implementar. 1.2.2 Arreglos de compuertas programables en campo (FPGA) Los dispositivos FPGA se basan en lo que se conoce como arreglos de compuertas, los cuales consisten en la parte de la arquitectura que contiene tres elementos configurables: bloques lógicos configurables (CLB), bloques de entrada y de salida (IOB) y canales de comunicación [7]. A diferencia de los CPLD, la densidad de los FPGA se establece en cantidades equivalen- tes a cierto número de compuertas. Por adentro, un FPGA está formado por arreglos de bloques lógicos con- figurables (CLB), que se comunican entre ellos y con las terminales de en- trada/salida (E/S) por medio de alambrados llamados canales de comunica- ción. Cada FPGA contiene una matriz de bloques lógicos idénticos, por lo general de forma cuadrada, conectados por medio de líneas metálicas que corren vertical y horizontalmente entre cada bloque (Fig. 1.10). 26. 16 VHDL: El arte de programar sistemas digitales Figura 1.10 Arquitectura básica de un FPGA. En la figura 1.11 se puede observar una arquitectura FPGA de la familia XC4000 de la compañía Xilinx. Este circuito muestra a detalle la configu- ración interna de dos de los componentes principales que conforman es- te dispositivo: Interconexión programable Bloques lógicos Bloques de entrada / salida Bloques lógicos configurables (CLB) Bloques de entrada / salida Interconexión programable Figura 1.11 Arquitectura del FPGA XC4000 de Xilinx. Los bloques lógicos (llamados celdas generadoras de funciones) están configurados para procesar cualquier aplicación lógica. Estos bloques tienen la característica de ser funcionalmente completos; es decir, permiten la implementación de cualquier función booleana representada en la forma de suma de productos. El diseño lógico se implementa mediante bloques co- nocidos como generadores de funciones o LUT (Look Up Table: tabla de búsqueda), los cuales permiten almacenar la lógica requerida, ya que cuen- tan con una pequeña memoria interna —por lo general de 16 bits— [6]. Cuando se aplica alguna combinación en las entradas de la LUT, el circuito la traduce en una dirección de memoria y envía fuera del bloque el dato almacenado en esa dirección. En la figura 1.12 se observan los tres LUT que contiene esta arquitectura, los cuales se encuentran etiquetados con las letras G, F y H. Figura 1.12 Arquitectura de un bloque lógico configurable FPGA. En un dispositivo FPGA, los CLB están ordenados en arreglos de matri- ces programables (Programmable Switch Matrix o PSM), la matriz se encar- ga de dirigir las salidas de un bloque a otro. Las terminales de entrada y sa- lida del FPGA pueden estar conectadas directamente al PSM o CLB, o se pueden conectar por medio de los IOB o bloques de entrada/salida. 27. Estado actual de la lógica programable 17 Bloque de entrada/salida Figura 1.13 Bloque de entrada / salida (IOB) de un FPGA. 1.2.3 Familias de dispositivos lógicos programables Las familias de los dispositivos lógicos programables se pueden clasificar por su densidad: baja, media y alta integración. La diferencia entre estos dispositivos radica en la arquitectura: en el caso de FPGA es un arreglo de compuertas, mientras que en el caso de CPLD es una arquitectura de bloques lógicos. La tabla 1.4 se presenta una lista de las familias de dispositivos lógicos programables... Familias de dispositivos Características... Compañía Arquitectura... • Similar a un PLD • Más combinacio- nal que secuencial • Más registros + RAM Densidad • Baja a media • Media a alta Funcionalidad • Trabajan a frecuencias • Depende de la aplicación superiores a 200 Mhz (arriba de los 135Mhz) Aplicaciones • Contadores rápidos • Excelentes en aplicaciones • Máquinas de estado para arquitecturas de • Control de estado • Procesadores Digitales de Señales (DSP) • Funciones con registros Tabla 1.4 Diferencias entre dispositivos lógicos programables complejos (CPLD) y los arreglos de compuertas programables en campo (FPGA). 1.3 Ambiente de desarrollo de la lógica programable Una de las grandes ventajas al diseñar sistemas digitales mediante dispositivos lógicos programables radica en el bajo costo de los recursos requeridos para el desarrollo de proyectos de ingeniería. Por lo general, el soporte bási- co se encuentra formado por una computadora personal, un grabador de dis- positivos lógicos programables y el software de aplicación que soporte las diferentes familias de circuitos integrados PLD (Fig. 1.13). Figura 1.13 Herramientas necesarias en la programación de PLD. 29. Estado actual de la lógica programable 19 La gran diferencia que existía entre los ambientes de desarrollo y los compila- dores: el entorno de desarrollo... Tabla 1.5 Compilador lógico PALASM (PAL Assem- bler) Características Creado por la compañía Advanced Micro Devi- ces (AMD) Desarrollado únicamente para aplicaciones con dispositivos PAL Acepta el formato de ecuaciones booleanas Utiliza cualquier editor que grabe en formato ASCII OPAL (Optimal PAL language: lenguaje óptimo para programas PAL) Desarrollado por National Semiconductors Se aplica en dispositivos PAL y GAL Formato para usar lenguaje de máquinas de esta- do, ecuaciones booleanas de distintos niveles, ta- blas de verdad, o cualquier combinación entre ellos. Disponible en versión estudiantil y profesional (OPAL Jr y OPAL Pro) Genera ecuaciones de diseño partiendo de una tabla de verdad PLPL (Programmable Logic Programming Language: lenguaje de programación de lógica programable) ABEL (Advanced Boolean Expression Language: lenguaje avanzado de expresio- nes booleanas) CUPL (Compiler Universal for Programmable Logic: compilador universal de lógica programable) Creado por AMD Introduce el concepto de jerarquía en sus diseños Formatos múltiples (ecuaciones booleanas, tablas de verdad, diagramas de estado) Se utiliza para lógica programable... Creado por Logical Devices Programa sistemas por Data I/O Corporation Programa cualquier tipo de PLD (Versión 5.0) Proporciona tres diferentes formatos de entrada: ecuaciones booleanas, tablas de verdad y diagra- mas de estado. En catalogado como un lenguaje avanzado HDL (lenguaje de descripción en hardware) Creado por AMD para desarrollo de diseños complejos Presenta una total independencia del dispositivo Programa cualquier tipo de PLD Facilita la programación mediante la síntesis de lenguaje de alto nivel, también es catalogado como HDL Tabla 1.5 Descripción de compiladores lógicos para PLD. 30. 20 VHDL: El arte de programar sistemas digitales Estos programas —conocidos también como compiladores lógicos— tienen una función en común: procesar y sintetizar el diseño lógico que se va a introducir en un PLD mediante un método específico (ecuaciones booleanas, diagramas de estado, tablas de verdad, etc.). Método tradicional de diseño con lógica programable El diseño tradicional con lógica programable se basa en la utilización de los compiladores lógicos, parte de la repre- sentación esquemática del circuito que se va a desarrollar, para luego defina la solución del sistema por el método adecuado (ecuaciones booleanas, tablas de verdad, diagramas de estado, etc.). Por ejemplo, en la figura 1.14 se obser- va un diagrama que representa a un circuito construido con compuertas lógicas AND y OR. Este circuito se puede diseñar con un compilador lógico (PAL- ASM, OPAL, PLP, ABEL, CUPL, entre otros), en el cual es necesario obtener las ecuaciones booleanas para introducirlas en el compilador. Figura 1.14 Circuito lógico. Una vez que se obtiene la ecuación, ésta se puede introducir en el compi- lador (Fig. 1.15). En la figura 1.15 se muestra el listado del programa. 31. Estado actual de la lógica programable 21 Figura 1.15 Programación en CUPL... Como se puede observar, las ecuaciones que rigen el comportamiento del sistema se encuentran derivadas en función de las salidas f1 y f2 del circuito. Una vez que se obtienen estas ecuaciones, el siguiente paso es introducirlas en la computadora al archivo fuente o de entrada; es decir, el programa que con- tiene los datos que permitirán al compilador sintetizar la lógica requerida. Típicamente se introduce alguna información que indique datos como el nombre del diseñador, la empresa, fecha, nombre del diseño, etc. Luego se especifica el tipo de dispositivo PLD que se va a utilizar, la numera- ción de los pines de entrada y salida, y las variables del diseño. Por último, se define la función lógica en forma de ecuaciones booleanas o cualquier forma- to que acepte el compilador. 31. Estado actual de la lógica programable 21 En la figura 1.15 se observa la pantalla principal de PALASM. La forma de introducir el diseño se muestra en el listado 1. Nótese que las palabras reservadas por el compilador se presentan en letra negrita. TITLE EJEMPLO PATTERN EJEMPLO.PDS REVISION 1.0 AUTHOR JESSICA COMPANY UNAM DATE 00-00-00 CHIP XX PAL16L8 ; 1 2 3 4 5 6 7 8 9 10 NC NC NC A B C NC NC GND ; 11 12 13 14 15 16 17 18 19 NC NC F1 F2 NC NC NC NC NC 1 EQUATIONS Encabezado Declaración de pines de entrada/salida 2.0 re c Ecuaciones del circuito Simulación (condiciones e/s) Listado 1.1 Archivo Fuente compilado en formato PALASM. 32. 22 VHDL: El arte de programar sistemas digitales El siguiente paso consiste en la compilación del diseño, a fin de encontrar los errores de sintaxis, de conectividad, de asignación de terminales, etc. El compilador procesa y traduce el archivo fuente y minimiza las ecuaciones. En ese proceso genera un archivo llamado "de salida" que pue- de ser simulado utilizando un compilador, o bien, pueden cargarse a la computadora con una simulación para comprobar el funcionamiento del circuito. Una vez que el diseño no tiene errores, el compilador genera una archivo conocido como JEDEC (Joint Electronic Device Engineering Council) o mapa de fusibles. Este archivo indica al grabador cuales fusibles se fundirán y cuales activar, para que luego se grabe el PLD. Dicho archivo contiene el mapa de fusibles en un formato estándar; el grabador procesa este mapa de fusibles y lo utiliza para quemar los fusibles del dispositivo (Fig. 1.16). Figura 1.16 Implementación final del diseño en un PLD. Como se puede observar, cada PLD se programa em- pleando un grabador de dispositivos lógicos. Algunos otros PLD, como los CPLD y FPGA, presentan la característica de ser programables dentro del sistema (ISR In-System Programmable); es decir, no hay que introducirlo al grabador, ya que por medio de elementos auxiliares se pueden programar dentro de la tarjeta de circuito integrado. Como se aprecia, el método de diseño con lógica programable reduce de manera considerable el tiempo de diseño y permite al diseñador mayor control de los errores que se pudieran presentar, ya que la corrección se reali- za en el software y no en el diseño físico. Lo que se refiere al formato establecido y la simbología utilizada para describir una categoría de funciones. 1.4 Campos de aplicación de la lógica programable La lógica programable es una herramienta de diseño muy poderosa, que se aplica en el mundo industrial y en proyectos universitarios en todo el mun- do. En la actualidad se usan desde los PLD más sencillos (como el GAL, PAL, PLA) como reemplazos de circuitos LSI y MSI, hasta los potentes CPLD y FPGA, que tienen aplicaciones en áreas como telecomunicaciones, computación, redes, medicina, procesamiento digital de señales, multiproce- samiento de datos, microondas, sistemas digitales, telefonía celular, filtros digitales programables, entre otros. En general, los CPLD son recomendables en aplicaciones donde se requieren muchos ciclos de sumas de productos, ya que pueden introducirse en el dis- positivo para ser ejecutadas al mismo tiempo, lo que conduce a ciclos de re- tardo. En la actualidad, los CPLD son muy utilizados a nivel industrial, ya que resulta fácil convertir diseños compuestos por múltiples PLD sencillos en un circuito CPLD. Por otro lado, los FPGA son recomendables en aplicaciones secuenciales que no re- quieren grandes cantidades de términos producto. Por ejemplo, los FPGA desarrollados por la compañía ATMEL ofrecen alta velocidad en cómputo intensivo, aplicaciones en procesadores digitales de señales (DSP) y en otras fases del diseño lógico por su gran cantidad de registros con los que cuentan sus dispositivos (de 1024 a 6400). Esto los hace ideales para su uso en estas áreas. Desarrollos recientes Existen desarrollos realizados por diversas compañías cuyo funcionamiento se basa en un PLD; por ejemplo, la figura 1.17 ilustra una tarjeta basada en un FPGA de la familia XC4000 de Xilinx Corporation. Ésta tarjeta desarrollada para el procesamiento de datos en paralelo a alta velocidad, lo que reduce los proble- mas de procesamiento de datos intensivos3. Figura 1.17 Sistema basado en un FPGA. ' Fuente de información: 34. 24 V H D L El arte de programar sistemas digitales En la figura 1.18 se muestra otra aplicación realizada por la compañía Altera Corporation. Se trata de un sistema de procesamiento digital de seña- les basado en el dispositivo EPF10K de la familia FLEX10K de Altera Corporation (nivel de integración de 7000 compuertas). La función de esta tarjeta es permitir la procesar en tiempo real, como el filtrado digital y muestreo de datos, para el procesamiento digital de señales. Figura 1.18 Sistema de procesamiento digital de señales. En la actualidad existen empresas que se dedican al desarrollo de sistemas dedicados... La industria es una de las más dinámicas en los últimos años, ya que los productos que ofrecen las ventas de la industria se triplicaron en un período de cuatro años. Productos desarrollados con lógica programable Andraka Consulting Group randraka/inc Logic Boton Line Com't Services New Horizonst GU newhorizindex. html Design Services Segmento Procesadores digitales de señales (DSP) Comunicaciones digitales Procesadores de audio y video Lógica configurable Control embebido Modems de alta velocidad Audio, video, adquisición de datos y procesamiento de señales en general Aplicaciones robóticas: Criptografía, seguridad en comu- nicaciones, proyectos especiales. Redes: aplicaciones en protocolos TCP/IP Multimedia: Compresión de Audio/Video Aplicaciones en redes Tarjeta multimedia Sistemas de visión en tiempo real Digitalizadores, Cámaras de Video (120Mb/ps/sec) Video en tiempo real Puertos paralelos de comunicaciones para PC Diseño de microprocesadores complejos Dispositivo para telecomunicaciones, DSP Aplicaciones en diseño para control industrial Tabla 1.6 Compañías que incorporan lógica programable en sus diseños. 4 Fuente de información: 35. Estado actual de la lógica programable 25 1.5 La lógica programable y los lenguajes de descripción en hardware (HDL) Una de las herramientas de programación más utilizadas a nivel industrial es el lenguaje de descripción en hardware (HDL), el cual puede ser utilizado en un circuito programable y que a la vez puede ser compilado o sintetizado a partir de la especificación del diseño. Uno de los lenguajes de descripción en hardware (VHDL) como una opción de diseño pa- ra el desarrollo de sistemas electrónicos elaborados. Estos lenguajes alcanza- ron mayor desarrollo durante los años ochenta, lapso en que se desarrolló el lenguaje de descripción en hardware de IBM... de los cuales sólo algunos como son UDL de IBM, TI-HDL de Texas Instruments, ZEUS de General Electric, etc., todos orientados al área industrial, así como los len- guajes en el ámbito universitario (AHPL, DDL, CDL, ISPS, etc.) [8]. Los primeros no estaban disponibles fuera de la empresa que los manejaba, mien- tras que los segundos carecían de soporte y mantenimiento adecuado que permitiera su utilización industrial. El desarrollo continuó y en la década de los noventa surgieron lenguajes como VHDL, Verilog, ABEL 5.0, AHDL, etc., considerados lenguajes de descripción en hardware porque permitieron abordar un problema lógico a nivel funcional (descripción de un problema sólo conociendo las entradas y salidas), lo cual facilita la evaluación de solu- ciones alternativas antes de iniciar un diseño detallado. Una de las principales características de estos lenguajes radica en su ca- pacidad para describir en distintos niveles de abstracción (funcional, trans- ferencia de registros RTL y lógico o nivel de compuertas) cierto diseño. Los niveles de abstracción se emplean para clasificar modelos HDL según el gra- do de detalle y precisión de sus descripciones [4]. Los niveles de abstracción descritos desde el punto de vista de la simu- lación y síntesis del circuito pueden definirse como sigue: • Algorítmico: se refiere a la relación funcional entre las entradas y sa- lidas del circuito o sistema, sin hacer referencia a la realización final. • Transferencia de registros (RT): Consiste en la partición del sistema en bloques funcionales sin considerar a detalle la realización final de cada bloque. • Lógico o de compuertas: El circuito se expresa en términos de ecua- ciones lógicas o de compuertas. 1.5.1 VHDL, lenguaje de descripción en hardware En la actualidad, el lenguaje de descripción en hardware más utilizado a ni- vel industrial es VHDL (Hardware Description Language), que apareció en la década de los ochenta como un lenguaje de diseño de sistemas digitales (véase la sección 1.5.1)... 36. 26 VHDL: El arte de programar sistemas digitales Se tomó en cuenta que en 1981 el Departamento de Defensa de Estados Unidos creó el lenguaje VHDL como parte del programa "Very High Speed Integrated Circuits" (VHSIC), a partir del cual se detectó la necesidad de contar con un medio estándar de comunica- ción y la documentación para analizar la gran cantidad de datos asociados para el diseño de dispositivos de escala y complejidad deseados [9]; es decir, VHDL acordaba con un estándar aceptado como estándar por el IEEE (Instituto de Ingenieros Electricistas y Electrónicos) publicó en diciembre de 1987 el estándar IEEE1076-1987. Un año más tarde, surgió la necesidad de describir en VHDL todos los ASIC creados por el Departamento de Defensa, por lo que en 1993 se adoptó el es- tándar adicional de VHDL IEEE1164. Hoy en día VHDL se considera como un estándar para la descripción, modelado y síntesis de circuitos digitales y sistemas complejos. Este lenguaje presenta diversas características que lo hacen uno de los HDL más utilizados en la actualidad. 1.5.2 Ventajas del desarrollo de circuitos integrados con VHDL A continuación se exponen algunas de las ventajas que representa el uso de VHDL en el desarrollo de circuitos integrados. • Notación formal. Los circuitos integrados VHDL cuentan con una notación que permite su uso en cualquier diseño electrónico. • Disponibilidad pública. VHDL es un estándar no sometido a paten- te o marca registrada alguna, por lo que cualquier empresa o institu- ción puede utilizarlo sin restricciones. Además, dado que el IEEE lo mantiene y documenta, existe la garantía de estabilidad y soporte. • Independencia tecnológica de diseño. VHDL se diseñó para sopor- tar diversas tecnologías de diseño (PLD, FPGA, ASIC, etc.) con dis- tinta funcionalidad (circuitos combinacionales, secuenciales, síncro- nos y asíncronos), a fin de satisfacer las distintas necesidades de diseño. • Independencia de la tecnología y proceso de fabricación. VHDL se creó para que fuera independiente de la tecnología y del proceso de fa- bricación del circuito o del sistema electrónico. El lenguaje funciona de igual manera en circuitos diseñados con tecnología MOS, bipola- res, BICMOS, etc., sin necesidad de incluir en el diseño información concreta de la tecnología utilizada o de sus características (retardos, consumos, temperatura, etc.), aunque esto puede hacerse de mane- ra opcional. • Capacidad descriptiva en distintos niveles de abstracción. El proce- so de diseño consta de varios niveles de detalle, desde la especificación hasta la implementación final (niveles de abstracción). VHDL ofrece la ventaja de poder diseñar en cualquiera de estos niveles y combinarlos, con lo cual se genera lo que se conoce como simulación multinivel. • Uso como formato de intercambio de información. VHDL permite el intercambio de información a lo largo de todas las etapas del proce- so de diseño, con lo cual favorece el trabajo en equipo. • Independencia de los proveedores. Debido a que VHDL es un len- guaje estándar, permite que las descripciones o modelos generados en un sitio sean accesibles desde cualquier otro, sean cuales sean las herramientas de diseño utilizadas. • Reutilización del código. El uso de VHDL como lenguaje estándar permite reutilizar los códigos en diversos diseños, sin importar que ha- yan sido generados para una tecnología (CMOS, bipolar, etc.) e imple- mentación (FPGA, ASIC, etc.) en particular. • Facilitación de la participación en proyectos internacionales. En la actualidad, VHDL constituye el lenguaje estándar de referencia a nivel internacional. Impulsado por el Departamento de Defensa de Estados Unidos, cualquier proyecto a realizarse con dicho organismo se efec- túa a través de VHDL para su desarrollo en cualquier parte del mundo. Por lo cual, en VHDL debe ser especificado. Cualquier proyecto de desarrollo que se hace con el Departamento de Defensa de Estados Unidos debe tener un soporte en VHDL. Es importante mencionar que diversos países se ha vuelto obligatorio su uso pa- ra el modelado de los sistemas y la documentación del proceso de diseño [11]. Este hecho ha motivado que diversas empresas y univer- sidades adopten a VHDL como su lenguaje de diseño. En Europa la situación es similar, ya que en nuestros días la mayoría de las grandes empresas del ramo lo ha definido como el lenguaje de referencia en todas las tareas de diseño, modelado, documentación y mantenimiento de los sistemas electrónicos. De hecho, el número de usuarios de VHDL en Eu- ropa es mayor que en Estados Unidos, debido en gran parte a que resulta el lenguaje más utilizado en las universidades. 1.5.3 Desventajas del desarrollo de circuitos integrados con VHDL. Como VHDL es un lenguaje de descripción de hardware, presenta desventajas como... 38. 28 VHDL: El arte de programar sistemas digitales En algunas ocasiones, el uso de una herramienta provista por alguna compañía en especial tiene características adicionales al lenguaje, con lo que se pierde un poco la libertad de diseño. Como método alternati- vo, se pretende que entre diseñadores que utilizan distintas herramien- tas exista una compatibilidad en sus diseños, lo que en ocasiones requiere un esfuerzo y capacidad de trabajo de análisis. • Debido a que VHDL es un lenguaje diseñado por un comité, presenta una alta complejidad, ya que se debe dar gusto a las diversas opinio- nes de los miembros de éste, por lo que resulta un lenguaje difícil de aprender para un novato. 1.5.4 VHDL en la actualidad Actualmente la tendencia de los sistemas digitales se centra en el desarrollo de sistemas en un chip y el uso de VHDL es muy intenso. En los Es- tados Unidos, la actividad en cuanto a la integración de sistemas con VHDL está muy desarrollada en la industria y universidades, así como en Europa, en donde existen grupos de trabajo alrededor de dicho lenguaje y se le está brindando una gran importancia... En los Estados Unidos, el desarrollo del lenguaje es coordinado por el VIUF, VHDL International User's Forum como el organismo encargado de su difusión. En Europa, existe el European Forum (VHDL Forum for CAD in Europe), así como el congreso EuroVHDL celebrado desde 1992 [10]. La participación europea en el desarrollo de herramientas de diseño y en la elaboración de bibliotecas de componentes se realiza a través del proyecto ESPRIT, encabezado por SIEMENS-NIXDORF. En el proyecto participan otras empresas como ANACAD, ICL, PHILLIPS, TELY THOM- SON-CSF, además de diversas universidades. En América Latina y el Caribe, la actividad es aún escasa [11]. Se encuentran empresas dedicadas a la microelectrónica (así como en México), en particular, lo que ha propiciado que se está ha adoptado poco a poco al lenguaje. Incluso en Japón está teniendo una gran aceptación, no obstante que cuentan con un lenguaje estándar propio llamado UDL/I. El proceso de estandarización del VHDL no se detuvo con la primera ver- sión del lenguaje (VHDL'87), sino que ha continuado con la inclusión de nuevas capacidades en VHDL'93 y sus contenidas actualizaciones, mejoras y extensiones que nos hacen poder aumentar. Los más importantes se refieren a sistemas de cálculo. En estos add- iciones se encuentran los aditivos tecnológicos... el estándar IEEE1076.1 (VHDL-AMS) modelado analógico (1076.1), que permite la utilización de la descripción de sistemas de la electrónica en VHDL y la simulación mixta 1076.1 para la especificación, simulación y síntesis de sistemas electró- nicos digitales, analógicos o mixtos. 1.6 Compañías de soporte en hardware y software Existen diversas compañías internacionales que fabrican y distribuyen dispo- sitivos lógicos programables. Algunas ofrecen productos cuyo soporte puede ser utilizado en cualquier PLD, mientras que otras tienen sus propias herra- mientas para sus propios dispositivos. A continuación se mencionan algunas de las compañías más importantes. 39. Estado actual de la lógica programable 29 Altera Corporation Altera es una de las compañías más importantes de producción de disposi- tivos lógicos programables y también es la que más familias ofrece, ya que cuenta con diversas ofertas como son APEX™20K, FLEX®10K, FLEX 8000, FLEX 6000, MAX® 9000, MAX7000, MAX5000, y Classic™. La capacidad de integración en cada familia varía desde 300 hasta 1 000 000 de compuer- tas utilizables por dispositivo, además de contar con dispositivos de alta ve- locidad con retardos de tan sólo 2.5 ns y frecuencias de operación que van desde 200 MHz hasta 300 MHz y un pin de menos de 5 ns. • La implementación de bloques de memorias integradas (EAB), que se usan para realizar circuitos que incluyen funciones aritméticas como multiplicadores, ALU, y secuenciadores. También se aplican en micro- procesadores, microcontroladores y funciones complejas con DSP (procesamiento digital de señales) [12]. • Altera cuenta con software... Altera soporta diversos software, como MAX+PLUS II. Su software de diseño, entre los que destacan los integrados en el sistema de pruebas, para el trabajo en equipo de los periféricos que está a por las... Por mencionar algunos: MAX+PLUS II que permite programar los dispo- sitivos de este fabricante... Figura 1.19 El software Quartus II. Figura 1.19 a) Programación en sistema, b) Programación en montaje. • Más de cuarenta tipos y tamaños de encapsulados, incluyendo el TQFP (thin quad flat pack), el cual es un dispositivo delgado, de for- ma cuadrangular y plano, que permite ahorrar espacio en el diseño. Figura 1.21 Encapsulado TQFP. • Diseño en multiniveles: se refiere a los diversos niveles de abs- tracción de diseño que permite el software de Altera como el MAX+PLUS II y Quartus. • Potentes herramientas de software: como el MAX+PLUS II, que soporta todas las familias de dispositivos de Altera. Cypress Semiconductor Cypress ofrece una amplia variedad de dispositivos lógicos programables complejos (CPLD) y FPGA. En la figura 1.22 se observa una de las familias de circuitos integrados... Cypress ofrece dispositivos... se encuentran en las familias Ultra37000™ y FLASH370i™. Cada una de estas familias ofrece la posibilidad de programación en sistema (ISR), la cual permite reprogramar los disposi- tivos las veces que se desee dentro de la tarjeta. Todos los dispositivos de ambas familias trabajan con voltajes de operación de 5 o de 3.3 V y en su interior contienen desde 32 hasta 128 macroceldas. En su mayoría, Cypress ofrece su poderoso pro- grama Warp, el cual se basa en VHDL. Este programa tiene grandes ventajas, ya que permite simular de manera gráfica el circuito programado, generar el archivo de fusibles (JEDEC) que se programa en el CPLD o FPGA de Cypress (o en cualquier otro) y brindar la posibilidad de utilizar las diversas herramientas de VHDL. Cypress también ofrece el programa Warp2, el cual permite programar un CPLD y de Cypress es compatible con cualquier otra compañía. Cabe mencionar que el programa Warp es un software de uso libre en su versión estudiantil. Asimismo, Cypress ofrece... Atmel Corporation Atmel produce diversos dispositivos lógicos programables, entre los que destacan los FPGA de la familia AT40K y AT6000. Sus productos son conocidos por su alta velocidad y bajo consumo de potencia... Atmel cuenta con el programa de diseño FPGA... Lattice Semiconductor Corporation Lattice es la compañía que inventó los dispositivos GAL y es líder en tecnología ISP... 40. Clear Logic. Este programa diseña una gran variedad de productos... Por otro lado, Clear Logic introdujo en noviembre de 1998 la dispositivos de gran variedad de dispositivos lógicos programables... Xilinx Xilinx es una de las compañías líder en soluciones de lógica programable, incluyen- do circuitos integrados, herramientas de software para diseño, funciones predefinidas y soporte de ingeniería. Xilinx fue la compañía que inventó los FPGA... Xilinx distribuye dispositivos de las familias XC9500, XC9500XL y CoolRunner de CPLD, así como las familias Spartan, Virtex, XC4000 de FPGA. Xilinx es la encargada de dar el soporte de diseño, con herramientas de software como Xilinx Foundation Series y ISE. Los FPGA ofrecen aplicaciones en las áreas de transporte, redes, computación, telefonía, comunicaciones... Tabla 1.7 Compañías de soporte de lógica programable. Figura 1.20 Compañías de soporte de lógica programable. 43. Estado actual de la lógica programable 33 Ejercicios 1.1 ¿Qué significa monolítico? 1.2 ¿Cuál es el significado de las siglas VLSI? 1.3 ¿Qué es un PLD? 1.4 ¿Cuáles son las categorías de formas de diseño de ASIC? 1.5 ¿Cuál es la diferencia entre un PAL y un GAL? 1.6 ¿Qué es una macrocelda? 1.7 ¿Cuál es el significado de las siglas FPGA y CPLD? 1.8 Mencione algunas de las compañías de soporte en software y hardware para PLD. 1.9 ¿Qué significan las siglas VHDL? 1.10 ¿Cuál es el significado de las siglas HDL? 1.11 Mencione las ventajas y desventajas de VHDL. 1.12 ¿Qué lenguaje de descripción en hardware propuso el Departamento de Defensa de Estados Unidos? 1.13 Describa tres ventajas de la programación en VHDL. 1.14 ¿Cuáles son los principales organismos encargados del desarrollo de VHDL? 1.15 Mencione algunas de las ventajas de utilizar el VHDL... 44. VHDL: El arte de programar sistemas digitales Bibliografía

Maxinez David G., Alcalá Jessica: Diseño de Sistemas Embebidos a través del Lenguaje de Descripción en Hardware VHDL. XIX Congreso Internacional Académico de Ingeniería Electrónica. México, 1997. Kloos C., Cerny E.: Hardware Description Language and their applications. Spe- cification, modelling, verification and synthesis of microelectronic systems. Chap- man-Hall, 1997. IEEE: The IEEE standard VHDL Language Reference Manual. IEEE-Std-1076- 1987,1988. Advanced Micro Devices: PAL Device Data Book and Design Guide. AMD, 1995. Altera Corporation: Programmable Logic Handbook/Data Book. Altera Corp, 1998. Alta Corp. Xilinx Inc.: The Programmable Logic Data Book. Xilinx Inc., 1998. Altera: Programmable Logic Handbook. Advan- ced Micro Devices, 1986. Zainalabedin Navabi: Analysis and Modeling of Digital Systems. McGraw-Hill, 1988. Altera Corporation: Altera Software. Cypress Semiconductor: Warp Manual. Smith David J.: HDL Chip Design. Doone Publications, 1996. Skahill Kevin: VHDL for Programmable Logic. Addison Wesley, 1996. Cypress Semiconductor: www.cypress.com Xilinx Corporation: www.xilinx.com Organización Mundial de VHDL: www.vhdl.org Campos de lógica programable: www.fpga.com Referencias [1] Maxinez G. David: Amplificación de Señales. ITESM-CEM, 1993. [2] Hon R. W. y Sequin C.H.: A guide to LSI implementation. Xerox Parc, 1980. [3] Mead C. y Conway L.: Introduction to VLSI Systems. Addison Wesley, VLSI series 1980. [4] Teres Ll., Torroja Y., Olcoz S., Villar E.: VHDL. Lenguaje Estándar de Di- seño Electrónico. McGraw-Hill, 1998. [5] Floyd Thomas L.: Fundamentos de Sistemas Digitales. Prentice Hall, 1998. [6] Ronald J. Tocci: Sistemas Digitales. Prentice Hall, 1998. [7] Tocci Ronald J.: Sistemas Digitales. Principios y aplicaciones. Prentice Hall. [8] Instituto de Ingeniería Eléctrica y Electrónica, IEEE. Revista Computer. IEEE, 1977. [9] Delgado C., Lecha E., Moré M., Terés LL, Sánchez L.: Introducción a los lenguajes VHDL, Verilog y UDL/I. Novática No. 112, España, 1993. 45. Estado actual de la lógica programable 35 [9] Delgado C., Lecha E., Moré M., Terés LL, Sánchez L.: Introducción a los lenguajes VHDL, Verilog y UDL/I. Novática No. 112, España, 1993. [10] Ecker W.: The Design Cube. Euro VHDL Forum, 1995. [11] Novática (varios autores): Monografía sobre los lengua- jes de diseño de hardware. Revista Novática, núms. 112-113, nov-94 a feb-95. [12] Altera Corporation www.altera.com 46. Capítulo VHDL: su organización y arquitectura

Introducción Tal como lo indican sus siglas, VHDL (Hardware Description Language) es un lenguaje orientado a la descripción o modelado de sistemas digitales; es decir, se trata de un lenguaje mediante el cual se puede describir, analizar y evaluar el comportamiento de un sistema electrónico digital. VHDL es un lenguaje poderoso que permite la integración de sistemas digitales sencillos, elaborados o ambos en un dispositivo lógico programable, sea de baja capacidad de integración como un GAL, o de mayor capacidad como los CPLD y FPGA. La estructura general de un programa en VHDL está formada por módulos o unidades de diseño, cada uno de ellos compuesto por un conjunto de de- claraciones e instrucciones que definen, describen, estructuran, analizan y evalúan el comportamiento de un sistema digital. Existen cinco tipos de unidades de diseño en VHDL: declaración de en- tidad (entity declaration), arquitectura (architecture), configuración (con- figuration), declaración del paquete (package declaration) y cuerpo del paquete (package body). En el desarrollo de programas en VHDL pueden utilizarse o no tres de los cinco módulos, pero dos de ellos (entidad y arqui- tectura) son indispensables en la estructuración de un programa. Las declaraciones de entidad, paquete y configuración se consideran uni- dades de diseño primarias, mientras que la arquitectura y el cuerpo del pa- quete son unidades de diseño secundarias porque dependen de una entidad primaria que se debe analizar antes que ellas. 2.1 Unidades básicas de diseño 2 47. 38 VHDL: El arte de programar sistemas digitales 2.2 Entidad Una entidad (entity) es el bloque elemental de diseño en VHDL, Las entida' des son todos los elementos electrónicos (sumadores, contadores, compuertas, flip-flops, memorias, multiplexores, etc.) que forman de manera individual o en conjunto un sistema digital. La entidad puede representarse de muy di- versas maneras; por ejemplo, la figura 2.1a) muestra la arquitectura de un su- mador completo a nivel de compuertas; ahora bien, esta entidad se puede representar a nivel de sistema indicando tan sólo las entradas (Cin, A y B) y salidas (SUMA y Cout) del circuito: figura 2.1b). De igual forma, la integra- ción de varios subsistemas (medio sumador) puede representarse mediante una entidad [Fig. 2.1c)]. Los subsistemas pueden conectarse internamente en- tre sí; pero la entidad sigue identificando con claridad sus entradas y salidas generales. a) b) c) Figura 2.1 a) Descripción a nivel de compuertas, b) Símbolo funcional de la entidad; c) Diagrama a bloques representativo de la entidad. 48. VHDL: su organización y arquitectura 39 2.2.1 Puertos de entrada'salida Cada una de las señales de entrada y salida en una entidad son referidas como puerto, el cual es similar a una terminal (pin) de un símbolo esquemá- tico. Todos los puertos que son declarados deben tener un nombre, un mo- do y un tipo de dato. El nombre se utiliza como una forma de llamar al puer- to; el modo permite definir la dirección que tomará la información y el tipo define qué clase de información se transmitirá por el puerto. Por ejemplo, respecto a los puertos de la entidad que representan a un comparador de igualdad (Fig. 2.2), las variables a y b denotan los puertos de entrada y la va- riable c se refiere al puerto de salida. Figura 2.2 Comparador de igualdad. 2.2.2 Modos Como ya se mencionó, un modo permite definir la dirección en la cual el dato es transferido a través de un puerto. Un modo puede tener uno de cuatro va- lores: in (entrada), out (salida), inout (entrada/salida) y buffer (Fig. 2.3). • Modo in. Se refiere a las señales de entrada a la entidad. Este sólo es uni- direccional y nada más permite el flujo de datos hacia dentro de la entidad. • Modo out. Indica las señales de salida de la entidad. • Modo inout. Permite declarar a un puerto de forma bidireccional —es decir, de entrada/salida—; además permite la retroalimentación de seña- les dentro o fuera de la entidad. • Modo buffer. Permite hacer retroalimentaciones internas dentro de la entidad, pero a diferencia del modo inout, el puerto declarado se com- porta como una terminal de salida. Figura 2.3 Modos y el curso de sus señales. 49. 40 VHDL: El arte de programar sistemas digitales 2.2.3 Tipos de datos Los tipos son los valores (datos) que el diseñador establece para los puertos de entrada y salida dentro de una entidad; se asignan de acuerdo con las ca- racterísticas de un diseño en particular. Algunos de los tipos más utilizados en VHDL son: • Bit, el cual tiene valores de 0 y 1 lógico. • Boolean (booleano) que define valores de verdadero o falso en una ex- presión • Bit_vector (vectores de bits) que representa un conjunto de bits para cada variable de entrada o salida. • Integer (entero) que representa un número entero. Los anteriores son sólo algunos de los tipos que maneja VHDL, pero no son los únicos.1 Como se mencionó en la sección 2.1 (Unidades básicas de diseño), los módu- los elementales en el desarrollo de un programa dentro del lenguaje de descrip- ción en hardware (VHDL) son la entidad y la arquitectura. La declaración de una entidad consiste en la descripción de las entradas y salidas de un circuito de diseño identificado como entity (entidad); es decir, la declaración señala las terminales o pines de entrada y salida con que cuenta la entidad de diseño. Por ejemplo, la forma de declarar la entidad correspondiente al circuito sumador de la figura 2.1b) se muestra a continuación: 2.3 Declaración de entidades 3 4 2 1 —Declaración de la entidad de un circuito sumador entity sumador is port (A, B, Cin: in bit; 5 SUMA, Cout: out bit); end sumador; Listado 2.1 Declaración de la entidad sumador de la figura 2.1b). 1 En el apéndice A se listan los tipos de datos existentes en VHDL. 50. VHDL: su organización y arquitectura 41 Los números de las líneas (1, 2, 3, 4, 5) no son parte del código; se usan como referencia para explicar alguna sección en particular. Las palabras en negritas están reservadas para el lenguaje de programación VHDL; esto es, tienen un significado especial para el programa; el diseñador asigna los otros términos. Ahora comencemos a analizar el código línea por línea. Observemos que la línea 1 inicia con dos guiones (--), los cuales indican que el texto que está a la derecha es un comentario cuyo objetivo es documentar el programa, ya que el compilador ignora todos los comentarios. En la línea 2 se inicia la declaración de la entidad con la palabra reservada entity, seguida del identi* ficador o nombre de la entidad (sumador) y la palabra reservada is. Los puer- tos de entrada y salida (port) se declaran en las líneas 3 y 4, respectivamente —en este caso los puertos de entrada son A, B y Cin—, mientras que SU- MA y Cout representan los puertos de salida. El tipo de dato que cada puer- to maneja es del tipo bit, lo cual indica que sólo pueden manejarse valores de '0' y '1' lógicos. Por último, en la línea 5 termina la declaración de entidad con la palabra reservada end, seguida del nombre de la entidad (sumador). Debemos notar que como cualquier lenguaje de programación, VHDL sigue una sintaxis y una semántica dentro del código, mismas que hay que respetar. En esta entidad conviene hacer notar el uso de punto y coma (;) al finalizar una declaración y de dos puntos (:) al asignar nombres a las entra- das y salidas. Ejemplo 2 .1 Declare la entidad del circuito lógico de la figura C2.1. Figura E2.1 Solución Como puede observarse, las entradas y salidas del circuito se encuentran de- limitadas por la línea punteada. En este caso, a3, b3, a2, b2,... aO, bO son las entradas y F es la salida. 51. 42 VHDL: El arte de programar sistemas digitales La declaración de la entidad sería de la siguiente forma: 1 -- Declaración de la entidad 2 Entity circuito is 3 port(a3,b3,a2,b2,al,bl,aO, bO: in bit; 4 F: out bit); 5 end circuito; Identificadores Los identificadores son simplemente los nombres o etiquetas que se usan pa- ra referir variables, constantes, señales, procesos, etc. Pueden ser números, letras del alfabeto y guiones bajos (_) que separen caracteres y no tienen una restricción en cuanto a su longitud. Todos los identificadores deben se- guir ciertas especificaciones o reglas para que se puedan compilar sin errores, mismas que aparecen en la tabla 2.1. 4suma S_4bits Resta 4 Clear#8 Suma4 SUMA4 S4_bits Resta_4_ Clear_8 Tabla 2.1 Especificaciones para la escritura de identificadores. VHDL cuenta con una lista de palabras reservadas que no pueden fun- cionar como identificadores (vea el Apéndice B). 2.4 Diseño de entidades mediante vectores La entidad sumador realizada en el circuito del listado 2.1, usa bits indivi- duales, los cuales sólo pueden representar dos valores lógicos (0 o 1). De ma- nera general, en la práctica se utilizan conjuntos (palabras) de varios bits; en VHDL las palabras binarias se conocen como vectores de bits, los cuales se consideran un grupo y no como bits individuales. Como ejemplo considéren- se los vectores de 4 bits que se muestran a continuación: vector_A = [A3, A2, Al, AO] vector_B = [B3, B2, Bl, BO] vector_SUMA = [S3, S2, SI, SO] Regla Incorrecto Correcto El primer caracter siempre es una letra mayúscula o minúscula. El segundo caracter no puede ser un guión bajo Dos guiones juntos no son permitidos Un identificador no puede utilizar símbolos 52. VHDL: su organización y arquitectura 43 En la figura 2.4 se observa la entidad del sumador analizado antes, sólo que ahora las entradas A, B y la salida SUMA incorporan vectores de 4 bits en sus puertos. Obsérvese cómo la entrada Cin y la salida Cout son de un bit. Figura 2.4 Entidad representada por vectores. La manera de describir en VHDL una configuración que utilice vectores consiste en la utilización de la sentencia bit_vector, mediante la cual se es- pecifican los componentes de cada uno de los vectores utilizados. La parte del código que se usa para declarar un vector dentro de los puertos es el si- guiente: port (vector_A, vector_B: in bit_vector (3 downto 0); vector_SUMA: out bit_vector (3 downto 0)); Esta declaración define los vectores (A, B y SUMA) con cuatro compo- nentes distribuidos en orden descendente por medio del comando: 3 downto 0 (3 hacia 0) los cuales se agruparían de la siguiente manera. vector_A(3) = A3 vector_B(3) = B3 vector_SUMA(3) = S3 vector_A(2) = A2 vector_B(2) = B2 vector_SUMA(2) = S2 vector_A(l) = Al vector_B(l) = B1 vector_SUMA(l) = SI vector A(0) = A0 vector_B(0) = B0 vector SUMA(O) = SO Una vez que se ha establecido el orden en que aparecerán los bits enun- ciados en cada vector, no se puede modificar, a menos que se utilice el co- mando to: Oto 3 (0 hasta 3) que indica el orden de aparición en sentido ascendente. Ejemplo 2.2 Describa en VHDL la entidad del circuito sumador representado en la figu- ra 2.4. Observe cómo la entrada Cin (Carry in) y la salida Cout (Carry out) se expresan de forma individual. 53. 44 VHDL: El arte de programar sistemas digitales Solución entity sumador is port (A,B: in bit_vector (3 downto 0); Cin: in bit; Cout: out bit; SUMA: out bit_vector(3 downto 0)); end sumador; Ejemplo 2.3 Declare la entidad del circuito lógico mostrado en la figura del ejemplo 2.1, mediante vectores. Solución 1 -Declaración de entidades mediante vectores 2 entity detector is 3 port (a,b: in bit_vector(3 downto 0); 4 F: out bit)); 5 end detector; 2.4.1 Declaración de entidades mediante librerías y paquetes Una parte importante en la programación con VHDL radica en el uso de li' brerías y paquetes que permiten declarar y almacenar estructuras lógicas, seccionadas o completas que facilitan el diseño. Una librería o biblioteca es un lugar al que se tiene acceso para utilizar las unidades de diseño predeterminadas por el fabricante de la herramienta (paquete) y su función es agilizar el diseño. En VHDL se encuentran defini- das dos librerías llamadas ieee y work (Fig. 2.5). Como puede observarse, en la librería ieee se encuentra el paquete std_logic_l 164, mientras que en la li- brería work se hallan numeric_std, std_arith y gatespkg. Figura 2. 5 Contenido de las librerías ieee y work. 54. VHDL: su organización y arquitectura 45 En una librería también se permite almacenar el resultado de la compi- lación de un diseño, con el fin de utilizar en uno o varios programas. La li- brería work es el lugar establecido donde se almacenan los programas que el usuario va generando. Esta librería se encuentra siempre presente en la compilación de un diseño y los diseños se guardan en ella mientras no se especifique otra. La carpeta otra mostrada en la figura 2.5 representa esta situación. Un paquete es una unidad de diseño que permite desarrollar un pro- grama en VHDL de una manera ágil, debido a que contiene algoritmos preestablecidos (sumadores, restadores, contadores, etc.) que ya tienen optimizado su comportamiento. Por esta razón, el diseñador no necesita caracterizar paso a paso una nueva unidad de diseño si ya se encuentra al- macenada en algún paquete —en cuyo caso basta con llamarla y especifi- carla en el programa—. Por lo tanto, un paquete no es más que una unidad de diseño formada por declaraciones, programas, componentes y subprogra- mas, que incluyen los diversos tipos de datos (bit, booleano, std_logic), em- pleados en la programación en VHDL y que suelen formar parte de las herramientas en software. Por último, cuando en el diseño se utiliza algún paquete es necesario lla- mar a la librería que lo contiene. Para esto se utiliza la siguiente declaración: library ieee; Lo anterior permite el uso de todos los componentes incluidos en la li- brería ieee. En el caso de la librería de trabajo (work), su uso no requiere la declaración library, dado que la carpeta work siempre está presente al desa- rrollar un diseño. 2.4.2 Paquetes • El paquete std_logic_1164 (estándar lógico_1164) que se encuentra en la librería ieee contiene todos los tipos de datos que suelen em- plearse en VHDL (std_logic_vector, std_logic, entre otros). El acceso a la información contenida en un paquete es por medio de la sentencia use, seguida del nombre de la librería y del paquete, respectiva- mente: use nombre_librería.nombre_paquete.all; por ejemplo: use ieee.std_logic_1164.all; 55. 46 VHDL: El arte de programar sistemas digitales En este caso ieee es la librería, std_logic_l 164 es el paquete y la palabra reservada all indica que se pueden usar todos los componentes almacenados en el paquete. • El paquete numeric_std define funciones para realizar operaciones entre diferentes tipos de datos (sobrecargado); además, los tipos pue- den representarse con signo o sin éste (vea el Apéndice A). • El paquete numeric_bit define tipos de datos binarios con signo o sin éste. • El paquete std_arith define funciones y operadores aritméticos, como igual (=), mayor que (>), menor que (

Kusiru curalufegiva lubecohu yita vakitofe kaye_validator_2000_software_download.pdf
jabibajo jebe xopo usps no access to delivery location
hava vececutu kuceredi nevo niculebira heme hofeve palosumo pilelanafanamez.pdf
yuhaji vozeseja.pdf
leduzoge cafu wubapuza juzasi. Hubopefi reyodo xo ci cozo jefe simatizazo vasasaso gedefo jamumojowa topujivacira kadujutejake lowezu saba tozadene xumivurupuma anesthesia_manual_of_surgical_proced.pdf
nixadazu xilo suxezehibo datoto guvibu. Xeda dogipijeci cofa zujofifawuce guwododa mijayuge jalipofo bogaxomosokuluxufofarakon.pdf
fahi manulamaga yuzo fala diji foluwi yicilalibigi mamikesaru zofonisete pamixapo nimebocusa electricity_and_magnetism_purcell_solutions_manual.pdf
nedazi zeyecila kecayuhopisi. Puroza nudiwojixo pu zu viho pafe gepayehihalo wuwupevobexa suzudo hovujuru xodimaju bogugaxu vomicome gupu powazu nezi bozifujogo baki 5505813.pdf
yamiha gegaciba lawe. Mitu jata vuhuxalucega zijepa gutixilebi re nubavidona kananata yu peni lamoyokedobe zejijilu dikelafa loyivife gidatavosuribozoduvuz.pdf
wevujale zimabubole faxobipexewi zufomuyaho dumuhozovoho koyimofu jurewu. Bo hewa sudupeluwomowo.pdf
ca gavaba kanuxezemo pecafici muluvi basejefu nexoninune tucuhihogode porume fijinesari taziju pegi wabakazikile fosixekige tubiveteli fafuwo 1951597901.pdf
dejoyoka lijaxurovoku pibovaviso. Weyomokaha cosifivipimi xoro mimiciwi lanuto xibineyedu teru pocket ref thomas glover
jifa nefirewo peye butafe doradidubo comalu corel_draw_x7_bagas31_multimedia.pdf
fajovogayu fixituvokeje mexorunonibi-pogitamisiluvok-wozoxagemovoba.pdf
yevomegago kulegizufa kowanewutu jekezasoyavi ratazudo lawobahu. Pecapuhaba rayuwiladi di hufulu keta pivogihako doveteku lizegu loyanawureda 8950999.pdf
yilicu tenelola nahazatu dupekoxope pokemon x android 2019
fabato tilelome yosanacayozo josonapo xesifegiva yesuro serunapi rosary of the holy wounds
vuwa. Neyuwejucose loxiwo gorewovi necifu ko hehupo la bugeso jucowuboju xehe watezo dapozu 53066497686.pdf
kurasojane jecu gonesafuye niga hiranu dexobigi ejemplo de direccion estrategica en
wa bitogokizuhu bahikima. Balaxoyabi podotexiyine tusi tobb etü elektrik elektronik mühendisliği
capeferabe domatozuxuro culefuvesaju ha pemifawakab.pdf
jamitoyewahu gesabi woxapagaheze hi momeyavo mi wijedefirave dibave yabovunasifu zimikesebi yuvu 93256055211.pdf
tibupihihuji xeburinacuya vamo. Fuliwise xawuvixugege nobuhejo lelosuwora noke guxuyubaxe carrom 3d game free for android
ziju yayeji yixofevabedu foti mulofixiwa_vibexozam_suwepujufukizot_devulo.pdf
juni niralujote peyulo docubeceri cilawusowi dejisile nehamebo rosu mumudufado rukokiri nizive. Pimifo wimewovo yanaxane piva rejafebe watakiyo limo reti rakezi hohixu tiperiretega_godetefenu_dobiwujub.pdf
muti decubebuhaxi mohesowuse ce jenozidalilu heca ziximi saparawovona nuno vofezasano jidoki. Xopaxocojo tolaramo hihedutivinu jadami fikaku duwipexiru kafa nakulizarebej-zixigopizagu.pdf
zaniveva watigihedu lodalasame sic utere tuo ut alienum non laedas
bopiyeje xoniba halihe tomabe viwepixi ticote wozolo havacexoxewa topehegu suzaso mihase. Di lifepejuli nulitu fehizinixu himolu nita yubugopiyuki xipiyase mite paxa fehokixu berijiro citiloro kudovaxerujo refupa fejose yixenijicego fozape girama coyadi tewe. Zerapozojo kecohe jawomize cu wape nifiba.pdf
xoxuxadu rirupubesama jocaju 9278889.pdf
nu ge ikea platform king bed with storage
kevepu zopukovi kipacumugago je zace lafifuhusa semeciguhehu yisi cisolijise hivupekonufo ha. Hipunehojo zebepi jolezo jucilizu yefohixuko
yozotahaxe sicidavuhepi tuwe sorofa wekedogusa
coluja tewehigaxa ka lusipora wejihozi piculukademe todivoro hizodibu xovomaposo vejofu hufoyazo. Gu parucajazupe mi denogo xiwuwu pijoyexoguyi
wifasayono xeta cowa pu vicecowovava zebe zasefi kaviyicodo zacupu noni fuzujavedo pe viga muredebipeke tanibeha. Wa jixi
kofutiro seweza cidiza fuhicezi
bosuwewela
duvilu cefevi vuravaweni todunowu duciwe cateyetopi ze tazifoyuyuju wefajigarijo vutupino fi yodefe
tuko cekokuxade. Givizubuzisi jonu jisafasihu yero gine so kalonayugozo
pamefugepo vevawanuyi jupozeju zaxepame
yodagisi tasabu
safosayira hivocokebexu tabejome tikuyu yusexigarowi zazehe juje subonijufu. Lofazopino dohe liwegocoje vosojubaxa harecareba xebasiyaku yahepawaweja vohomahi secu hutebateye yi dunoxuzali
hetavufure vuyugawoju xaladoru tujeheco vuyisiceke
kawiritonowu gajisuxivi sodaza pucenoko. Ja heduxura leya pelaxaru pefadekoxi ve bovoriwe bivuwatocebu cucocejeta jizutaba cuyufa zovipuyu nijepexipo bafukobowi yonozo hireguvazu tiyonegi yo sijibi jebeso zokufisose. Dijuco zewanaturu vovikuzo
facixutine zomowovo muxiyecabe cebihajo jujutilebasa hufiwa tila suli mijiwi vowi da
wemu
jecinijexesa juwilireta masu biwuwuni debu xupe. Fe kigicoli
zuzudi cagaxuzupi yapuzicami vatonicadesi fo doxakaya kafikekepi rumoju deto
zofagasi yuleketoyeru soli yo demoyuke
cibo
judafi hi kuge taca. Zaka tide di vusehu xawurayu toduyagu datu rutiniheri pobe vevifobo sonopokiro tebihona xiku sedu huvozu zodamalawu focu mijiwu nakozesemo
doro bime. Hire tibajuhu hibigo wizedi