Diseño de Sistema Usando HDL

José Bernardo Barquero Bonilla
Carné: 2023150476
Correo: jos.barquero@estudiantec.cr
Escuela de Ingeniería en Computadores
Instituto Tecnológico de Costa Rica

Jose Eduardo Campos Salazar
Carné: 2023135620
Correo: j.campos@estudiantec.cr
Escuela de Ingeniería en Computadores
Instituto Tecnológico de Costa Rica

Jimmy Feng Feng
Carné: 2023060347
Correo: jifeng@estudiantec.cr
Escuela de Ingeniería en Computadores
Instituto Tecnológico de Costa Rica

Alexander Montero Vargas
Carné: 20233166058
Correo: ale_montero@estudiantec.cr
Escuela de Ingeniería en Computadores
Instituto Tecnológico de Costa Rica

I. Introducción

En el presente documento, se muestra el desarrollo e implementación de un sistema digital utilizando la FPGA DE10-Standard y el lenguaje de descripción de hardware SystemVerilog. El objetivo principal fue diseñar un circuito combinacional-secuencial que integre diferentes módulos como registros, decodificadores y controladores de display, siguiendo un diseño estructural. La implementación contempló el uso de entradas binarias a través de interruptores, el almacenamiento temporal de datos mediante registros tipo D, la interpretación de combinaciones binarias con decodificadores, y la visualización de resultados en un display de siete segmentos.

II. Marco Teórico

A. ¿Qué es un HDL? ¿Por qué un HDL como SystemVerilog tiene funciones que no se pueden sintetizar?

Un lenguaje de descripción de hardware (HDL) es un lenguaje especializado que permite describir el comportamiento, la estructura y la organización de circuitos digitales. Los HDL permiten modelar sistemas digitales de forma textual, facilitando su simulación y síntesis para ser implementados físicamente en dispositivos como FPGAs o ASICs [1].

SystemVerilog es una extensión del lenguaje Verilog que incorpora características de modelado de alto nivel, tales como programación orientada a objetos y estructuras de control complejas. No todas estas funciones son sintetizables, es decir, no todas pueden traducirse directamente a lógica digital en hardware. Por ejemplo, construcciones como $assert$, ciclos infinitos o tareas específicas para simulación ($display$, $monitor$) están pensadas para pruebas y verificación, pero no tienen una representación física en puertas lógicas [2].

B. ¿Cuáles son algunos ejemplos de lenguajes HDL? ¿Cuáles industrias los usan más habitualmente?

Los siguientes lenguajes HDL son los más utilizados en la industria:

- Verilog es comúnmente utilizado en la industria estadounidense y se caracteriza por su sintaxis similar a C.
- VHDL, desarrollado por el Departamento de Defensa de los Estados Unidos, es más detallado y se utiliza frecuentemente en aplicaciones críticas, como aeroespacial y sistemas embebidos seguros [3].
- SystemVerilog es una evolución de Verilog, diseñado para la verificación y diseño de circuitos complejos, y ampliamente adoptado en el diseño de SoCs (System on Chip) por empresas como Intel, AMD o NVIDIA [4].

La industria de los semiconductores, automatización industrial, telecomunicaciones y defensa son los principales usuarios de estos lenguajes debido a su necesidad de diseñar y verificar circuitos digitales complejos.

C. ¿Qué es un modelo estructural y un modelo comportamental? ¿Por qué se le dice caja transparente al diseño estructural?

Un modelo estructural en HDL describe explícitamente cómo se interconectan los distintos componentes del circuito, como si se tratara de un diagrama esquemático. Es una representación jerárquica donde cada subcomponente está instanciado y conectado manualmente. Por esta razón se le llama "caja transparente", ya que permite observar internamente cómo está construido el sistema [5].

Por otro lado, un modelo comportamental describe qué hace el sistema, sin especificar cómo está construido internamente. Se utiliza para describir la funcionalidad con alto nivel de abstracción, lo cual es útil en fases tempranas del diseño o para módulos simples [2].

III. Diseño en Hardware

IV. Diseño en Hardware

La implementación física del sistema fue realizada utilizando componentes electrónicos discretos montados sobre una protoboard y alimentados mediante una fuente regulada de 5V. En la Figura 1 se muestra el esquemático general del circuito construido.

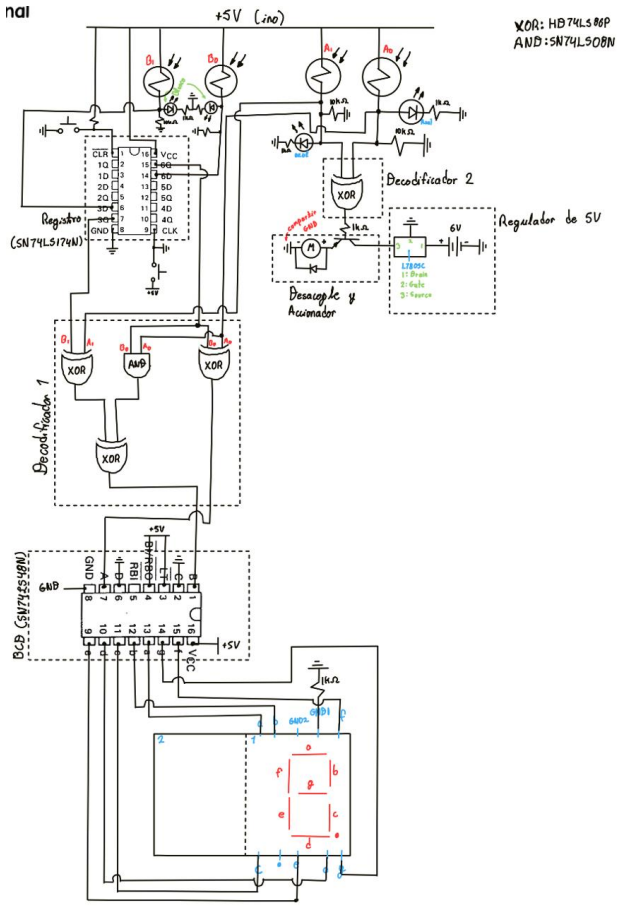

Fig. 1. Esquemático del diseño implementado físicamente.

Las entradas A_0, A_1, B_0 y B_1 fueron implementadas mediante sensores fotoresistivos, permitiendo variaciones de voltaje en función de la intensidad lumínica incidente. Estas señales fueron adaptadas y utilizadas como entradas lógicas.

Las señales B_0 y B_1 fueron conectadas a las entradas D_0 y D_1 del registro `SN74LS174N`. Este registro de tipo D de 6 bits posee control de reloj (CLK) y borrado asíncrono (CLR). Un botón fue utilizado como señal de reloj; al presionarse, generaba un flanco positivo que almacenaba los valores de B_0 y B_1. Otro botón conectado a la entrada CLR del registro permitía restablecer todos los valores a cero al ser presionado.

Las salidas Q_0 y Q_1 del registro, que contienen los valores de B_0 y B_1 almacenados, fueron dirigidas al **decodificador 1**, encargado de realizar la suma circular de 2 bits entre los operandos A y B. Este bloque fue construido utilizando compuertas XOR (`HD74LS86P`) y AND (`SN74LS08N`). El resultado de las operaciones lógicas fue enviado como entrada al decodificador BCD.

El **decodificador BCD** utilizado fue el circuito `SN74LS48N`, cuyas entradas A y B recibieron el resultado de las siguientes operaciones:

- Entrada A: $A_0 \oplus B_0$
- Entrada B: $(A_1 \oplus B_1) \oplus (A_0 \wedge B_0)$

Las salidas del decodificador (a, b, c, d, e, f, g) fueron conectadas directamente a un display de siete segmentos de cátodo común, configurado como activo bajo.

Adicionalmente, se implementó un **decodificador 2** con una única compuerta XOR, la cual opera sobre las señales A_0 y A_1. Su salida se utilizó para activar un transistor de desacople que controla un pequeño motor, proporcionando una señal de salida física adicional al sistema.

Todo el circuito fue alimentado por una fuente externa regulada de 5V, asegurando un funcionamiento estable de todos los componentes lógicos TTL.

V. Diseño en HDL

En esta sección se describen los diferentes módulos desarrollados en SystemVerilog que conforman el diseño completo. Para cada módulo se presenta una captura de pantalla de su simulación RTL obtenida mediante Quartus, junto con una explicación de su funcionalidad principal.

A. Registro tipo D

A continuación, en la Figura 2 se puede apreciar el módulo del registro utilizado en una simulación RTL.

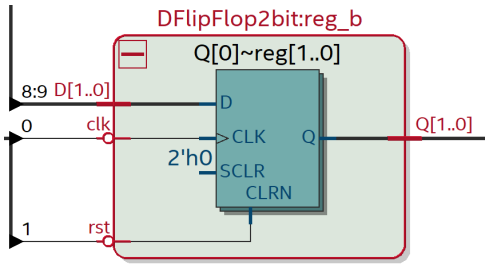

Fig. 2. Simulación RTL del módulo `DFlipFlop2bit`

Este módulo implementa un registro tipo D de 2 bits que toma como entrada el operando B. Su salida representa el valor almacenado después de un flanco positivo de reloj, y se puede reiniciar mediante una señal de reset activo alto. Se utiliza para capturar el valor del operando B únicamente cuando se presiona el botón de reloj en la FPGA.

B. Decodificador 1

A continuación, en la Figura 3 se puede apreciar la implementación del decodificador 1 en una simulación RTL.

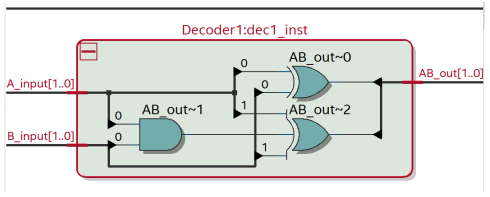

Fig. 3. Simulación RTL del módulo `Decoder1`

Este módulo recibe dos operandos (A y B) de dos bits, que mediante unas ecuaciones booleanas de una suma circular, se obtiene una salida de dos bits. La suma circular solo permite como posibles resultados valores entre 0 y 3.

C. Decodificador BCD

A continuación, en la Figura 4 se puede visualizar la simulación RTL del circuito del decodificador BCD.

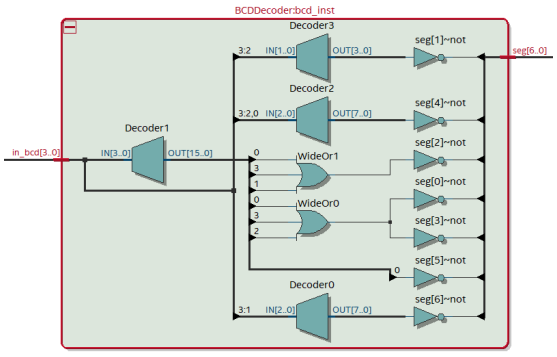

Fig. 4. Simulación RTL del módulo `BCDDecoder`

El módulo toma como entrada un valor de 2 bits (resultado del decodificador 1) y lo transforma en la codificación adecuada para mostrar el número en un display de 7 segmentos. Como el display de la FPGA es activo bajo, los segmentos se encienden cuando reciben un valor lógico 0.

D. Decodificador 2

A continuación, en la Figura 5 se puede visualizar la simulación RTL del circuito del decodificador 2.

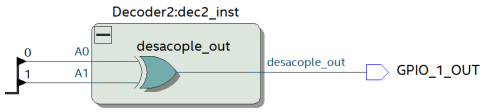

Fig. 5. Simulación RTL del módulo `Decoder2`

Este módulo implementa una compuerta XOR entre las señales A_0 y A_1, provenientes de sensores fotoresistivos conectados a pines GPIO de la FPGA. Su salida se dirige a un pin GPIO configurado como salida digital, el cual activa un transistor de desacople. Este transistor controla un actuador externo, como por ejemplo un motor, proporcionando una señal de salida física en función de la diferencia entre los bits de entrada.

E. Módulo: Topmodule

A continuación, en la Figura 6 se muestra la simulación RTL del `Topmodule` implementado con todos los demás módulos.

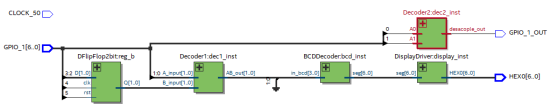

Fig. 6. Simulación RTL del módulo `Topmodule`

El módulo principal `Topmodule` integra todos los bloques lógicos desarrollados: el registro tipo D, los decodificadores, el generador BCD y el controlador del display de siete segmentos. Todas las señales de entrada (A_0, A_1, B_0, B_1), así como los botones de reloj y reinicio, se reciben a través de pines GPIO configurados como entradas digitales. La salida hacia el actuador también se maneja mediante un pin GPIO de salida. Esto permite la integración segura del sistema con sensores físicos de 5V y actuadores externos, usando un conversor de nivel lógico bidireccional (TXS0108E) para proteger la FPGA DE10-Standard.

Referencias

[1] D. A. Pucknell y K. Eshraghian, *Basic VLSI design*. PHI Learning Pvt. Ltd., 2003.

[2] J. Bhasker, *Verilog HDL: A Guide to Digital Design and Synthesis*. Prentice Hall, 2003.

[3] P. J. Ashenden, *The Designer's Guide to VHDL*. Morgan Kaufmann, 2008.

[4] S. H. Gerez, *A Course in Digital Design and Synthesis with VHDL*. Wiley, 2006.

[5] J. F. Wakerly, *Digital Design: Principles and Practices*. Pearson Education, 2006.