Laboratorio 1

Introducción a los lenguajes de descripción de hardware

Emerson Yaveth Monge Hernández
Gabriel González Muñoz

CE 3201 — Taller de Diseño Digital

Luis Barboza

22 de agosto de 2025

1. Introducción

Los lenguajes de descripción de hardware (HDL) permiten modelar sistemas digitales mediante dos enfoques complementarios. El modelado de comportamiento describe, tal como su nombre lo indica, cómo se comporta o qué hace un determinado módulo. Por otro lado, el modelado de estructura, describe un módulo complejo y cómo este está constituido de partes más sencillas. Ambos paradigmas son esenciales para diseñar sistemas jerárquicos y modulares.

Una vez descrito el sistema en HDL, el proceso de síntesis lógica transforma este código en un "mapa" que describe el hardware y sus interconexiones. Este "mapa" puede ser un archivo de texto o también puede ser mostrado en forma de esquemático de forma que sea posible visualizar el circuito que se programó. Este flujo de diseño es fundamental para implementar sistemas digitales en dispositivos de lógica programable [1].

Entre estos dispositivos, los FPGAs pertenecen a la familia de componentes con lógica programable. La base de esta lógica programable se sustenta en una matriz de bloques lógicos configurables. Estos bloques a su vez están enlazados por medio de una red de interconexiones completamente reprogramable. Celdas de memoria controlan los bloques lógicos así como las conexiones de forma que el componente logre cumplir con los requerimientos. También se tienen bloques de entrada/salida (I/O) para conexiones externas, mostrar resultados, etc. Esta arquitectura flexible es la que permite su adaptabilidad a una enorme gama de aplicaciones [2].

Actualmente, los modelos de FPGA más utilizados en la industria provienen de AMD (Xilinx) con sus familias Virtex, Kintex, Artix, Spartan y Zynq, muy populares por cubrir desde aplicaciones de bajo costo y consumo hasta sistemas de alto rendimiento e integración con procesadores ARM. Intel Agilex destaca en centros de datos y telecomunicaciones por su gran capacidad de procesamiento y flexibilidad, mientras que Lattice es preferida en IoT y dispositivos de borde por su eficiencia energética. Finalmente, Microchip/Actel con sus PolarFire e IGLOO son comunes en aeroespacial y defensa por su fiabilidad y tolerancia a entornos extremos.

Gracias a sus características únicas, en la actualidad las FPGA son utilizadas para el procesamiento de señales digitales (DSP), principalmente en telecomunicaciones y aplicaciones de imagen y video. Otro uso común es la aceleración de cómputo, se utilizan como aceleradores de hardware en varios centros de datos (Azure, Amazon, AWS) [2]. En sistemas críticos

como lo pueden ser aeroespaciales y de defensa, cumplen un rol importante pues se pueden reprogramar tras su fabricación, aumentando su vida útil [1].

2. Desarrollo

En este capítulo se presentan las soluciones propuestas y aplicadas a cada uno de los problemas del laboratorio, describiendo el razonamiento de diseño seguido para obtener los circuitos correspondientes.

2.1. Problema 1: Conversión de código binario a código Gray

Para resolver este problema se utilizó la **Propuesta 1**, basada en las reglas de conversión entre código binario y Gray. El bit más significativo se conserva directamente, mientras que los demás se generan mediante la operación lógica **XOR** entre pares de bits adyacentes. La elección del operador XOR se fundamenta en que su comportamiento equivale a una suma sin acarreo, lo que cumple con la definición del código Gray. De esta forma, se diseñó un circuito que implementa la conversión utilizando compuertas XOR en cascada.

Cuadro 1: Tabla de verdad del operador XOR

A	B	Z
0	0	0
0	1	1
1	0	1
1	1	0

2.2. Problema 2: Restador completo de 4 bits

La solución implementada corresponde a la **Propuesta 1**, la cual parte de la construcción de un **restador completo de 1 bit**. Este circuito posee tres entradas: el minuendo (A), el sustraendo (B) y el *borrow in*, así como dos salidas: el resultado (R) y el *borrow out*. A partir de la tabla de verdad se determinó la expresión lógica de cada salida, obteniendo un diseño compuesto por compuertas XOR, AND, OR y NOT. Finalmente, para obtener el restador de 4 bits, se conectaron en cascada cuatro restadores de 1 bit, propagando el *borrow* a través de cada etapa.

Cuadro 2: Tabla de verdad del restador completo de 1 bit

Bin	A	B	Bout	R
0	0	0	0	0
0	0	1	1	1
0	1	0	0	1
0	1	1	0	0
1	0	0	1	1
1	0	1	1	0
1	1	0	0	0
1	1	1	1	1

2.3. Problema 3: Contador de 4 bits con reset asincrónico

Para este problema se aplicó la **Propuesta 1**, en la cual se construyó un contador empleando **flip-flops tipo T** conectados en cascada y activados en el flanco de bajada del reloj. El primer flip-flop recibe la señal de reloj principal, mientras que los siguientes se activan con la salida del flip-flop anterior. Esta configuración permite que las salidas cambien de manera secuencial, generando el conteo binario. Adicionalmente, se incluyó una señal de **clear (CLR)** para reiniciar el contador a cero cuando sea necesario.

Cuadro 3: Tabla de verdad simplificada del contador de 4 bits con CLR

Q0	Q1	Q2	Q3	CLR
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	1	1	1	1
1	1	0	0	1
1	1	0	1	1
1	0	1	0	1
1	0	1	1	1
⋮	⋮	⋮	⋮	⋮
0	0	0	0	0

3. Resultados y Análisis

3.1. Problema 1

En la Figura 1 se presenta la simulación realizada en *ModelSim*, donde se observa la conversión de números binarios a su respectiva codificación Gray. La tabla generada por el testbench confirma que los resultados obtenidos coinciden con la teoría, ya que el bit más significativo se conserva y los siguientes se obtienen mediante la operación lógica `XOR` entre bits adyacentes.

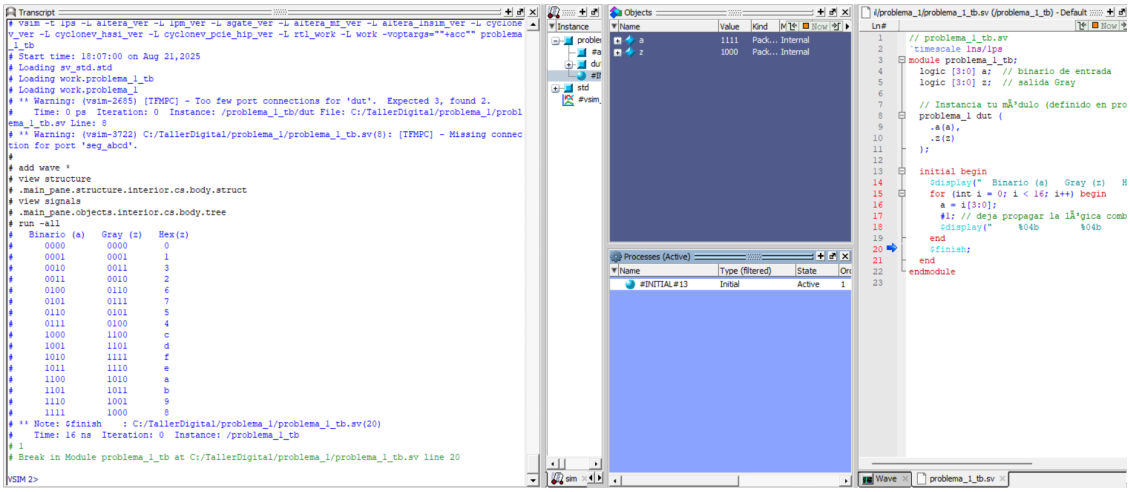

Figura 1: Simulación en ModelSim del conversor Binario a Gray.

Posteriormente, el diseño fue cargado en la FPGA, utilizando los switches como entradas binarias y el display de 7 segmentos como salida codificada. En la Figura 2 se muestra el resultado para una entrada específica, donde la visualización del valor `F` en el display corresponde al valor esperado según la codificación Gray.

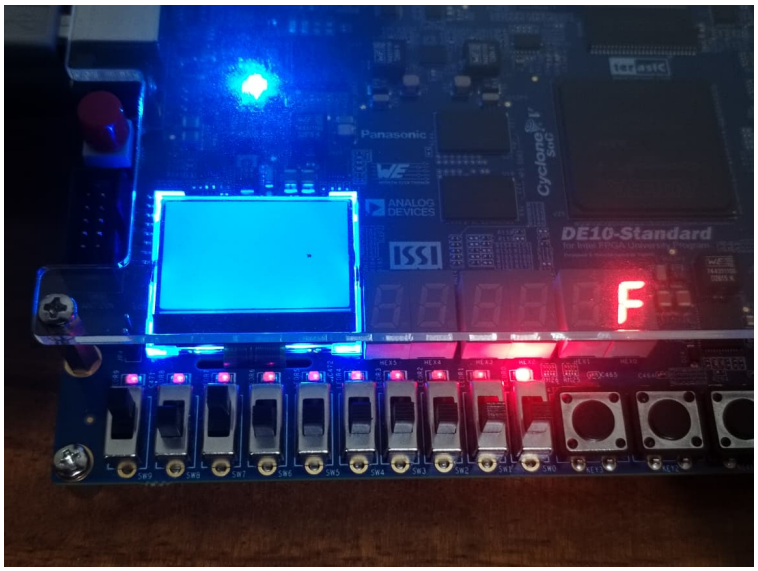

Figura 2: Implementación en FPGA mostrando el resultado en el display de 7 segmentos.

Para este problema, los resultados de la simulación y de la prueba práctica son coherentes con la teoría. La implementación confirma que la conversión de binario a Gray mediante compuertas lógicas `XOR` es exitosa y cumple con el objetivo del problema.

3.2. Problema 2

En la figura 3 se presenta la simulación realizada para cuatro pares de operandos así como el valor de C_{in} y C_{out} con cada uno de ellos. La simulación es exitosa gracias a que puede apreciarse que los resultados de cada par de operandos es correcto.

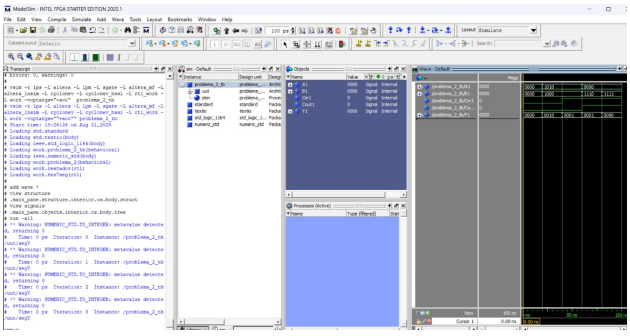

Figura 3: Simulación en ModelSim del restador completo de 4 bits.

Una vez la simulación fue exitosa se procedió a implementar la solución en la FPGA utili-

zando los switches como entradas binarias y tres displays de 7 segmentos como salida que muestran minuendo, sustraendo y diferencia respectivamente y en hexadecimal. En la Figura 4 se muestra el resultado para $10_{(10)} - 8_{(10)}$ apreciándose en el display de resultado el valor esperado ($2_{(10)}$).

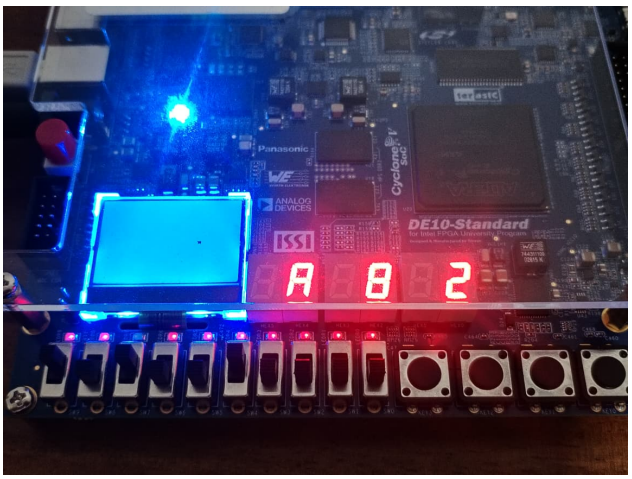

Figura 4: Implementación en FPGA del restador mostrando minuendo, sustraendo y diferencia respectivamente.

Como pudo apreciarse tanto simulación como implementación concuerdan con la teoría esperada para un restador completo cumpliendo así el objetivo del problema al haber diseñado e implementado un restador completo de 4 bits con modelado estructura en VHDL.

3.3. Problema 3

En lo que respecta a este problema, por parte de la simulación hubo éxito parcial, pudo implementarse de manera exitosa un self-checking testbench; sin embargo, en cuanto a simulación no hubo éxito con seis bits (ver Fig. 7) debido a que los ticks de reloj y los tiempos no se ajustan del todo bien al ser un incremento no lineal respecto a dos y cuatro bits los cuales sí pudieron simularse a la perfección como puede apreciarse en las figuras 5 y 6 respectivamente. Afortunadamente, el problema sólo reside en la simulación dado que la implementación en FPGA es totalmente funcional.

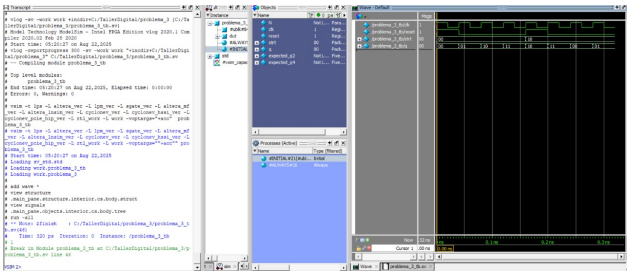

Figura 5: Simulación del contador parametrizable con 2 bits.

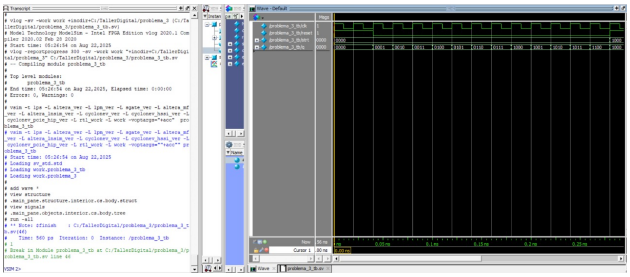

Figura 6: Simulación del contador parametrizable con 4 bits.

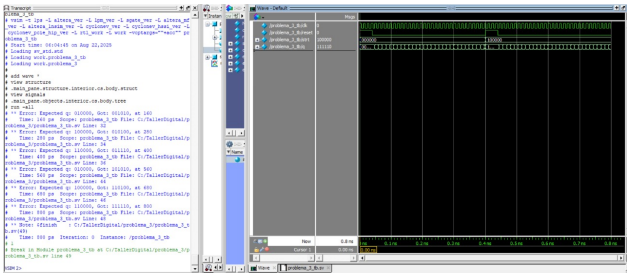

Figura 7: Simulación del contador parametrizable con 6 bits.

Referencias

[1] E. Monmasson and M. Cirstea, "FPGA Design Methodology for Industrial Control Systems—A Review," *IEEE Transactions on Industrial Electronics*, vol. 54, pp. 1824–1842, 7 2007.

[2] K. Vipin and S. A. Fahmy, "FPGA dynamic and Partial reconfiguration," *ACM Computing Surveys*, vol. 51, pp. 1–39, 7 2018.