Tarea I: Introducción a diseño digital en HDL

Dr.-Ing Alfonso Chacón-Rodriguez

1. Introducción

La demanda del mercado actual en la microelectrónica exige diseños digitales altamente complejos, que para implementarlos se requiere de ayuda asistida por computadora. Las herramientas EDA aumentan la productividad de los ingenieros al abstraer detalles de la implementación de los sistemas digitales por medio de un lenguaje de descripción de hardware (HDL) más natural para describir funcionalidades y comportamientos.

Los lenguajes HDL se utilizan para la síntesis de diseños digitales para ser fabricados en silicio o en FPGA. Las FPGAs, son circuitos integrados reconfigurables que de acuerdo su programación implementan una especificación funcional a partir de un código HDL.

2. Objetivo general

Introducir al estudiante al desarrollo de un sistema digital utilizando lenguajes de descripción de hardware.

3. Objetivos específicos

1. Elaborar una implementación de un diseño digital en una FPGA.
2. Construir un *testbench* básico para validar las especificaciones del diseño.
3. Implementar el algoritmo de generación de los códigos de Gray.
4. Coordinación de trabajo en equipo mediante el uso de herramientas de control de versiones.
5. Practicar planificación de tareas para trabajo de grupo.

4. Especificación del diseño

Se solicita el desarrollo de un circuito decodificador de Gray. El mismo deberá construirse según las pautas fundamentales de diseño digital sincrónico. El circuito constará de tres subsistemas:

1. Subsistema de lectura y decodificación de código Gray.
2. Subsistema de despliegue de código ingresado traducido a formato binario en luces LED.
3. Subsistema de despliegue de código ingresado y decodificado en display de 7 segmentos.

4.1. Subsistema de lectura y decodificación de código Gray

El subsistema de lectura tomará de cuatro conmutadores el código ya generado según la tabla 1.

La entrada del código deberá ser capturada por el sistema principal.

El subsistema luego deberá traducir la entrada capturada a un código binario antes de enviarse al siguiente subsistema.

Tabla 1: Código Gray de 4 bits a implementar.

Código decimal	Código binario de 4 bits	Código Gray de 4 bits
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

4.2. Subsistema de despliegue de código ingresado traducido a formato binario en luces LED

Este subsistema toma los datos ya pasados a código binario y los despliega en cuatro luces LED.

4.3. Subsistema de despliegue de código decodificado en display de 7 segmentos.

Este subsistema toma los datos en código binario y los despliega los dispositivos 7 segmentos que deberán alambrarse en una protoboard, de forma decimal.

5. Guía de diseño

Los estudiantes deberán generar un diagrama de bloques para cada subsistema, con su funcionalidad descrita y su esquema de interconexión. Deberá presentarse adecuadamente la ruta de datos desde la salida hasta la entrada, con una descripción comportamental de cada sub-bloque dentro de los subsistemas (i.e., muxes, decos, registros, etc.). No será necesario llevar la descripción de los sub-sistemas a álgebra booleana, pero para cada salida, será necesario resolver la ecuación booleana mínima que maneja los leds o los 7 segmentos. Al menos un ejemplo de simplificación (ya sea por álgebra booleana, mapas de Karnaugh u otro método usado) deberá presentarse en el informe. No se permitirá el uso de constructos avanzados de SystemVerilog en esta tarea. Todo la lógica decodificación, multiplexado y manejo de señales deberá escribirse con ecuaciones de Boole codificadas en SystemVerilog.

Sería necesario alambrar algunas partes en una protoboard para las entradas de código binario y las salidas al 7-segmentos. Un ejemplo de alambrado se ofrece en la figura reffig:proto.

Note que las salidas definidas para el manejo de los segmentos de los dispositivos de 7-segmentos se conectan a ambos. El encendido de estos dispositivos se hace con los transistores PNP conectados a los ánodos de cada 7-segmento por separado. Debido al alambrado actual, ambos disposotivos representarán el mismo número. Por tanto, usted deberá conectar un conmutador o switch similar a los de los códigos grey de entrada al decodificador para seleccionar si los 7-segmentos desplegarán ya sea las unidades o las decenas del código traducido (es decir, decidir cuál de los transistores se enciende).

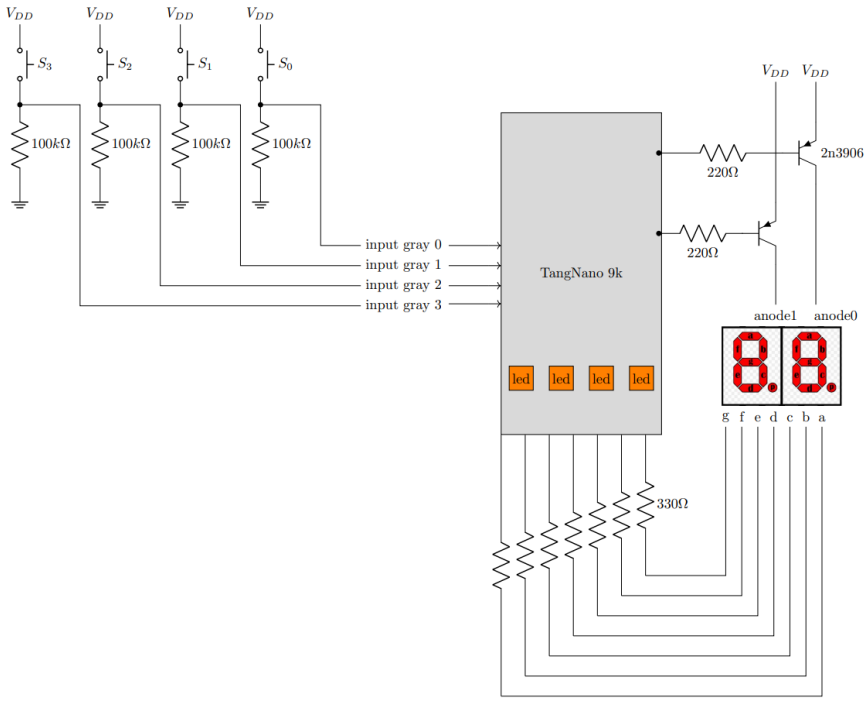

Figura 1: Ejemplo de alambrado en una protoboard con dispositivos recomendados. Aplique las definiciones correctas de tensión en cada pin. dado en el listado5

Por otro lado, es recomendable que el V_{DD} usado para alimentar los conmutadores de entrada como los transistores sean 3.3 V. Por favor tenga cuidado al definir entonces los voltajes que manejarán estos pines. Use cómo guía el la configuración de la TangNano dado en el listado5.

Listing 1: Definición de conexiones y tensiones de pin dentro de la TangNano.

```
O_LOC  "catodo_po[6]"  29;
IO_PORT "catodo_po[6]"  IO_TYPE=LVCMOS33; //Cathode g

IO_LOC  "catodo_po[5]" 30;
IO_PORT "catodo_po[5]" IO_TYPE=LVCMOS33; //Cathode f

IO_LOC  "catodo_po[0]" 51;
IO_PORT "catodo_po[0]" IO_TYPE=LVCMOS33; //Cathode e

IO_LOC  "catodo_po[1]" 42;
IO_PORT "catodo_po[1]" IO_TYPE=LVCMOS33; //Cathode d

IO_LOC  "catodo_po[4]" 53;
IO_PORT "catodo_po[4]" IO_TYPE=LVCMOS33; //Cathode c

IO_LOC  "catodo_po[3]" 54;
IO_PORT "catodo_po[3]" IO_TYPE=LVCMOS33; //Cathode b

IO_LOC  "catodo_po[2]" 55;
IO_PORT "catodo_po[2]" IO_TYPE=LVCMOS33; //Cathode a

IO_LOC  "anodo_po[1]" 57;
IO_PORT "anodo_po[1]" IO_TYPE=LVCMOS33; //anode digit 1

IO_LOC  "anodo_po[0]" 56;
IO_PORT "anodo_po[0]" IO_TYPE=LVCMOS33; //anode digit 0

IO_LOC  "clk_pi" 52;
IO_PORT "clk_pi" IO_TYPE=LVCMOS33 PULL_MODE=UP; // internal clock 27 MHz

IO_LOC  "rst_pi" 3;
IO_PORT "rst_pi" IO_TYPE=LVCMOS18; // BTN reset

IO_LOC  "codigo_bin_led_po[0]" 10;
IO_PORT "codigo_bin_led_po[0]" DRIVE=8 IO_TYPE=LVCMOS18 PULL_MODE=DOWN; //Led [0]

IO_LOC  "codigo_bin_led_po[1]" 11;
IO_PORT "codigo_bin_led_po[1]" DRIVE=8 IO_TYPE=LVCMOS18 PULL_MODE=DOWN; //Led [1]

IO_LOC  "codigo_bin_led_po[2]" 13;
IO_PORT "codigo_bin_led_po[2]" DRIVE=8 IO_TYPE=LVCMOS18 PULL_MODE=DOWN; //Led [2]

IO_LOC  "codigo_bin_led_po[3]" 14;
IO_PORT "codigo_bin_led_po[3]" DRIVE=8 IO_TYPE=LVCMOS18 PULL_MODE=DOWN; //Led [3]

IO_LOC  "codigo_gray_pi[0]" 28;
IO_PORT "codigo_gray_pi[0]" IO_TYPE=LVCMOS33 PULL_MODE=DOWN;  //Input gray [0]

IO_LOC  "codigo_gray_pi[1]" 27;
IO_PORT "codigo_gray_pi[1]" IO_TYPE=LVCMOS33 PULL_MODE=DOWN; //Input gray [1]
```

```
IO_LOC  "codigo_gray_pi[2]" 26;
IO_PORT "codigo_gray_pi[2]" IO_TYPE=LVCMOS33 PULL_MODE=DOWN; //Input gray [2]

IO_LOC  "codigo_gray_pi[3]" 25;
IO_PORT "codigo_gray_pi[3]" IO_TYPE=LVCMOS33 PULL_MODE=DOWN; //Input gray [3]
```

5.1. Sobre la codificación del HDL

Se utilizarán SystemVerilog y el suite de herramientas abierto usado para el primer tutorial del curso para desarrollar el sistema completo. Se seguirá el formato visto en clase para el estilo de codificación, cuyo apego por parte de los estudiantes formará parte de la evaluación final.

5.2. Sobre la verificación

Para cada subsistema, será obligatorio presentar simulaciones tanto a nivel de RTL (pre-síntesis) como con información de temporizado (post-síntesis y post-colocación-y-ruteo, las que se deberán presentar al asistente durante el horario de consulta del curso. Para ello, el grupo deberá generar las pruebas de banco o *testbenches* necesarios

5.3. Sobre el informe

Los estudiantes deberán presentar un informe en formato *Markdown* donde se las siguientes características del diseño final (el archivo se cargará como archivo `README.md` del repositorio):

1. Una descripción general del funcionamiento del circuito completo y de cada subsistema.
2. Diagramas de bloques de cada subsistema y su funcionamiento fundamental, según descritos en la sección 5.
3. Un ejemplo de la simplificación de las ecuaciones booleanas usadas para los leds o de los 7-segmentos.
4. Ejemplo y análisis de una simulación funcional del sistema completo, desde el estímulo de entrada hasta el manejo de los 7 segmentos.
5. Análisis de consumo de recursos en la FPGA (LUTs, FFs, etc.) y del consumo de potencia que reporta las herramientas.
6. Análisis de principales problemas hallados durante el trabajo y de las soluciones aplicadas.

La entrega final del reporte (último día para actualizar el repositorio) será en la fecha establecida en el cronograma del curso, a no ser que el profesor abra un plazo extra que deberá ser publicado por el TEC Digital, por el canal del curso.

6. Evaluación (100 %)

La evaluación de la tarea se llevará a cabo según la siguiente rúbrica:

Funcionamiento del circuito	60
Informe	40
Total	100

Tabla 2: Distribución de la calificación.

El circuito funciona totalmente y las simulaciones son funcionales	100
El circuito funciona parcialmente, pero las simulaciones son totalmente funcionales	80
El circuito no funciona, pero las simulaciones son total o parcialmente funcionales	60
El circuito no funciona, y las simulaciones tampoco son funcionales del todo	25

Tabla 3: Ponderación de la calificación.

Para la calificación proporcional por cada sección, se aplicarán los siguiente criterios de ponderación: Para más información, revise los tutoriales y videos en [2, 1].

Referencias

[1] David Medina. *Video tutorial para principiantes. Flujo abierto para TangNano 9k.* Jul. de 2024. URL: https://www.youtube.com/watch?v=AKO-SaOM7BA.

[2] David Medina. *Wiki tutorial sobre el uso de la TangNano 9k y el flujo abierto de herramientas.* Mayo de 2024. URL: https://github.com/DJosueMM/open_source_fpga_environment/wiki.