Tarea II: Diseño digital sincrónico en HDL

Sr. David Medina
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 sincrónico utilizando lenguajes de descripción de hardware.

3. Objetivos específicos

1. Elaborar una implementación de un diseño digital sincrónico en una FPGA.
2. Construir un *testbench* básico para validar las especificaciones del diseño.
3. Comprender los conceptos de sincronización de datos asincrónicos.
4. Implementar un algoritmo de captura de datos de un teclado hexadecimal.
5. Implementar una sencilla función de suma aritmética en un HDL.
6. Implementar un algoritmo de despliegue de datos en tres dispositivos de 7 segmentos.
7. Coordinación de trabajo en equipo mediante el uso de herramientas de control de versiones.
8. Practicar planificación de tareas para trabajo de grupo.

4. Especificación del diseño

Se solicita el desarrollo de un circuito que capture dos números enteros positivos de al menos tres dígitos decimales cada uno, a partir de un teclado hexadecimal y que despliegue la suma sin signo de los mismos en cuatro dispositivos de 7 segmentos. El mismo deberá construirse según las pautas fundamentales de diseño digital sincrónico. El circuito constará de tres subsistemas:

1. Subsistema lectura de teclado hexadecimal. Este deberá encargarse de la captura, eliminación de rebote y sincronización de dos números de hasta tres dígitos decimales de entrada a partir de un teclado mecánico de formato hexadecimal.

2. Subsistema de suma aritmética de los dos datos.

3. Subsistema de despliegue de los números ingresados y del resultado de la suma en cuatro dispositivos de 7 segmentos.

El circuito deberá funcionar a una frecuencia de reloj de 27 MHz.

4.1. Subsistema de lectura del teclado mecánico hexadecimal

El subsistema de lectura tomará de un teclado hexadecimal como el de la figura 1 los datos de entrada de dos números decimales positivos de tres dígitos. Para ello, se deberá, primero, sincronizar los datos de manera adecuada al capturarlos en la FPGA, eliminando cualquier posible rebote o ruido mecánico.

Los estudiantes deberán idear un algoritmo de entrada de datos que permita irlos metiendo en un formato similar al de una calculadora tradicional, de manera que sea posible verlos desplegarse en los 7-segmentos a medida que son ingresados. Ello significa que, de alguna manera, los estudiantes deberán generar un control mediante una FSM que permita decidir el momento en que se han ingresado los dígitos del primer dato, los del segundo dato, y cuando se debe ejecutar la suma para desplegar el resultado.

Existen ya soluciones en varios repositorios libres para el problema de lectura de un teclado hexadecimal (algunas no muy eficientes y que no respetan algunos de los criterios vistos en clase para escribir buen código SV, como esta acá: `https://github.com/SamarthWalse10/Hexadecimal-Keypad-Encoder-Verilog?tab=readme-ov-file`, pero que les pueden servir de inicio). La solución tradicional es aprovechar el arreglo fila-columna de estos teclados para generar un código binario equivalente a partir de la conexión en un determinado par fila-columna en función del botón apretado. Una opción es ir barriendo las columnas con un contador de anillo (que ya veremos en clase) y estar monitoreando las filas. Si de repente las filas son distintas de cero, significa que se apretó alguna tecla. Se captura el código de la fila y se complemente con el de la columna que se está barriendo en ese momento y ya se tiene un par fila-columna para identificar el valor.

Ahora, lo primero es averiguar qué tipo de configuración tiene el teclado que se va a usar (si ya tiene resistencias de pull-up o pull-down incorporadas, etc.). Primero definan la conexión eléctrica del teclado antes de definir cómo lo van a leer (siempre recordando los niveles de tensión adecuados para los pines de la TangNano). Los estudiantes están autorizados a usar estos repositorios como base para su trabajo, siempre y cuando los citen de manera adecuada, por respeto a dichos autores. Más referencias en [1].

El formato de codificación de los datos que se guardarán en esta etapa queda a criterio de los estudiantes, pero se recomienda una codificación binaria simple para facilitar el trabajo de la siguiente unidad.

4.2. Subsistema de suma aritmética de los dos datos

Este subsistema toma los datos ya almacenados en la FPGA y ejecuta la suma aritmética de los mismos, según el formato definido por el grupo. El sistema entregará la respuesta al siguiente subsistema, siempre de manera sincrónica. Se permitirá el uso de todas las facilidades del SystemVerilog para realizar esta suma.

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

Este subsistema toma los datos en código binario y los despliega de forma decimal en los dispositivos de 7 segmentos que deberán alambrarse en una protoboard. Se deberá diseñar el sistema de refresco necesario para manejar con cátodos comunes, los leds de 7-segmentos, alimentados por los 4 ánodos.

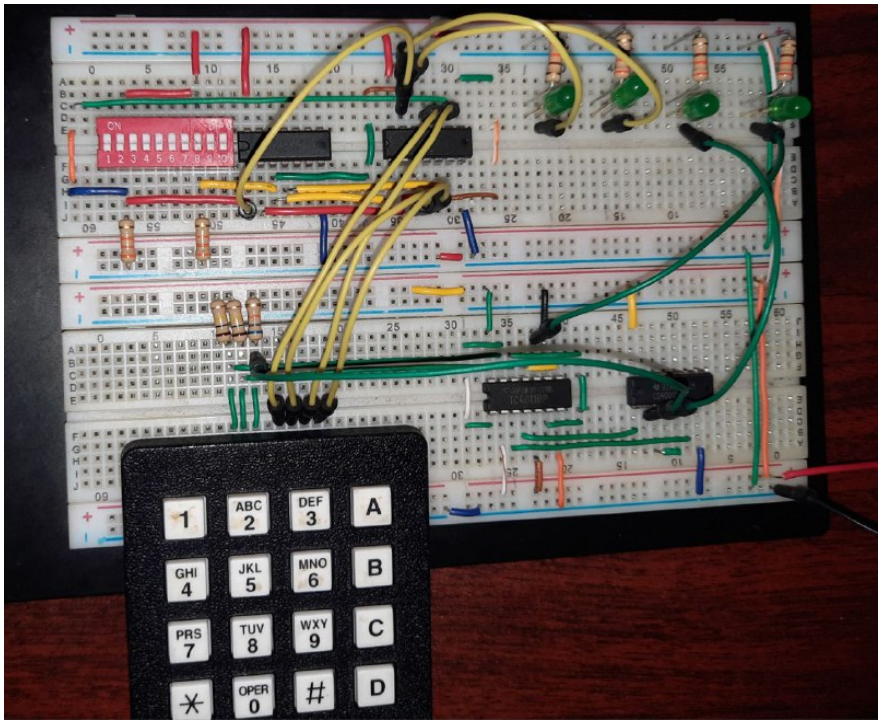

Figura 1: Ejemplo de un teclado hexadecimal conectado a una protoboard. El grupo pueda asignar las teclas no numéricas para las funciones que desee. Verifique que las resistencias de pull-up o pull-down no están ya incorporadas en el teclado.

5. Guía de diseño

Los estudiantes deberán generar un diagrama de bloques para cada subsistema, con su funcionalidad descrita y su esquema de interconexión, los cuales deberán presentarse en el informe. Deberá presentarse un plano completo del sistema total, con la ruta de datos identificada desde la salida hasta la entrada, con un nombre descriptivo de cada sub-bloque dentro de los subsistemas por los que pasa dicha ruta de datos (i.e., muxes, decos, registros, funciones específicas diseñadas por los estudiantes, etc.).

Se permite el uso de constructos avanzados de SystemVerilog en esta tarea. Todo la lógica de decodificación, multiplexado y manejo de señales deberá escribirse según los esquemas de codificación de SystemVerilog recomendados en clase.

Recuerde todas las precauciones usadas en la tarea anterior para el manejo de los 7-segmentos y las tensiones necesarias en la placa donde se encuentra la TangNano.

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 (ver [3]). 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. Revisen especialmente la codificación de las FSM según el formato expuesto en clase.

5.2. Sobre la sincronización

Las señales externas deben sincronizarse con el reloj interno de la TangNano. Para ello, es necesario además de registrar dichas señales, eliminar potenciales rebotes.

Cada subsistema deberá estar adecuadamente registrado a la entrada y salida. El flujo de datos debe ser indicado de manera explícita. No es necesario mostrar en detalle los bloques en que se implementen las máquinas de estados finitos (FSM) que se diseñen, pero sí sus diagramas de estado y las señales de control que manejan cada bloque en la ruta de datos.

Se utilizará como señal del reloj, el pin ya provisto de 27 MHz en la TangNano. Solo podrá usarse un reloj en todo el sistema. Para los sistemas más lentos, será necesario generar las bases de tiempo por medio de divisores de frecuencia (relojes) que garanticen el adecuado refrescamiento tanto de la lectura del teclado como del despliegue de los datos.

5.3. 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.

5.4. 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. Un párrafo resumen de introducción.
2. Una definición general del problema, de los objetivos buscados y de las especificaciones planteadas en el enunciado.
3. Una descripción general del funcionamiento del circuito completo y de cada subsistema.
4. Diagramas de bloques de cada subsistema y su funcionamiento fundamental, según descritos en la sección 5.
5. Diagramas de estado de todas las FSM diseñadas (si existen), según descritos en la sección donde se describe cada subsistema.
6. Ejemplo y análisis de una simulación funcional del sistema completo, desde el estímulo de entrada hasta el manejo de los 7 segmentos.
7. Análisis de consumo de recursos en la FPGA (LUTs, FFs, etc.) y del consumo de potencia que reporta las herramientas.
8. Reporte de velocidades máximas de reloj posibles en el diseño (mínima frecuencia de reloj para este diseño: 27 MHz).
9. 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 1: Distribución de la calificación.

Para la calificación proporcional por cada sección, se aplicarán los siguiente criterios de ponderación:

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 2: Ponderación de la calificación.

Para más información, revise los tutoriales y videos en [3, 2].

Referencias

[1] Andrew House. *Hex Keypad Explanation.* Nov. de 2009. URL: `https://www-ug.eecg.toronto.edu/msl/nios_devices/datasheets/hex_expl.pdf`.

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

[3] 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`.