PRÁCTICA 4 – SISTEMAS EMPOTRADOS

Síntesis de circuitos en Verilog

Lucas Serrano Jiménez

César San Blas Leal

tarea 1

**1. Busca información sobre los principales fabricantes de FPGAs (Xilinx/AMD, Altera/Intel, Microchip y Lattice) y compara las arquitecturas de cada uno de ellos destacando las diferencias en sus arquitecturas (celdas básicas, FPGAs SRAM vs FPGAs flash), ventajas e inconvenientes de cada una.**

* **Xilinx**

Una de las primeras compañías de FPGAs. Su arquitectura está basada en celdas de tablas de búsqueda (LUTs), configurables por el usuario. Mayormente utiliza FPGAs SRAM, lo que significa que la configuración es volátil y necesita ser recargada cada vez que se enciende. Sin embargo, esto permite una mayor flexibilidad y rápida reconfiguración. Estos factores hacen que se reconozca esta marca por su alto rendimiento y flexibilidad, aunque el uso de SRAM conlleva un consumo de energía relativamente alto.

* **Altera**

Altera emplea una arquitectura basada en bloques lógicos adaptativos (ALMs), que permiten una mayor optimización y eficiencia en el diseño. Este fabricante ofrece modelos tanto con SRAM como con Flash. Estos últimos destacan en la capacidad de almacenar la configuración sin necesidad de reprogramación tras el encendido, pero sacrifican reconfigurabilidad y velocidad de actualización comparadas con las SRAM. Altera también incorpora una interfaz de usuario que hace la programación más visual y simple.

* **Microchip**

Utiliza arquitecturas FPGA que incluyen bloques lógicos, LUTs y RAMs integradas para aplicaciones específicas. Ofrece FPGAs basadas en SRAM y FPGAs antifusibles, las cuales ofrecen una mayor seguridad pero anulan la reprogramabilidad. Es por ello que esta compañía tiene bastante presencia en comunicaciones, defensa y aviación.

* **Lattice**

Lattice emplea una arquitectura FPGA basadas en LUTs y bloques lógicos y mayoritariamente usa SRAM. Tiene un enfoque en FPGAs de menor tamaño y menor consumo de energía, ideales para aplicaciones de bajo consumo y portabilidad, lo que también conlleva una menor capacidad de procesamiento en comparación con los competidores de gama alta.

**2. Las FPGAs son dispositivos flexibles en los que se puede implementar lógica digital compleja como microcontroladores, puertos Ethernet, etc. Explica qué es una “soft macro” o “soft IP core” y qué es un “hard macro” o “hard IP core”.**

* **Soft Macro**

Este término se refiere a un componente configurable diseñado en lenguaje de descripción de hardware (HDL), destinado a ser implementado en la lógica programable de una FPGA o ASIC. Son altamente flexibles y reconfigurables, permitiendo la personalización y optimización según las necesidades del proyecto.

* **Hard Macro**

Un Hard Macro es un componente prediseñado y pre-sintetizado para su implementación. Estos bloques IP están optimizados y diseñados para un hardware específico, proporcionando un rendimiento superior y eficiencia en términos de área y consumo de energía. A diferencia de las Soft Macros, estos componentes están fijados en su diseño y estructura física, lo que implica que no son reconfigurables.

**3. Actualmente hay una interesante comunidad de desarrollo entorno a RISC-V. Busca información sobre este proyecto y explica cómo se están implementando en FPGAs. ¿Se puede implementar con soft y con hard IP cores?**

Se trata de un proyecto interesante dado que RISC-V es una arquitectura de código abierto para microprocesadores y la flexibilidad de las FPGAs permite implementarla adecuadamente. Esta implementación se puede realizar mediante soft y hard IP cores.

La opción Soft IP Core es la más flexible porque consiste en adecuar el procesador para las necesidades pertinentes. Por otro lado, la opción Hard IP Core consiste en utilizar material prefabricado y resulta más rápida, aunque menos flexible, que la alternativa anterior.

**4. Los SoCs incorporan módulos diversos como, por ejemplo, microcontroladores (ARM, RISC-V, PowerPC). Busca información sobre las arquitecturas más modernas tipo SoC (System on Chip) identificando su arquitectura, tipo de módulos y periféricos que incorporan.**

Tres de las arquitecturas más populares SoC son ARM, RISC-V y Power PC.

ARM se caracteriza por su bajo consumo de energía y su alto rendimiento. Estas arquitecturas se utilizan en una amplia gama de dispositivos, desde smartphones y tablets hasta televisores y electrodomésticos.

Por otro lado, la arquitectura RISC-V es una arquitectura de código abierto que se caracteriza por su flexibilidad y su potencial para reducir los costes. Las arquitecturas RISC-V se utilizan también en smartphones y tablets, además de en otros ámbitos como automóviles y servidores.

Finalmente, la arquitectura PowerPC es una arquitectura de 32 bits que se utiliza en una amplia gama de dispositivos, desde ordenadores personales hasta servidores. Las arquitecturas PowerPC se caracterizan por su alto rendimiento y su compatibilidad con el software existente.

tarea 2

**2.1. Carga el ejemplo *leds* e interpreta el fichero de formas de onda VCD generado por la simulación anterior.**

Este ejemplo, tal y como se explicó durante la clase, simplemente carga en los pines de 4 leds un estado de nivel alto. Al ejecutar la simulación, muestra un gráfico con un estado *on* durante todo el tiempo que dura el programa.

A screenshot of a computer

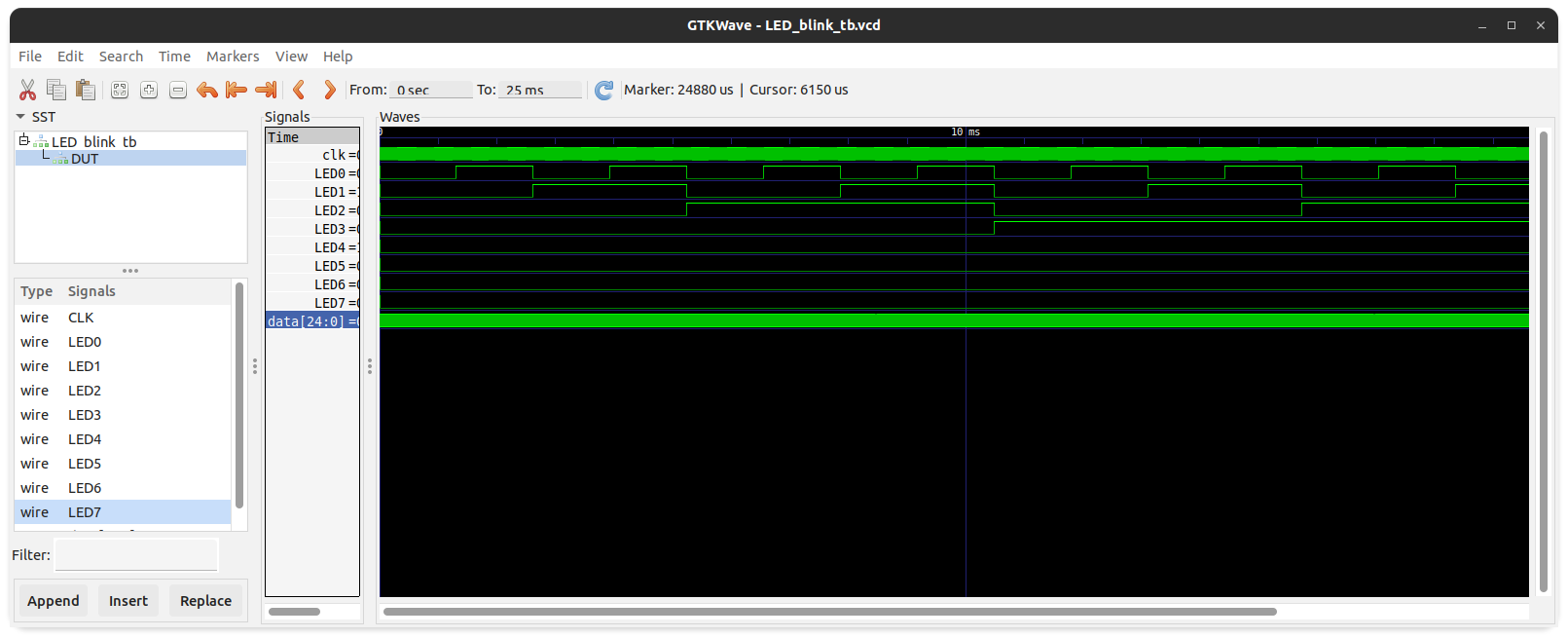
Description automatically generated

**2.2 Ahora vamos a sintetizar este código y a cargarlo en nuestra placa. Explica el funcionamiento de LED\_Blink.v, detallando qué se hace en cada línea de código. ¿Por qué exactamente vemos parpadear el LED con una frecuencia de más o menos 1 segundo?**

Tras sintetizar los códigos de LED\_blink y cargarlos en la FPGA se observa el LED aparentemente encendido constantemente, como se muestra en [este vídeo demostrativo](https://youtu.be/1sVzIcZZddE). Este comportamiento se debe a que el LED tendrá un periodo de encendido y apagado de 212 ciclos en un reloj de 12 MHz por lo que resulta en un tiempo de aproximadamente 5 ns, inapreciable al ojo humano.

**2.3. Sabiendo cómo funciona el código del ejercicio anterior, modifica el código para realizar un contador usando todos los LEDs de su tarjeta FPGA. Se debe poder apreciar el apagado y encendido de los LEDs.**

En este caso, para poder apreciar el apagado y encendido de los LEDS, se aumenta el número de bits del registro, asignando los últimos a los 8 LEDS que posee la FPGA. En este caso se ha aumentado a 24 el número de bits del contador, por lo que ahora el periodo del cambio de estados es de 224 ciclos. Asignando cada uno de los últimos 8 bits del registro a los LEDS de la placa se obtiene una escalera de frecuencias, esta vez apreciable al tratarse de un periodo considerablemente mayor. El comportamiento de este LED se muestra en [este vídeo demostrativo.](https://youtube.com/shorts/GFjswYOTGzw?feature=share)



tarea 3

**3.1 Crea un proyecto de simulación que incluya los ficheros creados y simula tu módulo. Captura las formas de onda y explica qué sucede en la simulación.**

Con la escala adecuada, se puede observar que la señal varía y cambia de estado. Sin embargo, como se explicó en la tarea anterior, a pesar de que el LED sí parpadea, lo hace a una frecuencia tan rápida que es inapreciable.

A screenshot of a computer

Description automatically generated

**3.2 Realiza un testbench en Verilog para simular el diseño del apartado 2.3 y aporta las formas de onda.**

En este caso al asignar cada LED a un bit del registro, se puede apreciar la escalera de frecuencias que generan las señales. Debido a la naturaleza binaria del registro usado para controlar el estado de los LEDs, cada uno tendrá el doble de periodo que el anterior.

A screenshot of a computer

Description automatically generated

tarea 4

**4.1 Utiliza el código del ejercicio 2.3 y modifica lo necesario para que al pulsar el botón 2 (SW2) de la placa, el contador se reinicie.**

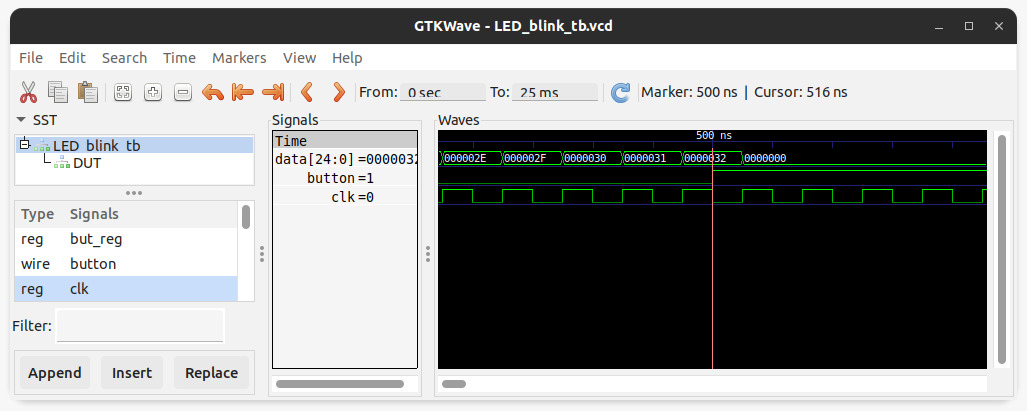
En el apartado 2.3 el contador aumentaba en uno con cada ciclo de reloj. Esta nueva versión comprueba antes el estado del botón 2 y, si se ha pulsado, reinicia el contador a cero mediante *data<=0*. Si no se ha pulsado, continúa aumentando la cuenta normalmente.

Para esta modificación se ha debido indicar el botón como entrada del módulo e indicar en el archivo .pcf su conexión al botón 2.

**4.2 Crea el testbench y sintetiza tu diseño cuando la simulación sea correcta (aporta formas de onda y explicación). El test tiene que simular las entradas y comprobar automáticamente que las salidas se comportan como se espera. Como resultado de la comprobación, el test tiene que imprimir en pantalla un mensaje indicando si el test ha fallado o no.**

Partiendo del testbench LED\_blink\_tb.v hubo que añadir el botón como la nueva entrada del módulo y un contador que simule que se pulsa tras un tiempo determinado. Además, inmediatamente antes de la sentencia que finaliza la simulación se ha incluido un pequeño mensaje que muestra por terminal que se ha concluido sin fallos.

Como se puede comprobar en las imágenes, la simulación muestra el efecto deseado de reinicio del contador.

****

**Texto

Descripción generada automáticamente**

**4.3 Prueba que el diseño funciona en la placa (aporta vídeo).**

[Vídeo demostrativo.](https://youtube.com/shorts/pe8jFKOxjaw?feature=share)

**4.4 El efecto de los rebotes en los interruptores es posible que esté afectando negativamente al sistema. Diseña un módulo basado en un contador que sólo considere el botón cuando se ha pulsado durante al menos 10 ms. Diseña un test para comprobar su funcionamiento y pruébalo en la placa (aporta vídeo).**

El nuevo módulo se llama *debounce.* Como entradas únicamente presenta el reloj y el botón del que se quieren eliminar rebotes y, como salida, la señal del botón pulsado. Utilizando el reloj integrado en la placa se ha calculado y utilizado el número de ciclos que corresponden a 10 ms:120000 ciclos.

Con cada flanco de subida del reloj se actualiza un contador hasta que hayan pasado 10 ms. En ese caso se actualiza y se devuelve el estado pulsado del botón con los rebotes eliminados. Mientras se llega a 10 ms, se comprueba que el estado que se lee del botón se mantiene igual al leído inmediatamente antes. Si es así, se aumenta en uno el contador. Si no, se reinicia la cuenta.

[Vídeo demostrativo.](https://youtu.be/KW78cGsr4C4)

**4.5 Modifica el código para que al pulsar el botón 1 (SW1) el comportamiento del contador cambie. Inicialmente el contador sumará y al pulsar el botón 1, el contador restará, al volver a pulsarlo el comportamiento cambiará de nuevo.**

La resolución de este apartado consiste en añadir el botón 1 a la configuración de los módulos y, cuando se pulse, reducir el contador en uno en lugar de continuar aumentándolo. Además se han eliminado los rebotes de las pulsaciones de los dos botones, instanciando el módulo creado anteriormente.

La sección de código que responde a este apartado es la siguiente:

wire w\_btn2;//reset

wire w\_btn1;//restar

debounce d2(CLK, BUTTON2, w\_btn2);

debounce d1(CLK,BUTTON1,w\_btn1);

always@(posedge CLK)

begin

if(w\_btn2==1)

data<=0;

else if(w\_btn1==1)

data<=data-1;

else

data<=data+1;

end

**4.6 Testea tu nuevo diseño (aporta formas de onda y explicación). El test tiene que simular las entradas y comprobar automáticamente que las salidas se comportan como se espera. Como resultado de la comprobación, el test tiene que imprimir en pantalla un mensaje indicando si el test ha fallado o no.**

**4.7. Prueba que el diseño funciona correctamente también en la placa (aporta vídeo).**

[Vídeo demostrativo.](https://youtube.com/shorts/CFGx_Fg1Czs?feature=share)

Como se puede apreciar, al restar lo suficiente se vuelve al led anterior.

tarea 5

En este apartado se explicará el funcionamiento de la máquina de estados implementada. En primer lugar, como se puede observar en la siguiente figura, se muestra el código con los requerimientos solicitados en el guion de la práctica en cuanto a la máquina de estados. El resto de código se puede consultar en el archivo *fsm.v*.

A computer screen shot of a program code

Description automatically generated

A continuación, se realizará una simulación para comprobar su funcionamiento. La siguiente imagen representa el inicio de la simulación. Antes de pasar a analizar las ondas cabe destacar que las señales A y B simulan las señales producidas al pulsar dos botones. En el inicio del programa, al encontrarse en el estado de espera, todos los leds se encuentran encendidos. Aproximadamente, a los 50 ns de haber comenzado el programa se efectúa un cambio de estado pulsando los dos botones simultáneamente. Esto lleva al programa al segundo estado, el de programación. Se puede apreciar como en este estado solo se encuentran encendidos los cuatro LEDs laterales. A los 100 ns de haber comenzado se pulsa el botón A, incrementando el contador que controla el comportamiento de los LEDs en el modo de ejecución. Acto seguido, se pasa al modo de ejecución pulsando el botón 2, nótese como cambia el valor de la variable CurrentState. En este momento, tal como indicaba el guion de la práctica, se espera aproximadamente un segundo. Esta operación se ha llevado a cabo con la creación de un módulo diseñado específicamente para este fin. Este se llama oneSec y, puesto que se trabaja con un reloj de 12 MHz, toma los siguientes ciclos en activarse:

Número de ciclos = 12000000 ciclos/segundo · 1 segundo = 12000000 ciclos

Por simplicidad en el diseño y para reducir la carga computacional, en la simulación se han establecido a 120000 ciclos. El registro elegido para tal fin, puede llegar hasta 131072. Se han elegido 17 bits para este puesto que:

217 > 120000 > 216

A screenshot of a computer

Description automatically generated

Cuando el contador llega al número establecido, en este caso 120000 ciclos, se activa la flag ONESEC, permitiendo al programa comenzar con la ejecución. Puesto que en la fase de programación solo se ha pulsado el botón A una vez, dará comienzo la ejecución del Comando 1. Este, como se indicaba en la práctica, realizaba un parpadeo del LED 0, similar al del ejercicio 2.1.

A screenshot of a computer

Description automatically generated

Continuando con el análisis de la simulación, para mostrar otro comando de ejecución diferente y testear el cambio a otro estado mientras se encuentra en funcionamiento aún el anterior, se ha simulado la pulsación de los dos botones, pasando así del estado de ejecución al de programación. Esta vez, como se puede apreciar en la figura, tras pasar al estado de programación (Estado 2) se pulsa dos veces el botón A por lo que al pulsar el botón B y entrar en modo de ejecución, se pasará al Comando 2. Este consiste en un parpadeo simultáneo de los LEDs. Al no estar especificado en el guion su frecuencia, se ha tomado un ciclo.

A screenshot of a computer

Description automatically generated

El testbench encargado de la simulación descrita previamente se muestra en la siguiente figura.

A screenshot of a computer program

Description automatically generated