Instituto Tecnológico de Costa Rica
Escuela de Ingeniería Electrónica
EL3313 Taller de Diseño Digital
Prof. M.Sc. Kaled Alfaro Badilla
II semestre 2024

Documentación del laboratorio 3: Microcontrolador

Grupo 1

Integrantes:
Brayan de Jesús Barquero Madrigal
Keylor David Muñoz Soto
Jeffrey Isaac Salas Quiel

28 Noviembre del 2024

Índice

1. Introducción

Este documento describe el diseño e implementación de un microcontrolador basado en la arquitectura RISC-V, utilizando como base el procesador PicoRV32 [1] y el toolchain [2] para la programación y validación del ensamblador que ejecutará la FPGA para recibir y mostrar imágenes. El objetivo principal fue diseñar los bloques adicionales como la ROM, RAM, UART, Y periféricos implementados en SystemVerilog, que permitan la interacción entre una laptop, una FPGA Nexys 4 DDR y la transmisión de imágenes hacia un LCD conectado a una Tang Nano 9k.

Se integraron herramientas como IP-Cores en Vivado [3] para la gestión de memoria y un script en Python [4] para la comunicación entre una laptop y la Nexys 4 DDR, logrando un sistema funcional capaz de recibir, almacenar y desplegar imágenes en tiempo real.

2. Arquitectura del Microcontrolador

El proyecto utiliza el procesador PicoRV32 del repositorio [1], el cual implementa el conjunto de instrucciones RV32I. Este procesador sirvió como núcleo central del microcontrolador, al cual se le añadieron los siguientes bloques:

- **Memorias ROM y RAM:** Implementadas mediante IP-Cores de VIVADO [3] para almacenar datos y programas.
- **Manejador de buses:** Responsable de la comunicación entre los bloques y periféricos.
- **Controladores periféricos:** Incluyen control para LEDs, interruptores y botones.
- **Interfaz UART:** Permite la comunicación entre la laptop, la Nexys 4 DDR y la Tang Nano 9k.

2.1. Diagrama de Bloques del Sistema

El siguiente diagrama sustraído del enunciado del laboratorio 3 ilustra la arquitectura general del sistema:

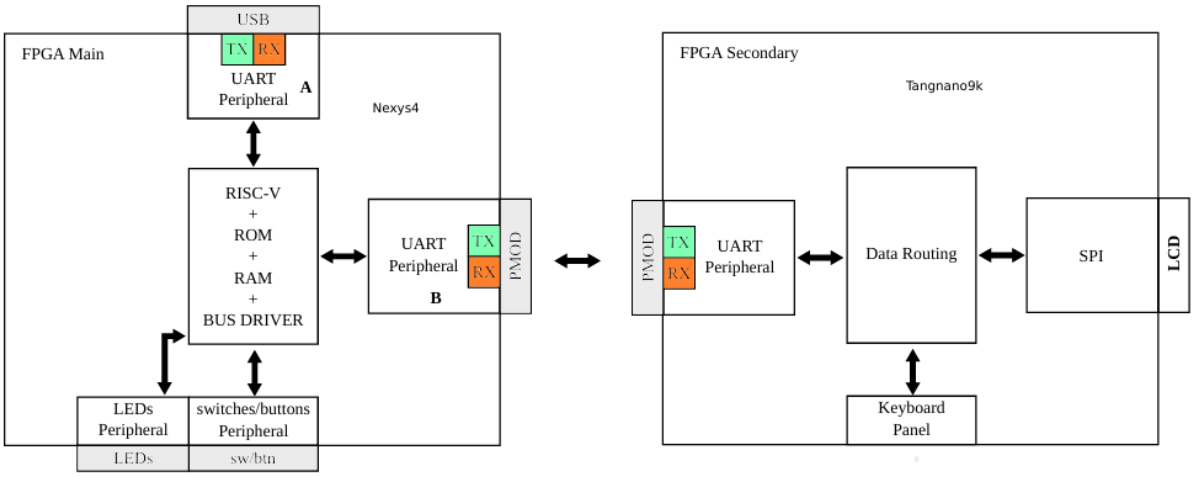

Figura 1: Diagrama para el programa en ensamblador RISC-V.

3. Generación de Reloj

Para la generación del reloj de 10 MHz requerido por el sistema, se utilizó un PLL configurado en VIVADO [3]. Este bloque garantiza la estabilidad y precisión del sistema al sincronizar las operaciones.

4. Memorias ROM y RAM

Las memorias se implementaron mediante IP-Cores en Vivado, configurados con los parámetros siguientes:

4.1. Configuración de la Memoria ROM

- **Tipo:** Memoria ROM síncrona.
- **Ancho de palabra:** 32 bits.
- **Profundidad:** 16 KB.
- **Modo de acceso:** Solo lectura.
- **Archivo de inicialización:** `rom.coe`

4.2. Configuración de la Memoria RAM

- **Tipo:** Memoria RAM síncrona.
- **Ancho de palabra:** 32 bits.
- **Profundidad:** 256 KB.
- **Modo de acceso:** Lectura y escritura.
- **Interfaz:** Compatible con bus AXI4.

4.3. Esquema de Conexión

Ambas memorias se conectan al manejador de buses, que coordina el acceso entre el procesador y los periféricos. La comunicación es síncrona, utilizando un reloj común generado por el PLL.

5. Transmisión de Imágenes

Para la creación de las imágenes en el formato `.raw`, se creo un script en **Python** [4] al que se le entrega una imagen en formato `.png` y devuelve esa misma imagen ahora en formato `.raw` que es el que se utilizará para transferir las imágenes a la FPGAs para que posteriormente sean mostrada en el LCD.

Para la transmisión de las imágenes se desarrolló otro script en **Python** [4] que permite enviar imágenes en formato `.raw` desde la laptop a la Nexys 4 DDR. Este script emplea la comunicación UART para la transferencia de datos. A continuación, se describen los pasos principales:

1. La imagen en formato `.raw` es cargada desde la laptop.
2. Los datos son divididos en paquetes y enviados a través de la UART.
3. La Nexys 4 DDR recibe los paquetes y los almacena en su memoria RAM para su posterior uso.

6. Programa en Ensamblador de RISC-V (RV32I)

El programa en ensamblador fue desarrollado en etapas, utilizando una máquina de estados que incluye los siguientes estados principales:

6.1. Estados del programa

- **REPOSO:** Estado inicial, donde el microcontrolador espera comandos de la laptop o periféricos como UART, y los botones y switches de la Nexys 4 DDR.
- **ALMACENAMIENTO:** Los datos de las imágenes son recibidos y almacenados en la RAM, con cada imagen almacenada se enciende un LED en la Nexys 4.
- **RETRANSMITIR:** Reenvía a la laptop un comando para indicar si el Nexys 4 está listo para recibir imágenes o no.
- **DESPLEGAR:** Las imágenes almacenadas en la RAM de la Nexys 4 son enviadas a la Tang Nano 9k para ser mostradas en el LCD, según la tecla presionada en el teclado hexadecimal.
- **LIBERAR:** Al accionar alguno de los switches se borra una de las imagenes de la RAM, correspondiendo al número de switch accionado.

El diagrama de este programa, hecho con la herramienta [5], se muestra a continuación:

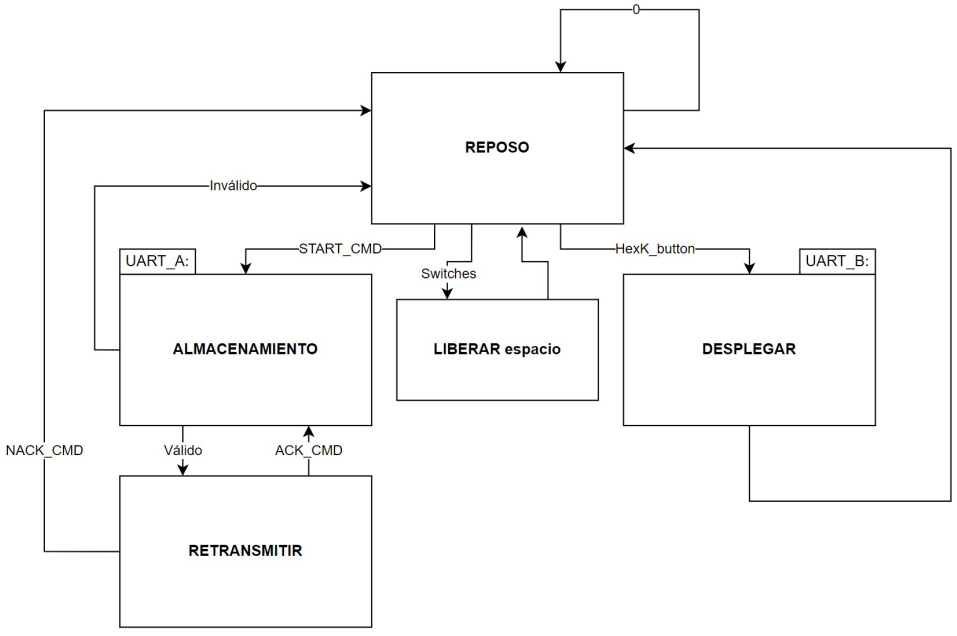

Figura 2: Diagrama para el programa en ensamblador RISC-V.

Estado	Descripción	Acción Principal
REPOSO	Espera comandos	Monitorear UART
ALMACENAMIENTO	Recibe y almacena datos	Escribir en RAM
RETRANSMITIR	Envía datos al LCD	Leer desde RAM y transmitir
DESPLEGAR	Muestra la imagen	Controlar SPI

Tabla 1: Estados del Programa en Ensamblador.

6.2. Implementación en el microcontrolador

Para que el microcontrolador ejecute el programa creado en ensamblador de RISC-V se utilizó el toolchain [2], este programa utiliza únicamente el conjunto de instrucciones RV32I que es el que soporta el microcontrolador, para compilar el programa correctamente también se debe tomar en cuenta que se está usando este conjunto de instrucciones, para lo que se utilizó la bandera de compilación `-march=rv32i`. Para crear el archivo `.hex` o directamente el `.coe` que utiliza Vivado, se desensambla el archivo compilado y se inspecciona el archivo desensamblado con `objdump`, con la ayuda de un comando de teclas en Visual Studio [6] se selecciona la columna de instrucciones y se acomodan para crear el archivo `.coe` para añadir este programa a la ROM del microcontrolador en Vivado.

7. Pruebas y Validación

7.1. Testbench General

Se creó un testbench global para la FPGA Nexys 4 DDR. Este testbench validó la funcionalidad integrada de todos los bloques y el correcto funcionamiento del programa en ensamblador.

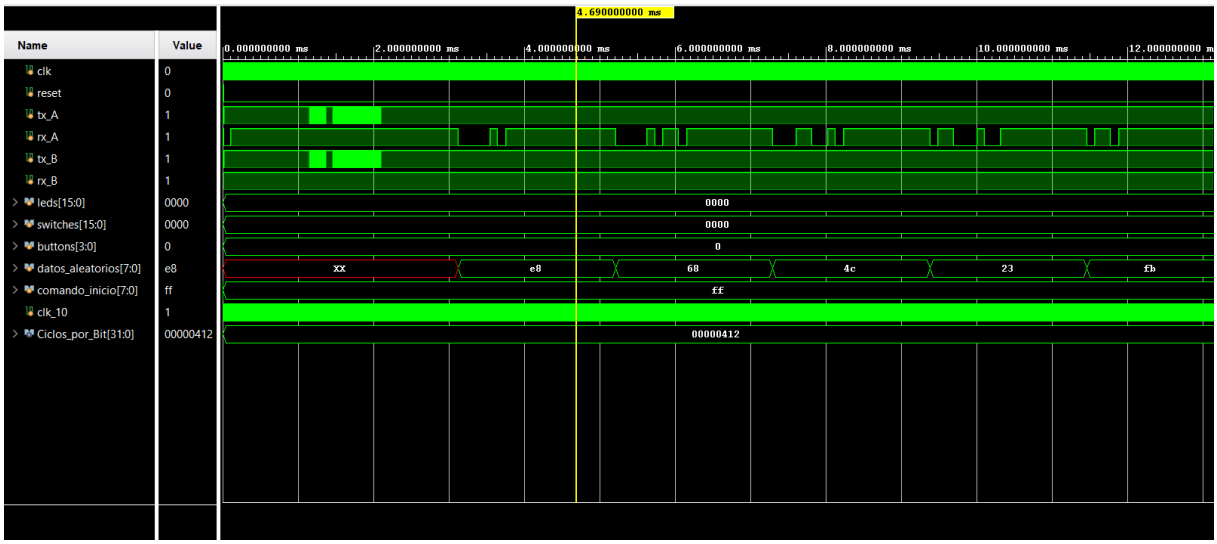

Figura 3: Testbench del Nexys 4 DDR con RISC-V core.

7.2. Testbench UART

De manera específica, se desarrolló un testbench para validar el correcto funcionamiento de la interfaz UART, asegurando una comunicación precisa entre la laptop y el Nexys

4, y también entre el Nexys 4 y el Tang Nano 9k.

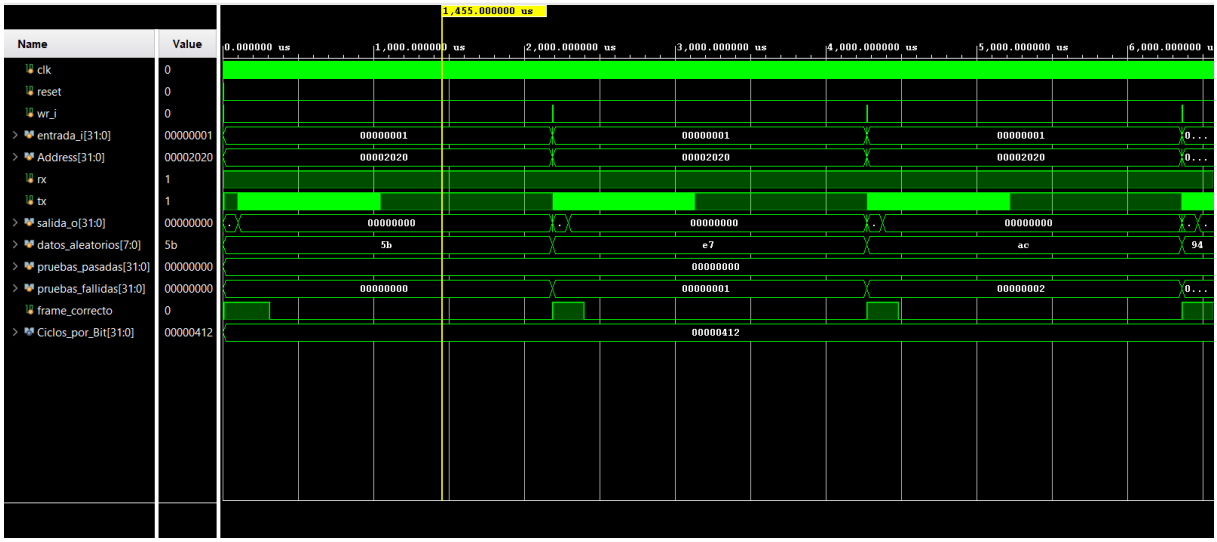

Figura 4: Testbench del bloque UART.

8. Resultados

El sistema completo logró cumplir con los requerimientos establecidos. Las imágenes fueron correctamente enviadas desde la laptop a la Nexys 4 DDR, y luego transferidas a la Tang Nano 9K para ser desplegadas en el LCD.

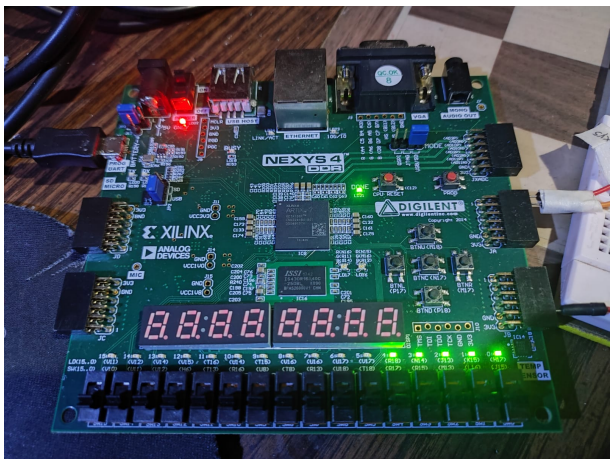

Figura 5: Carga de imágenes y LEDs Nexys.

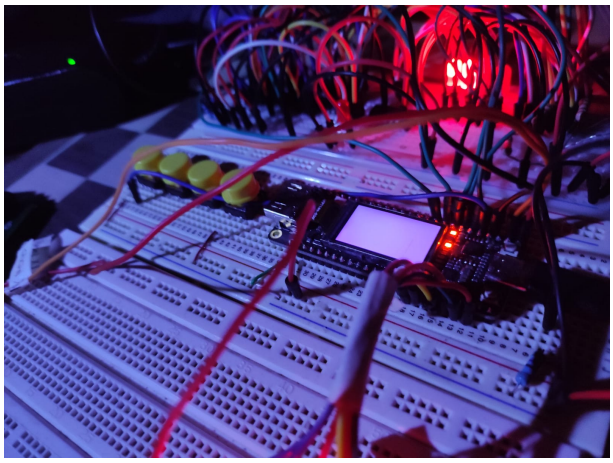

Figura 6: Carga de imágenes al LCD

Figura 7: Cambio de colores.

9. Conclusiones

El desarrollo e implementación del microcontrolador basado en la arquitectura RISC-V y el procesador PicoRV32 permitió cumplir exitosamente con los objetivos del proyecto, demostrando la viabilidad de diseñar un sistema funcional y altamente integrado en FPGA para aplicaciones de procesamiento y transmisión de imágenes. A continuación, se presentan las principales conclusiones:

1. **Integración de Bloques Personalizados:** La implementación de bloques adicionales, como memorias RAM y ROM mediante IP-Cores de Vivado, así como la integración de una interfaz UART y periféricos, permitió extender la funcionalidad del procesador PicoRV32. Esto demuestra la flexibilidad de la arquitectura RISC-V para adaptarse a requisitos específicos.

2. **Diseño y Validación de Periféricos:** Los periféricos diseñados, como LEDs, interruptores y botones, facilitaron la interacción del sistema con el usuario, mientras que la interfaz UART estableció un canal de comunicación confiable entre la laptop y las FPGAs. La validación exhaustiva de estos bloques mediante testbench específicos garantizó su correcto funcionamiento.

3. **Transmisión y Procesamiento de Imágenes:** El sistema logró transmitir imágenes en formato `.raw` desde una laptop a la FPGA Nexys 4 DDR y, posteriormente, enviarlas al LCD de la Tang Nano 9K. Este flujo de datos, basado en el almacenamiento temporal en la RAM y la gestión del procesador, valida la capacidad del sistema para manejar aplicaciones en tiempo real.

4. **Programación en Ensamblador RISC-V:** La programación en ensamblador permitió optimizar los recursos disponibles en el procesador PicoRV32. La máquina de estados implementada en el programa demostró ser efectiva para gestionar las diferentes operaciones del sistema, como la recepción, almacenamiento, retransmisión y visualización de datos.

5. **Uso de Herramientas:** El uso de herramientas modernas, como Vivado para la configuración de IP-Cores, el toolchain de RISC-V para la generación de archivos `.coe`, y scripts en Python para la generación y transmisión de imágenes, facilitó un flujo de trabajo eficiente y modular. Estas herramientas demostraron ser esenciales para abordar proyectos de esta complejidad.

6. **Pruebas y Validación Global del Sistema:** Los resultados obtenidos en las pruebas globales y específicas (como el testbench de la interfaz UART) confirmaron la correcta integración y sincronización entre los bloques. Esto asegura la robustez y fiabilidad del sistema para aplicaciones prácticas.

Referencias

[1] Clifford Wolf, "PicoRV32: A Small 32-bit RISC-V CPU Core", YosysHQ GitHub Repository. [Online]. Disponible en: https://github.com/YosysHQ/picorv32

[2] RISC-V International, "RISC-V GitHub Repository". [Online]. Disponible en: https://github.com/riscv/

[3] AMD, "Vivado Design Suite". [Online]. Disponible en: https://www.xilinx.com/products/design-tools/vivado.html

[4] Python Software Foundation, "Python Programming Language". [Online]. Disponible en: https://www.python.org/

[5] draw.io, [En línea]. Disponible en: https://www.draw.io

[6] Microsoft Corporation, "Visual Studio Code". [Online]. Disponible en: https://code.visualstudio.com/