TEC | Tecnológico de Costa Rica

Instituto Tecnológico de Costa Rica
Escuela de Ingeniería en Computadores
Fundamentos de Arquitectura de Computadores (CE1107)

Bitácora del proyecto:

Máquinas de estados y protocolo de comunicación SPI

Profesor:
Luis Alberto Chavarría Zamora

Estudiantes:
Adriel Sebastián Chaves Salazar
Ana Melissa Vásquez Rojas
Daniel Duarte Cordero
Sergio Salazar Núñez

II Semestre 2024

TABLA DE CONTENIDOS

1. Introducción

Este documento presenta la bitácora detallada del proyecto grupal 1 realizado como parte del curso de Fundamentos de Arquitectura de Computadores. El objetivo principal de esta bitácora es registrar de manera minuciosa todos los procedimientos, decisiones y avances alcanzados durante el desarrollo del proyecto.

A lo largo de las próximas páginas, se ofrecerá una descripción exhaustiva de los procesos implementados en el diseño del proyecto, incluyendo las tablas empleadas para la organización de datos, las ecuaciones booleanas utilizadas en la lógica del sistema, así como las herramientas de automatización que fueron clave para optimizar el desarrollo.

La bitácora está organizada cronológicamente, con una clara división por días de trabajo, lo que permitirá al lector seguir de manera coherente la evolución del proyecto y comprender el flujo de trabajo adoptado. Cada entrada diaria incluirá:

- Objetivos planteados para el avance.
- Descripción de las tareas realizadas.
- Problemas encontrados y soluciones aplicadas.
- Resultados obtenidos.
- Reflexiones sobre el avance del proyecto.

Este enfoque sistemático no solo proporcionará un registro exhaustivo del trabajo realizado, sino que también servirá como una valiosa herramienta de aprendizaje y evaluación, tanto para el estudiante como para los instructores. A través de esta bitácora, se podrá apreciar el proceso de toma de decisiones, la aplicación práctica de los conocimientos teóricos y el desarrollo de habilidades críticas en el campo de [área específica del proyecto].

2. Semana 1: Estudio del problema y conceptos

2.1. Fecha: 01/10/2024

2.1.1. Objetivos

- Investigar el diseño y funcionamiento de una ALU (Unidad Aritmético-Lógica).
- Analizar los componentes y las operaciones básicas que debe realizar la ALU: suma, resta, AND, OR.
- Explorar diferentes métodos para la implementación de banderas de carry y overflow.

2.1.2. Descripción de las tareas

Se investigó sobre las ALUs, que son componentes clave dentro de la arquitectura de un procesador, responsables de realizar operaciones aritméticas y lógicas. En particular, se estudió cómo implementar operaciones como suma, resta, AND y OR utilizando verificación lógica con herramientas como Quartus. Además, se determinó cómo implementar las banderas (carry, overflow) para indicar condiciones especiales después de una operación, lo cual es esencial para el correcto funcionamiento de una ALU.

2.1.3. Reflexión sobre el estudio

El análisis permitió una mejor comprensión de cómo las operaciones aritméticas y lógicas afectan a los distintos registros en una ALU. También se identificaron las mejores prácticas para la implementación lo que resultará clave en la fase de simulación y verificación del proyecto.

2.2. Fecha: 02/10/2024

2.2.1. Avance del proyecto: Implementación inicial en Quartus

Se implementaron las operaciones básicas de la ALU en el software Quartus, específicamente las operaciones de suma, resta, AND y OR. Se utilizaron módulos separados para cada operación con el objetivo de modularizar el diseño y facilitar las pruebas posteriores. Durante

esta fase, se integró también un módulo de simulación para probar cada operación individualmente.

2.2.2. Problemas encontrados y soluciones aplicadas

- **Problema:** Dificultades para asegurar que las operaciones de resta y suma manejaban correctamente sin la bandera de carry.
  - **Solución:** Necesidad de implementar la bandera de carry.

2.3. Fecha: 03/10/2024

2.3.1. Avance del proyecto: Primer commit con ALU completa

Se realizó el commit con la implementación completa de la ALU, incluyendo las operaciones aritméticas y lógicas junto con la bandera de carry. En esta versión inicial, se verificó la funcionalidad de cada operación mediante simulaciones.

2.3.2. Problemas encontrados y soluciones aplicadas

- **Problema:** Algunos errores en la propagación del carry en operaciones.
  - **Solución:** Se ajustó la lógica de los módulos para mejorar la propagación del carry, realizando pruebas hasta lograr resultados consistentes.

2.4. Fecha: 06/10/2024

2.4.1. Objetivos

- Estudiar los conceptos de UART, PWM (Pulse Width Modulation) y el proceso de ”handshake” para su implementación en hardware.
- Investigar el protocolo UART y su uso para la comunicación asíncrona entre dispositivos, asegurando una transmisión fiable de datos.
- Comprender cómo la modulación por ancho de pulso (PWM) controla la potencia entregada a dispositivos como motores y su relevancia en sistemas embebidos.

- Analizar el proceso de ”handshake” y su importancia para verificar la integridad de la comunicación UART.

2.4.2. Descripción de las tareas

Se realizó una revisión profunda del protocolo UART, que es esencial para la transmisión asíncrona de datos entre dispositivos. A diferencia de la comunicación síncrona, UART no requiere una señal de reloj compartida; en su lugar, utiliza tasas de baudios predefinidas para sincronizar el envío y recepción de datos. Se estudiaron las características clave del UART, incluyendo su estructura de trama (bit de inicio, bits de datos, bit de paridad opcional y bits de parada) y los mecanismos de control de flujo, que permiten la comunicación entre dispositivos de diferentes velocidades sin pérdida de datos.

Además, se exploró el concepto de PWM, que se utiliza para controlar la potencia entregada a dispositivos como motores mediante la modulación del ciclo de trabajo de una señal. Se entendió cómo este método ajusta la energía suministrada mientras mantiene una frecuencia constante, siendo ampliamente utilizado para regular la velocidad de motores y la intensidad de LEDs. Finalmente, se investigó el proceso de ”handshake” en el contexto de UART, que se utiliza para confirmar que ambas partes están listas para intercambiar datos antes de comenzar la transmisión. Esto asegura que la comunicación sea confiable y que no haya pérdida de datos durante el intercambio.

2.4.3. Problemas encontrados y soluciones aplicadas

- **Problema:** Inicialmente, hubo dudas sobre cómo asegurar que las tasas de baudios estuvieran correctamente sincronizadas entre el transmisor y el receptor en UART.
  - **Solución:** Se revisaron las especificaciones de sincronización de UART, destacando la importancia de configurar correctamente los parámetros de ‘CLKS PER BIT‘ para garantizar que ambos dispositivos utilicen la misma tasa de baudios.

2.4.4. Resultados obtenidos

Se adquirió una comprensión sólida del protocolo UART, enfocada en su implementación para garantizar una comunicación asíncrona fiable entre dispositivos. También se comprendió la aplicación del PWM para el control de potencia y el papel del ”handshake” en

UART para asegurar la integridad de la comunicación. Estas bases teóricas serán fundamentales para el desarrollo de los módulos de transmisión y recepción en la FPGA, asegurando que la comunicación entre dispositivos sea robusta y eficiente.

2.4.5. Reflexión sobre el avance del proyecto

El estudio profundo del protocolo UART y el proceso de ”handshake” ha sido esencial para establecer una base técnica sólida. La claridad en el funcionamiento de la comunicación asíncrona permitirá una implementación precisa de los módulos de transmisión y recepción. La próxima etapa consistirá en el diseño estructural de estos módulos, asegurando que todos los elementos aprendidos se integren correctamente para lograr una comunicación eficiente y sin errores en el sistema.

2.5. Fecha: 07/10/2024

2.5.1. Objetivos

- Implementar los principios de programación para la comunicación serial UART, asegurando que los módulos sean estructurales y funcionen de manera robusta.
- Configurar e integrar la transmisión UART (TX) en la FPGA.
- Probar la sincronización precisa entre los módulos de UART y el sistema de reloj para mantener una transmisión eficiente y sin errores.
- Documentar todos los avances, incluyendo la configuración de CLKS PER BIT para asegurar la tasa de baudios correcta.

2.5.2. Descripción de las tareas

La implementación se centró en el desarrollo y prueba del módulo de transmisión UART (‘uart_tx‘) estructurado. El módulo utiliza una máquina de estados finita (FSM) para gestionar el proceso completo de transmisión de datos seriales: desde el estado de espera (‘s_IDLE‘), pasando por la transmisión del bit de inicio (‘s_TX_START_BIT‘), los bits de datos (‘s_TX_DATA_BITS‘), el bit de parada (‘s_TX_STOP_BIT‘), hasta la limpieza final (‘s_CLEANUP‘). Se aseguraron de que todas las transiciones estuvieran bien sincronizadas con el reloj del sistema.

En la configuración del módulo, se estableció un parámetro de ‘CLKS_PER_BIT = 5208‘, calculado para lograr una tasa de baudios de 9600 con un reloj de 50 MHz, asegurando que la transmisión mantuviera la sincronización precisa entre el dispositivo emisor y receptor. La FSM controla las señales de ‘o_Tx_Active‘, ‘o_Tx_Serial‘, y ‘o_Tx_Done‘, permitiendo la monitorización del estado de la transmisión y la gestión correcta de los datos enviados.

Además, se realizaron pruebas unitarias en Quartus para verificar la transmisión correcta y la sincronización de las señales en diferentes etapas del ciclo de transmisión. Las simulaciones se llevaron a cabo con una configuración de prueba (‘uart_tx_tb‘), que permitió confirmar que la línea UART operaba de manera continua y sin interrupciones en cada bit transmitido.

2.5.3. Problemas encontrados y soluciones aplicadas

- **Problema:** Durante las pruebas iniciales, se detectó una desincronización en la transmisión de bits causada por una configuración incorrecta del contador de reloj.
  - **Solución:** Se ajustó el valor de ‘CLKS_PER_BIT‘ y se verificó que coincidiera exactamente con la frecuencia del reloj del sistema y la tasa de baudios deseada. Se realizaron pruebas adicionales para garantizar la estabilidad en cada ciclo de transmisión.
- **Problema:** Hubo errores en el manejo del índice de bits durante la transmisión, lo que provocaba que no todos los bits fueran enviados correctamente.
  - **Solución:** Se corrigió la lógica del ‘r_Bit_Index‘ para asegurar que cada bit se transmitiera en el orden correcto. Se implementaron verificaciones adicionales en la FSM para garantizar que la transmisión pasara sin interrupciones a través de todos los estados definidos.

2.5.4. Resultados obtenidos

El módulo ‘uart_tx‘ fue implementado y probado, logrando una transmisión confiable de 8 bits de datos con todas las señales de control funcionando. Las simulaciones confirmaron que la transmisión ocurría de manera estable y sincronizada, y que los ajustes de la FSM permitieron una operación eficiente.

2.5.5. Reflexión sobre el avance del proyecto

La implementación del módulo de transmisión UART en la FPGA marcó un paso significativo en la comunicación serial del proyecto.

2.6. Fecha: 08/10/2024

2.6.1. Objetivos

- Mejorar y optimizar el funcionamiento del sistema UART para asegurar al menos un 80% de operatividad.
- Verificar y ajustar la transmisión de 8 bits de datos mediante UART, asegurando la sincronización precisa y una transmisión robusta.
- Validar los cambios implementados a través de simulaciones adicionales utilizando herramientas como Quartus y Verilog.

2.6.2. Descripción de las tareas

Se realizaron ajustes y mejoras en el módulo de transmisión UART (‘uart_tx‘), con el objetivo de lograr una transmisión confiable de 8 bits. Se integraron cambios clave, como la configuración correcta de los pines de transmisión (‘txPin‘) y recepción (‘rxPin‘) para UART en hardware, asegurando que los datos enviados y recibidos fueran consistentes.

Para facilitar la transmisión, se implementó el uso de una UART por software (bit-banging), ajustando los tiempos precisos para cada bit a través de la función ‘sendByte‘. La función garantiza que cada bit se envíe en el momento correcto, y que los bits de inicio y parada se respeten para mantener la integridad de la comunicación. Se añadieron mejoras en la sincronización del módulo, donde el parámetro ‘CLKS_PER_BIT‘ fue cuidadosamente calibrado para mantener la tasa de baudios correcta, minimizando la posibilidad de desincronización. También se añadieron funciones de recepción que leen cada bit con precisión temporal para garantizar que los datos recibidos coincidan exactamente con los enviados.

Las pruebas de simulación incluyeron el uso del simulador Questasim integrado con Quartus, donde se pudo validar el funcionamiento del sistema en condiciones simuladas, asegurando que el flujo de datos entre el módulo emisor y receptor fuera coherente y eficiente.

2.6.3. Problemas encontrados y soluciones aplicadas

- **Problema:** Inicialmente, la transmisión de 8 bits no se completaba correctamente debido a problemas de desincronización en la señal de reloj.
  - **Solución:** Se ajustaron los parámetros de temporización en la función de transmisión por software para asegurar que cada bit se enviara en la ventana de tiempo correcta. Se verificó que los valores de 'CLKS_PER_BIT' fueran adecuados para la tasa de baudios configurada de 9600.
- **Problema:** Las señales de control no reflejaban adecuadamente el final de la transmisión, lo que generaba inconsistencias en la recepción de los datos.
  - **Solución:** Se revisó la lógica de la FSM ('Finite State Machine') del módulo 'uart_tx', asegurando que las transiciones entre los estados 'IDLE', 'TX_START_BIT', 'TX_DATA_BITS', y 'TX_STOP_BIT' fueran suaves y precisas. Además, se añadieron señales adicionales para indicar la finalización de la transmisión.

2.6.4. Resultados obtenidos

Se logró una mejora significativa en el funcionamiento del sistema UART, alcanzando un 80% de operatividad. La transmisión de los 8 bits fue validada exitosamente en simulaciones, donde se pudo observar que las señales 'o_Tx_Serial', 'o_Tx_Active', y 'o_Tx_Done funcionaban como se esperaba, permitiendo una transmisión estable y sin errores. También se confirmó que las funciones de recepción sincronizaban correctamente la señal, minimizando el riesgo de pérdida de datos.

2.6.5. Reflexión sobre el avance del proyecto

Con las optimizaciones realizadas, el sistema UART ha progresado considerablemente.

2.7. Fecha: 11/10/2024

2.7.1. Avance del proyecto: Modificación de la ALU y adición de banderas

En este día se modificó la ALU para reducir la cantidad de hardware necesario, optimizando los recursos utilizados. Además, se añadieron más banderas, incluyendo una bandera para

indicar cuando el resultado es cero (zero flag) y una para overflow. Las pruebas fueron satisfactorias y se mejoró la eficiencia del diseño.

2.7.2. Problemas encontrados y soluciones aplicadas

- **Problema:** La reducción de hardware puede afectar la precisión de algunas operaciones.
  - **Solución:** Se optimizó el código, realizando simulaciones para confirmar que las optimizaciones no afectaban la precisión de los resultados.

2.8. Fecha: 12/10/2024

2.8.1. Objetivos

- Completar el desarrollo del sistema UART para asegurar una operatividad del 100%.
- Integrar la funcionalidad completa de transmisión (TX) y recepción (RX) en el sistema UART.
- Realizar pruebas exhaustivas para validar la funcionalidad completa del sistema.

2.8.2. Descripción de las tareas

El módulo UART fue completado con presencia de secciones comportamentales y con la integración de las funcionalidades de transmisión y recepción. Se realizaron ajustes finales en el módulo de recepción (RX), asegurando que la sincronización entre la transmisión y la recepción se realizara de manera eficiente. En particular, se ajustaron los contadores de reloj y la lógica de control basada en los parámetros de reloj para garantizar que los bits se recibieran correctamente. Las pruebas incluyeron la verificación de los submódulos ‘uart_tx‘ y ‘uart_rx‘, así como la sincronización de señales entre ambos. Se implementaron cambios adicionales para la validación de los bits de inicio y parada, y se optimizó el uso de flip-flops manuales para la gestión de estados.

2.8.3. Problemas encontrados y soluciones aplicadas

- **Problema:** La sincronización entre los módulos de transmisión y recepción no era precisa.

  - **Solución:** Se realizaron ajustes en los contadores de reloj y las señales de control para garantizar una correcta sincronización entre los módulos, utilizando la lógica FSM para asegurar una transmisión fluida.

- **Problema:** Algunos datos no se recibían correctamente durante la operación continua.

  - **Solución:** Se ajustó la lógica de recepción para mejorar la precisión en la detección de los bits de inicio y parada. Se implementaron cambios en el submódulo ‘uart_rx_sync‘ para mejorar la sincronización de la señal y se ajustaron los multiplexores de entrada y salida.

2.8.4. Resultados obtenidos

Las funcionalidades de transmisión y recepción operaron sin errores, logrando una operatividad del 100%. Las pruebas verificaron la correcta interacción entre los módulos ‘uart_tx‘ y ‘uart_rx‘, asegurando que los datos transmitidos fueran recibidos y decodificados correctamente. Se implementó además el proceso de handshake, que fue verificado para garantizar la integridad de la conexión serial.

2.8.5. Reflexión sobre el avance del proyecto

La integración de transmisión y recepción, junto con las pruebas exhaustivas, asegura un buen funcionamiento.

2.9. Fecha: 13/10/2024

2.9.1. Avance del proyecto: Integración de la FSM para control de la ALU

Se realizó la implementación de una Máquina de Estados Finitos (FSM) para controlar las operaciones de la ALU. Esta FSM decide cuál operación realizar dependiendo de la entrada recibida, facilitando el control secuencial de la ALU. La FSM se integró con los módulos de la ALU previamente desarrollados.

2.9.2. Problemas encontrados y soluciones aplicadas

- **Problema:** La FSM no siempre seleccionaba correctamente la operación lógica a realizar.
  - **Solución:** Se depuró el código de la FSM, asegurando que los estados de entrada se procesaban correctamente para seleccionar la operación adecuada.

2.10. Fecha: 13/10/2024

2.10.1. Objetivos

- Completar el desarrollo del sistema de comunicación serial UART de manera estructural.
- Integrar completamente el modelo estructural del UART, asegurando el correcto funcionamiento de transmisión (TX) y recepción (RX).
- Optimizar el uso de recursos mediante la implementación de multiplexores y otros módulos estructurales.
- Documentar los avances y realizar simulaciones para validar el correcto funcionamiento.

2.10.2. Descripción de las tareas

El trabajo se desarrolló en tres avances importantes para completar la comunicación serial UART de manera estructural al 100%:

Avance 1: Desarrollo de la comunicación serial completo, integración del modelo estructural
Se comenzó con la implementación del módulo 'Timer', que fue crucial para la sincronización de señales y generación de pulsos de 1Hz, permitiendo un control preciso del envío de datos. Este módulo incluyó el uso de contadores BCD y comparadores para detectar valores específicos, mejorando la sincronización general del sistema. Además, se trabajó en 'uart_tx_mux_3', que gestiona las señales de control 'o_Tx_Active' para definir cuándo el transmisor está activo. Se ajustaron componentes adicionales como 'uart_tx_pkg' para definir los estados de la FSM y facilitar la integración del modelo estructural. Se realizaron pruebas iniciales para verificar la estabilidad de las señales y la correcta activación/desactivación de la transmisión, asegurando que la lógica estructural respetara las transiciones del estado 's_IDLE' a 's_TX_START_BIT'.

Avance 2: Desarrollo de la UART TX estructural completa
Se avanzó con la integración completa del módulo de transmisión ‘uart_tx‘. Se añadieron módulos críticos como ‘uart_tx_mux_5‘, que se diseñó como un multiplexor 2:1 para gestionar la selección de estados en la FSM, basándose en la estructura modular. Se optimizó la sincronización entre la señal de reloj y las operaciones de transmisión mediante la implementación de submódulos como ‘uart_tx_clk_counter‘ y ‘uart_tx_bit_index‘, asegurando que cada bit se transmitiera en el momento adecuado. Se realizaron simulaciones para validar la funcionalidad estructural y se confirmaron transiciones suaves entre los estados de transmisión.

Avance 3: Desarrollo de la implementación de la comunicación serial completa al 100%
Se completó la implementación general del sistema de comunicación UART, incluyendo la integración de los módulos de recepción y la estructura completa del sistema. Se utilizó ‘D_FF_Cell‘ para la gestión de flip-flops D que ayudaron a evitar problemas de sincronización. Se desarrollaron módulos adicionales como ‘uart_rx_sync‘ para asegurar la correcta alineación de los datos recibidos, reemplazando bloques con estructuras más eficientes y seguras para la recepción de señales. Además, se añadieron multiplexores adicionales (‘uart_rx_mux_1‘ a ‘uart_rx_mux_10‘) que permitieron una gestión precisa de la selección de señales, optimizando el flujo de datos y la transición entre estados dentro de la FSM. Se llevaron a cabo simulaciones para validar el diseño estructural completo, asegurando que tanto la transmisión como la recepción funcionaran de manera eficiente y sin errores.

2.10.3. Problemas encontrados y soluciones aplicadas

- **Problema:** La sincronización entre los diferentes estados de la máquina de estados finita (FSM) no era precisa en algunas simulaciones, lo que causaba errores durante la transmisión.
  - **Solución:** Se implementó un ajuste en la lógica del módulo de multiplexores ‘uart_tx_mux_5‘, permitiendo una transición controlada y precisa entre los estados definidos en ‘uart_tx_pkg‘.
- **Problema:** El consumo de recursos de hardware era mayor de lo esperado debido a la lógica combinacional no optimizada, aumentando el uso de compuertas lógicas.
  - **Solución:** Para optimizar la implementación estructural, se reorganizaron los componentes básicos, utilizando multiplexores y flip-flops tipo D para reducir el consumo de recursos.

2.10.4. Resultados obtenidos

Se completó la estructura del sistema de comunicación UART para la transmisión y recepción, logrando una integración total del modelo estructural. Los módulos de multiplexores y la FSM se integraron correctamente, optimizando el uso de recursos y asegurando una comunicación eficiente y sincronizada. La integración de los flip-flops D permitió una reducción significativa en el uso de compuertas lógicas, lo que mejoró el rendimiento general del sistema. La implementación estructural completa permite un flujo bidireccional y seguro de datos.

2.10.5. Reflexión sobre el avance del proyecto

El avance logrado en la integración del sistema UART y la implementación estructural es significativo.

3. Semana 3: Validación de Soluciones

3.1. Fecha

References