Instituto Tecnológico y de Estudios Superiores de Monterrey
(Campus Guadalajara)

Diseño con lógica programable (Gpo 101)

Evidencia 1 : Reporte de Proyecto Final

Docentes:
Gonzalo Gutiérrez Ramos
Gilberto Ochoa Ruiz
Hiram Rayo Torres Rodríguez

Alumnas:
Elizabeth Jáuregui Zárate (A01253381)
Vanessa Cerda Carrillo (A01612852)
Ana María Rodríguez Peña (A01741831)

Fecha de entrega:
16/03/2025

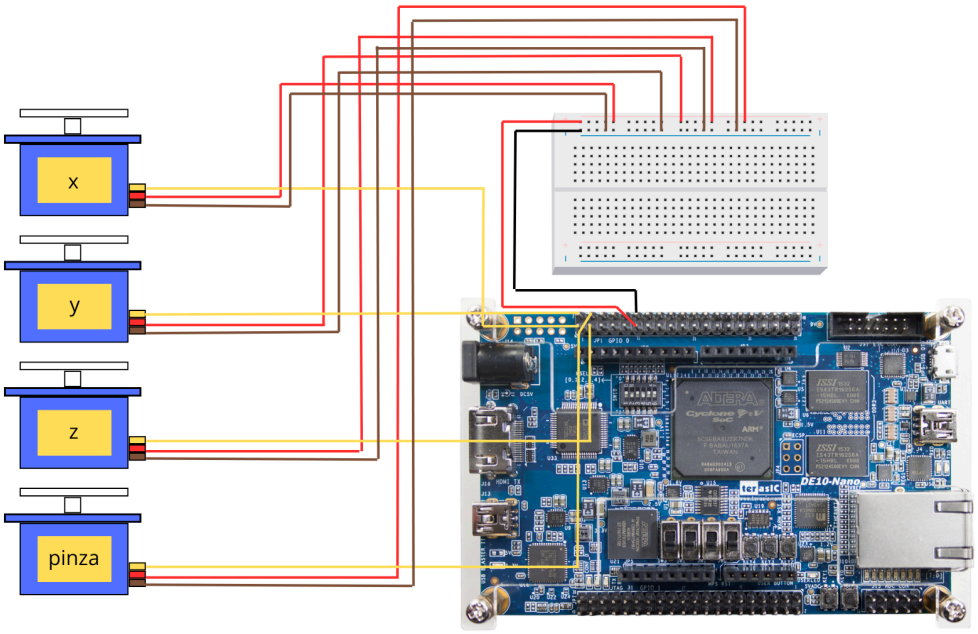
Resumen

El presente reporte técnico busca detallar el proceso llevado a cabo para el funcionamiento de un brazo robótico, desde sus especificaciones hasta su validación y prueba. El proyecto consta del desarrollo pleno de un brazo robótico controlado mediante un FPGA (Field Programmable Gate Array), específicamente una tarjeta DE10-Lite MAX10, que procesa datos de un acelerómetro para mover cuatro servomotores SG90, uno de ellos destinado al funcionamiento de una pinza y los otros tres destinados al movimiento del brazo en tres diferentes grados de movimiento (X, Y y Z).

El objetivo principal de este proyecto es demostrar la capacidad de las tarjetas FPGA de llevar a cabo la lectura de datos del acelerómetro interno, en los ejes X, Y y Z y convertirlos en señales PWM para controlar los servomotores cada 10 milisegundos y desplegar estos valores de lectura en una pantalla utilizando una interfaz VGA. De igual manera se tiene como requisito implementar una memoria ROM para ejecutar movimientos predefinidos sin depender del acelerómetro y así mejorar la precisión y automatización del brazo robótico.

En la planeación del proyecto, se establecieron especificaciones, la cuales constaban del control exitoso de tres servomotores con base a los datos del acelerómetro y la comunicación entre el sensor y los servos usando la tarjeta DE10-Lite. En la fase del modelado, se diseñó el diagrama de bloques lógicos para definir la interacción entre módulos y componentes internos, al igual que un diagrama de interacción entre los distintos componentes electrónicos. Mientras que en la implementación se llevó a cabo toda la descripción de hardware necesaria, donde se emplearon herramientas como Quartus Prime. De igual manera se realizó el ensamblaje del brazo y la conexión de los componentes. Por último en la fase de validación y pruebas, se realizaron pruebas de respuesta del brazo a distintos valores de coordenadas, y se realizaron pruebas de pulsaciones (test bench) con el fin de corregir errores en la interpretación de datos y garantizar la correcta ejecución de los movimientos.

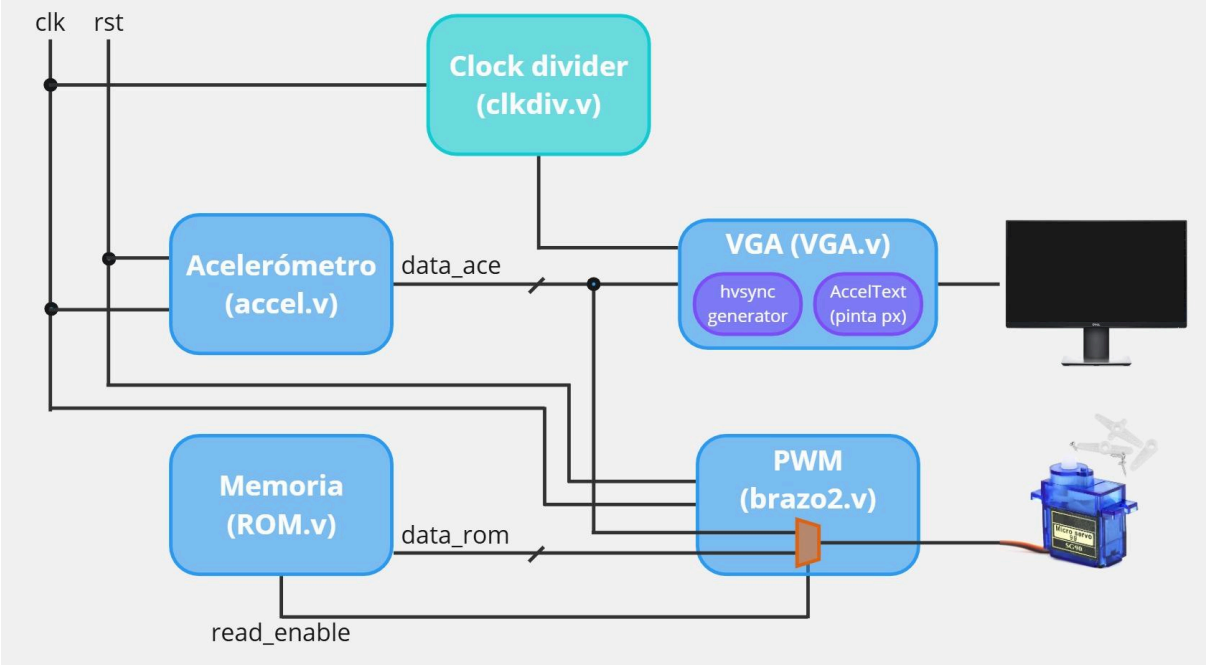
Diagrama de componentes electrónicos/físicos



Componentes electrónicos:

- FPGA DE10-Lite MAX10 (Placa de desarrollo)
- Servomotores SG90 (x, y, z, pinza)
- Protoboard (Placa de pruebas)
- Jumpers (Cables de conexión)
    - Color Rojo: Alimentación de 5V.
    - Color Negro: Alimentación de tierra.
    - Color Amarillo: Señales de control.
- Chasis de Brazo Robótico

Diagrama de bloques lógico



Descripción detallada de cada bloque

Acelerómetro (módulo "accel") (top) (wrapper):

- El módulo "accel.v" es el componente principal diseñado para interactuar con los datos de salida del acelerómetro interno de la tarjeta y realizar las conexiones entre módulos internos (módulo top). También proporciona una interfaz visual a través de una salida VGA para visualizar los datos cada 10 ms.

  Tiene como entrada los siguientes componentes:

  - Relojes (ADC_CLK_10, MAX10_CLK1_50, MAX10_CLK2_50)
  - Botones (KEY[1:0])
    - KEY[1]: Reset del sistema.
    - KEY[0]: Control del mecanismo de la pinza del brazo.
  - Interruptores (SW[9:0])
    - SW[0]: Valida la reproducción de movimientos predefinidos en la memoria ROM.
  - Acelerómetro (GSENSOR_CS_N, GSENSOR_INT[2:1], GSENSOR_SCLK, GSENSOR_SDI, GSENSOR_SDO)

    - Interfaz SPI para comunicarse con el sensor de aceleración.

Y tiene como salidas los siguientes componentes:

- Displays de 7 segmentos (HEX0 - HEX5)
    - (HEX0-HEX2): Valores del eje X.
    - (HEX3-HEX5): Valores del eje Y.
- LEDs (LEDR[9:0])
    - Valores del eje Z.
- Salidas GPIO (GPIO[35:0])
    - (GPIO[0] - GPIO[3]): Control de servomotores.
- Salida VGA (VGA_R, VGA_G, VGA_B, VGA_HS, VGA_VS)
    - Muestra visualmente los datos del acelerómetro en un monitor.

El procesamiento del acelerómetro consta de una conversión SPI, que se encarga de obtener los valores X,Y y Z del acelerómetro (spi_control) y mediante las señales de reloj generadas por PLL, se utiliza un contador para actualizar estos valores cada 2.5 millones de ciclos y guardarlos en registros. Por su parte, hace la conexión interna de estos valores de X,Y y Z a módulos como VGA (módulo VGA.v) y PWM (módulo brazo2.v). Al igual que a la memoria ROM (módulo ROM.v).

Memoria (módulo "ROM")

- El módulo "ROM.v" define una memoria ROM parametrizable, que almacena y distribuye valores predefinidos de tres conjuntos de datos (data_x, data_y, data_z).

  Tiene como entrada los siguientes componentes:
    - "ce" (Chip Enable)
        - Valida el acceso a la ROM.
    - "read_en" (Read Enable)
        - Valida el acceso a la lectura de datos.
    - "address"
        - Dirección de la ROM.

  Y tiene como salidas los siguientes componentes:
    - "data_x"
        - Dato leído de la ROM en la dirección especificada para el eje X.
    - "data_y"
        - Dato leído de la ROM en la dirección especificada para el eje Y.

- “data_z”
    - Dato leído de la ROM en la dirección especificada para el eje Z.

Almacena tres bancos de datos o arreglos (mem_x, mem_y, mem_z) en memoria interna, cada uno con 2^(ADDRESS_WIDTH) posiciones de 16 bits. Se cargan archivos en formato hexadecimal (“ROM_x_hex.hex”, “ROM_y_hex.hex” y “ROM_z_hex.hex”), permitiendo leer valores predefinidos, en este caso, ocho datos para el movimiento de cada servomotor.

Clock divider (módulo “clkdiv”)

- Mediante un contador, genera una señal de reloj dividida (en este caso, de 25.175 MHz para el VGA) mediante un contador.

  Tiene como entrada los siguientes componentes:
    - “clk”
        - Reloj interno de 50 MHz.
    - “rst”
        - Reset del sistema.

  Y tiene como salida el siguiente componente:
    - “clk_div”
        - Señal de reloj dividida de 25.175 MHz.

VGA (módulos “VGA”, “hvsync_generator” y “AccelText”)

- El módulo hvsync_generator genera las señales de sincronización horizontal (hsync) y vertical (vsync) y administra contadores en X y en Y para delimitar el área de visualización (640×480 píxeles).

  El módulo “VGA.v” tiene como entrada los siguientes componentes:
    - “clk”
        - Reloj interno de 50 MHz.
    - “ax”
        - Dato de aceleración del eje X.
    - “ay”
        - Dato de aceleración del eje Y.
    - “az”
        - Dato de aceleración del eje Z.

Y tiene como salidas los siguientes componentes:

- “pixel”
    - Valor RGB del píxel a mostrar en pantalla.
- “hsync_out”
    - Señal de sincronización horizontal de VGA.
- “vsync_out”
    - Señal de sincronización vertical de VGA.

El módulo AccelText recibe datos reducidos del acelerómetro (ax, ay, az), extrae el signo y los dígitos (centenas, decenas y unidades) y, mediante funciones, genera patrones para representar los caracteres X, Y y Z, los símbolos = y -, y los dígitos del 0 al 9. Finalmente, el módulo VGA utiliza un clock divider para obtener un reloj de 25.175 MHz, integra los módulos anteriores y asigna los colores a los pines VGA del FPGA (mediante los pines VGA_R, VGA_G y VGA_B) para escribir el texto final en la pantalla.

PWM (módulo “brazo2”):

- Implementa la generación de señales PWM para el control de servomotores.

  Tiene como entradas las siguientes señales:
    - “clk”
        - Reloj interno de 50 MHz.
    - “rst”
        - Reset del sistema.
    - “read_en”
        - Señal de habilitación de la lectura en memoria.
    - “data_rom”
        - Dato leído de la memoria ROM (usado solo si la señal anterior está habilitada).
    - “data_ace”
        - Dato del acelerómetro (usado solo si la señal de lectura en memoria está deshabilitada).

  Y su salida es:
    - “pwm_out”
        - Señal conectada a cada servomotor para regular su movimiento.

Primero, el módulo decide si usar los datos guardados en las memorias ROM o los del acelerómetro según la señal de *read enable* (SW[0]). Calcula el valor absoluto del dato seleccionado y lo convierte en un pulso, restringido entre PULSE_MIN y PULSE_MAX. Un contador con un divisor de velocidad (SPEED_DIVIDER) ajusta el duty cycle hacia el valor objetivo (target_DC) para suavizar los movimientos del brazo. Finalmente, otro contador compara el duty cycle con el periodo PWM (PWM_PERIOD) para generar la señal de salida (pwm_out).

Desarrollo del reto

La estrategia principal que empleamos fue *divide y vencerás*, que nos permitió desarrollar de manera eficiente cada una de las partes del proyecto. Cada integrante del equipo se enfocó en un área específica, asegurando un trabajo más ágil y organizado. De esta manera se dividió el problema en subproblemas más pequeños y fáciles de solucionar, ahorrando así tiempo y eficientando la resolución del proyecto.

Comenzamos con la especificación, donde definimos los requerimientos del sistema: el control de tres servomotores mediante los ejes X, Y y Z del acelerómetro, la conversión de datos en movimiento mecánico y la comunicación a través de la tarjeta MAX10 DE10 LITE. Luego pasamos al modelado, en el que representamos el sistema mediante diagramas de bloques para visualizar la interacción entre los componentes. En la implementación, ensamblamos los componentes y programamos la lógica para leer los datos del acelerómetro y convertirlos en señales PWM, ajustando mecánicamente el brazo para garantizar rangos de movimiento adecuados. Posteriormente, realizamos pruebas cargando el código en la tarjeta y verificamos su desempeño en el brazo robótico, mientras un monitor conectado por VGA mostraba en tiempo real las lecturas del acelerómetro. Finalmente, en la fase de validación y pruebas, evaluamos la respuesta del brazo ante diferentes inclinaciones, corrigiendo errores como movimientos bruscos.

A lo largo del proceso, cuando surgían dificultades, colaboramos entre nosotras para resolverlas, permitiendo un progreso continuo hasta alcanzar el funcionamiento óptimo del brazo. Cada parte del equipo aportó al proyecto con habilidades y fortalezas específicas, que facilitaban la resolución de problemas y la integración eficiente de cada componente. Gracias a esta colaboración, logramos optimizar el diseño del programa completo, mejorar la precisión de los movimientos y asegurar la estabilidad del sistema.

Video del funcionamiento (minuto 2:43 a 4:05)

https://drive.google.com/file/d/1LMPzqL0qh5L919CeOGyJ1_Nz1ovrTL90/view?usp=sharing