DOCUMENTO DE DISEÑO

Taller de Diseño Digital

Tecnológico de Costa Rica

I Semestre 2021

Josué Cubero Montero
David Quesada Calderón
Gerald Valverde Mc kenzie

Este es un documento es con fines académicos para el curso de Taller de Diseño Digital impartido en el Tecnológico de Costa Rica, el cuál conforma la malla curricular de la carrera de Ingeniería en Computadores.

Tabla de contenido

Introducción

En este documento se describirá el problema de ingeniería en términos de requerimientos de diseño y limitantes, el estado del arte, estándares, normas, entre otros.

Problema

El proyecto planteado consiste en el diseño e implementación de un computador básico basado en la arquitectura ARMv4 por medio del lenguaje de descripción de hardware System Verilog.

El sistema consiste en un procesador que sea capaz de cargar datos desde una memoria ROM, estos datos consisten en caracteres de código ASCII de [0-126] que se encontraran encriptados con el fin de que estos datos puedan ser cargados y aplicarles tres posibles algoritmos de descifrado. Estos algoritmos de descifrado consisten en las operaciones XOR, NOT y suma 2. Finalmente, los datos descifrados deben almacenarse en una memoria para posteriormente ser desplegados en un monitor por medio de un controlador VGA.

Requerimientos

Las funcionalidades del sistema se encuentran representadas por medio del código F##, mientras que las tareas de cada funcionalidad son dadas por el código T##.

Datos precargados (F01)

Se debe implementa una memoria ROM para cargar los caracteres por descifrar.

T01: Implementación de un código en Python capaz de realizar la lectura de un archivo “.txt” con los datos cifrados.

T02: Elaboración de un código en el leguaje Python que sea capaz de transformar caracteres cifrados en un archivo de inicialización de memoria (MIF).

T03: Crear una memoria de tipo ROM en el leguaje de descripción capaz de cargar los caracteres cifrados.

T04: Realizar un testbench que permita evaluar el funcionamiento de la ROM.

Filtrado de Texto (F03)

Se debe implementar el filtrado de texto por medio de tres algoritmos.

T05: Se debe elaborar el algoritmo XOR el cuál tomará un número de 3 bits para aplicarle la operación XOR a cada uno de los caracteres de la ROM.

T06: A cada carácter en la ROM se le debe operar la función NOT a cada uno de sus bits.

T07: El tercer algoritmo consiste en sumarle 2 al dato almacenado.

T09: Se debe elaborar un módulo de filtrado el cual consiste en la implementación de un MUX para la selección de la operación deseada.

Controlador VGA (F04)

Los datos descifrados deben representarse por medio de un monitor, para dicha funcionalidad se debe implementar un controlador VGA capaz de leer la memoria de datos y representarlos en la salida RGB de la FPGA.

T10: Se debe implementar un divisor de la señal de reloj que permita que las señales de clk para el controlador VGA sean de 25 MHz.

T11: Se debe implementar un módulo de lectura de memoria el cual sea capaz de leer cada uno de los caracteres descifrados.

T12: Se debe implementar un método en Python capaz de convertir imágenes de 32 bits en archivos tipo MIF de anchura de 3 bits, siendo la profundidad de la memoria de 1024 bits.

T13: Se debe crear memorias capaces de almacenar desde el ASCII 32 hasta el carácter ASCII 126.

T14: Implementación de MUX capaz de seleccionar la memoria por mostrar dado un caracter de entrada.

T15: Implementar un módulo capaz de decodificar los bits de la memoria ASCII en el código extendido de RGB de 28 bits para la salida a la VGA.

T16: Se debe realizar un mapeo de la pantalla para asignarle los caracteres de la memoria de datos.

T17: Implementación de un controlador VGA que permita inicializar las variables de H_SYNC, V_SYNC, SYN_B, BLANK_N

Start: Botón que inicia el proceso de filtrado. Se le debe aplicar una función antirebote.

Procesador (F05)

El procesador consiste en la implementación de un microcomputador empleando la arquitectura ARMv4.

T18: Se deben realizar las instrucciones de 32 bits del procesador basadas en el ARMv4 ISA, por medio de Visual ARM.

T19: Se deben parsear las instrucciones ARM a sus respectivos valores en hexadecimal por medio del software CPUlator.

T20: Se debe implementar una memoria de instrucciones de 32 bits, y precargarla con cada una de las instrucciones.

T21: Se debe realizar un Register file para almacenar cada uno de los 15 registros empleados en las instrucciones ARM.

T22: Implementación de una ALU encargada de cada una de las operaciones del sistema.

T20: Se debe realizar una memoria de datos en la que se almacenen cada uno de los caracteres descifrados.

T21: Se debe implementar una Unidad de Control del procesador.

UI (F06)

El usuario debe interactuar el sistema por medio de la interfaz de la FPGA.

T22: Se debe implementar un botón de inicio del sistema con la función anti rebote.

T23: Se debe implementar un botón de reset del sistema con la función anti rebote.

T24: Se deben realizar simulaciones a cada una de las etapas del sistema.

Estado del arte

Ensamblador

El lenguaje ensamblador es un leguaje de bajo nivel, en el caso de ARM este viene definido por el ISA, en este se realizan operaciones básicas a registros memorias y valores constantes (Harris,2015, p.297).

ARMv4

ARM es una de las microarquitecturas de procesadores más empleadas a nivel mundial. El primer procesador ARM1 fue desarrollado por Acorn Computer para micr computadoras BBC Micro1985. Tras una serie de mejoras en 1993 ARMv4 fue introducido como una arquitectura de 32 bits que permite realizar las funciones load y store para partes de los datos (Harris,2015, p.350).

ISA

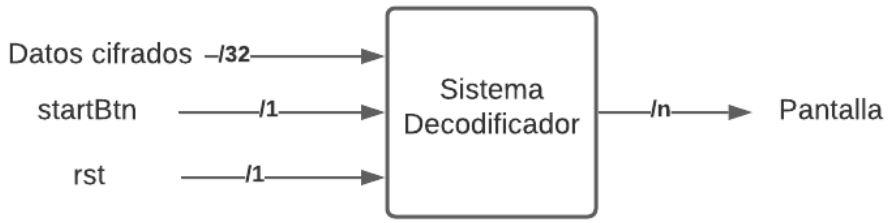
Por sus siglas en inglés Instruction Set Arquitecture se refiere al conjunto de instrucciones que permiten controlar el CPU por medio de software (Ltd. ARM, 2021).
El set de instrucciones se emplea como insumo para elaborar el software que controla el sistema, a su vez es de suma importancia conocer la estructura interna de cada una de las instrucciones para poder acoplar de manera adecuada cada uno de los componentes de hardware del sistema.

Diseño del sistema

Diagramas

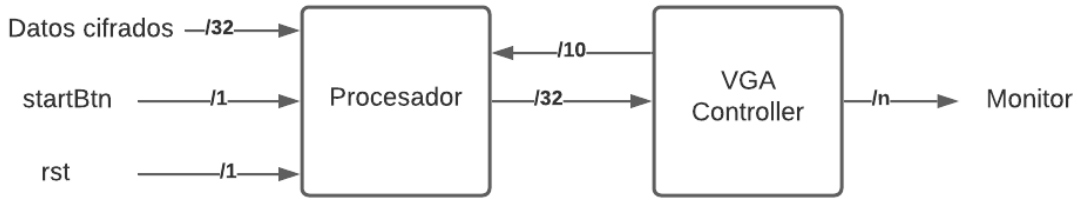
En la fase de diseño, se emplea el uso de diagramas modulares para mostrar una abstracción del sistema.

Diagrama de primer nivel



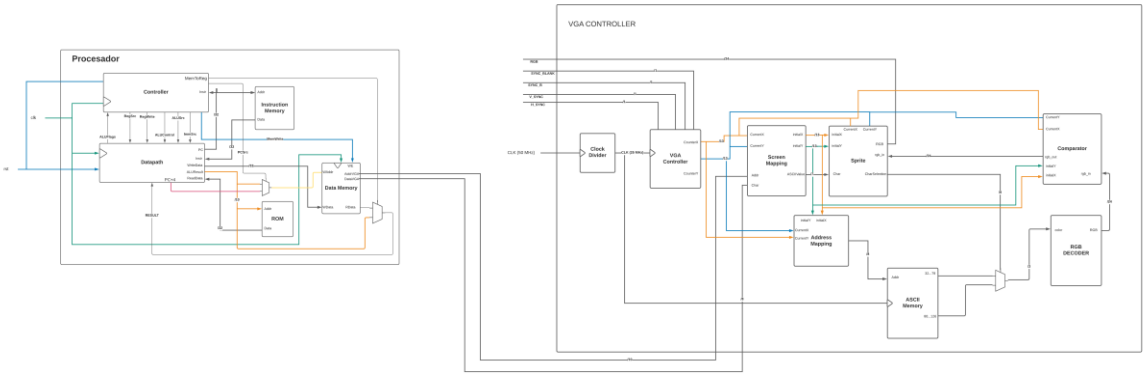
Como se observa el sistema consiste un módulo decodificador de datos que tiene como entrada datos ASCII cifrados, al operar el botón startBtn comienza a descifrar los datos para de manera posterior mostrarlos en la pantalla por medio de un conector VGA.

Diagrama de segundo nivel



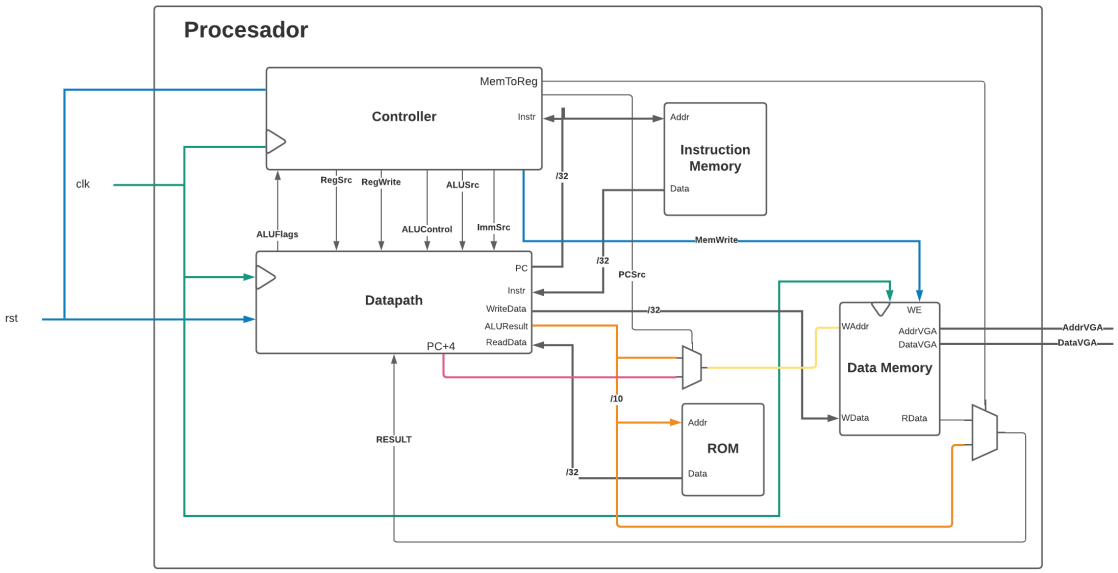
Como se observa el sistema consiste en dos módulos principales un procesador encargado de decodificar todos los datos y seguidamente los datos descifrados son obtenidos por el controlador VGA y este se encarga de mostrar los datos en pantalla.

Diagrama de tercer nivel



De manera más detallada se observa la fase del procesador en el siguiente diagrama.

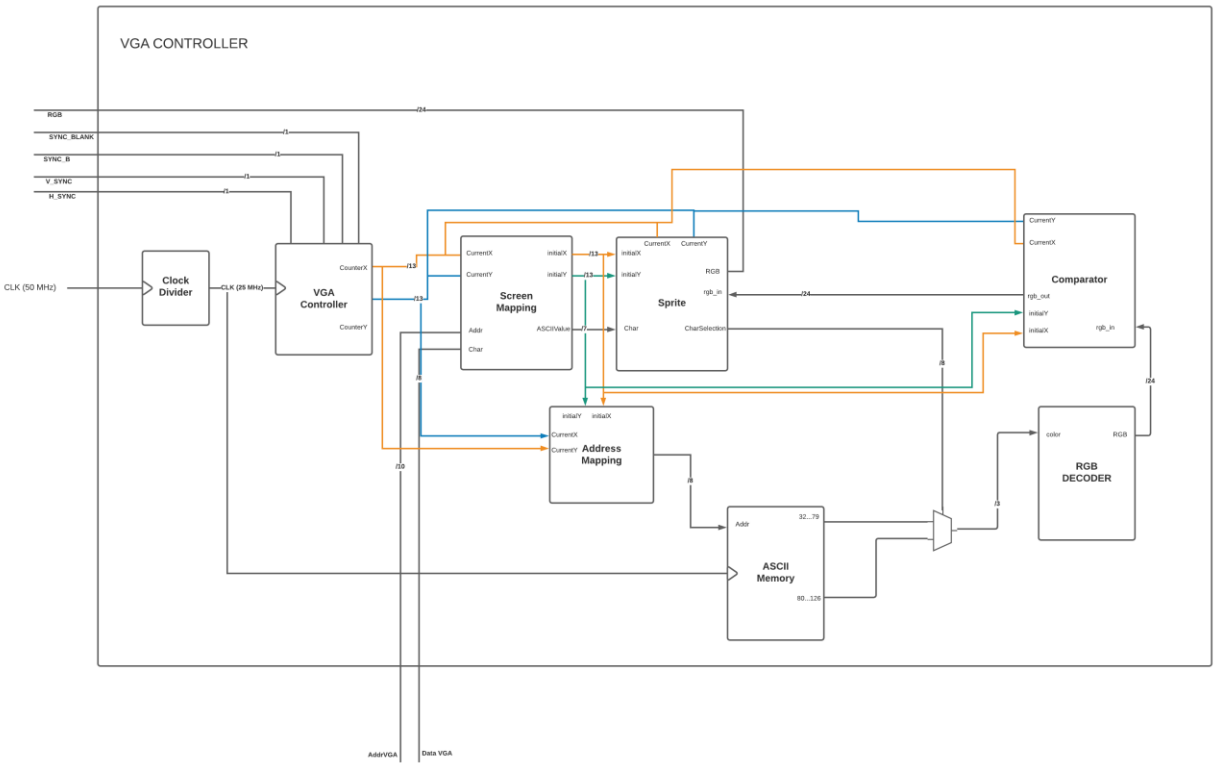
Módulo del procesador.



Como se observa el procesador es basado en la arquitectura ARMv4 en el que se emplea una memoria de instrucciones de 32 bits, una memoria de tipo ROM en la que se realizará la lectura de los datos cifrados.

La microarquitectura del procesador se encuentra dividida en dos módulos los cuáles interactúa entre sí. Las cuales son el Datapath y la Unidad de Control. El módulo Datapath opera los bits de los datos, este contiene diferentes estructuras de memoria, registros y multiplexores. La Unidad de Control recibe los datos de la instrucción actual y le dice a la Datapath como ejecutar la instrucción. La Unidad de Control activa los registros, la selección de los multiplexores y las señales de escritura de las señales.

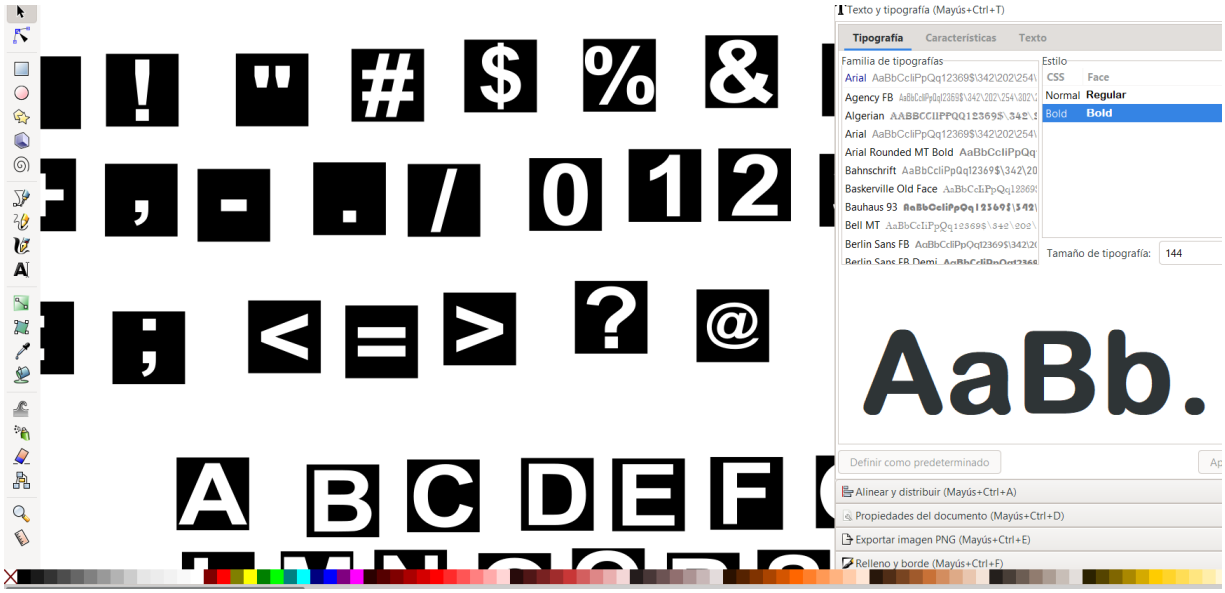
Módulo del controlador VGA.



El VGA Controller es el encargado de recopilar los datos descifrados por el procesador y mostrarlos en un monitor. Este módulo recibe la señal de reloj dada por la FPGA la cual es de 50 MHz y la divide a 25 MHz debido a que esta es la frecuencia necesaria para la VGA. Seguidamente se encuentra el módulo VGA Controller el cual activa las señales para el conector VGA (SYNC_BLANK, H_SYNC, V_SYNC y SYNC_B), además de realizar el contador de los pixeles en el eje x y eje y. Los valores de estos contadores se utilizan para elaborar un mapeo de la pantalla asignando el valor del carácter ASCII en un cuadrante determinado. El módulo Address Mapping es el encargado de realizar el mapeo de los caracteres dada la posición de píxel actual. Los caracteres ASCII son almacenados en memorias de tipo ROM, dando como resultado 3 bits que simbolizan el color. Este color de 3 bits pasa por otro de 24 bits para la señal de salida de la VGA.

Código ASCII

Para la elaboración del proyecto se implementa el código ASCII estándar el cual está comprendido desde el carácter 32 hasta el 126. Para dicha implementación se elaboran imágenes de cada uno de los caracteres ASCII por medio del software libre Inkscape.



Posteriormente se procede a realizar el procesamiento de las imágenes por medio de Python convirtiendo cada píxel en datos de 3 bits los cuáles son almacenados en un archivo tipo MIF, los cuales nos permiten inicializar las memorias ASCII de tipo ROM.

Referencias

Ltd., ARM. (2021). Instruction Sets - Arm Developer. Accesado 15 Junio 2021, de https://developer.arm.com/architectures/instruction-sets

Sarah Harris y David Harris (2015). Digital design and computer architecture: arm edition. Morgan Kaufmann, 2015