TEC | Tecnológico de Costa Rica

Instituto Tecnológico de Costa Rica
Área Académica de Ingeniería en Computadores
CE 4301 — Arquitectura de Computadores I

Proyecto Grupal 1

Diseño e Implementación de un ASIP para la generación de gráficos y texto

Fecha de asignación:	15 septiembre de 2021	Fecha de entrega:	22 octubre de 2021
Grupos:	3-4 personas	Profesores:	Luis Chavarría Zamora

Mediante el desarrollo de este proyecto, el estudiante aplicará los conceptos de arquitectura de computadores en el diseño e implementación en hardware de un *Application Specific Instruction Set Processor* (ASIP) para generación de gráficos mediante un editor de texto. Atributos relacionados: **A**nálisis de **P**roblemas (AP), el cual se encuentra en **A**vanzado (A).

1. Descripción General: Generación de gráficos

Los gráficos por computador son un área de la informática visual, donde se emplean computadoras tanto para generar imágenes visuales sintéticamente como integrar o cambiar la información visual y espacial del mundo "real".

A pesar de que existen procesadores especializados para realizar su generación (GPU) para efectos prácticos de este proyecto se pretende crear un ASIP que permita la generación de gráficos 2-D por computador con el fin de crear una imagen a partir de estructuras geométricas simples (círculos, líneas, elipses, triángulos), En la Figura 1 se muestra el granero básico, generado mediante coordenadas. Este tipo de técnicas son usadas en imágenes vectoriales.

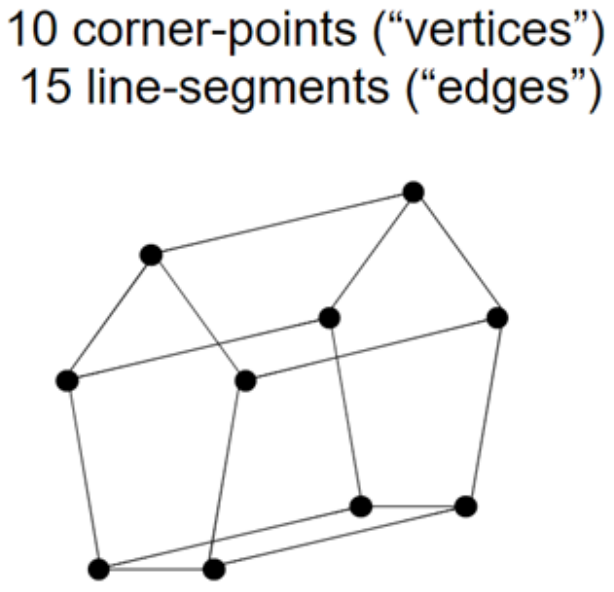

Figura 1: Granero básico generado mediante coordenadas

El sistema propuesto utilizará el algoritmo de Bresenham para generación de líneas y círculos principalmente. Del mismo modo puede revisar los siguientes enlaces:

- Rellenado de polígonos.
- Generación de figuras.
- Libro de generación de gráficos.

2. Especificación

Se le solicita desarrollar una **arquitectura** y una **microarquitectura** que realice los siguientes dos procesos:

1. **Generación de texto mediante gráficos**: Para esto solamente use líneas rectas para convertir un texto en memoria y representarlo en una imagen 250 × 250 binarizada con vectores de cada letra. Para cada letra del texto debe usar el alfabeto mostrado en la Figura 2. Represente cada letra en un espacio de 5×5 píxeles, con un espacio de 1 píxel a la derecha y abajo. El texto comienza en la esquina superior izquierda.

   El sistema debe garantizar que cuando haya una palabra que no quepa en el resto del reglón, debe comenzar una nueva línea. Observe que algunas líneas del alfabeto son diagonales, deben proponer la representación mediante coordenadas para cada letra en un espacio de 5×5 píxeles (el grupo debe proponer una representación de la coma y el punto, estos deben ser parecidos a la representación tradicional). No hay puntos y aparte en la representación.

   Las letras usarán el algoritmo de Bresenham y una representación similar a la sugerida en la Figura 1. El sistema debe identificar el siguiente espacio y las coordenadas.

Figura 2: Alfabeto en formato vectorial.

2. **Introducción de firma:** Al final del texto el grupo debe agregar una firma que incluya al menos un círculo y por lo menos cinco líneas. Esta firma debe ser personalizada para cada grupo y lo debe colocar al final del texto como una letra más de 5×5.

El sistema debe cumplir las siguientes características:

1. El texto debe ser almacenado en memoria y no puede tener menos de 100 palabras. Sugerencia: use algún poema o alguna estrofa de una canción.

2. La imagen de salida es de resolución de 250×250 deben ser almacenada en memoria. La misma es binarizada (fondo blanco con bit en uno y el texto en negro con un bit en cero). Cada letra es 5×5 con un espacio de 1 píxel a la izquierda y abajo de cada letra.

3. El sistema debe permitir al usuario cargar el código de ensamblador con la implementación de cada letra y la firma al final.

4. El sistema debe permitir la interacción del usuario observar el flujo del programa ensamblador.

Cada caracter (letra y firma) debe ser programática (sugerencia de lectura). Se deben seguir los siguientes requisitos generales de funcionalidad:

1. El diseño completo debe poder ser sintetizable para una tarjeta de desarrollo Terasic DE1-SoC-M TL2 (debe caber todo ahí, inclusive la imagen de salida).

2. La imagen de salida debe ser almacenada en memoria (se recomienda usar el bloque IP de la biblioteca de Quartus).

3. Se debe usar periféricos para iniciar la transformación del texto a imagen.

4. La salida del sistema se debe reflejar en n pines GPIO en *stream*. Las/los estudiantes deben escoger el número de pines en función del estudio del problema. Esta salida será escrita en un txt que será leído e interpretado por un software de alto nivel libre para dibujar el texto con la firma de forma gráfica.

5. El ISA debe ser eficiente y congruente, con criterios de diseño definidos. **Es importante pero no mandatorio hacer reuniones con el profesor para guía.**

2.1. Requisitos de Arquitectura ISA:

1. Debe diseñar un conjunto de instrucciones y arquitectura que permita solucionar el problema planteado, considerando detalles como:

   a) Modos de direccionamiento.

   b) Tamaño y tipo de datos.

   c) Tipo y sintaxis de las instrucciones.

   d) Registros disponibles y sus nombres.

   e) Codificación y descripción funcional de las instrucciones

   Tome en cuenta que estos detalles deben ser justificados desde el punto de vista de diseño (complejidad, costo, área, recursos disponibles). Las/los estudiantes deben realizar un análisis en un software de alto nivel para encontrar los valores idóneos.

2. Las instrucciones a desarrollar son libres así como el tipo de datos. Aunque no hay un límite en cuanto la cantidad de instrucciones, es importante que provea al menos instrucciones para control de flujo, operaciones aritméticas-lógicas, acceso a memoria.

3. El ISA debe ser personalizado y realizado por los estudiantes, no se aceptarán ISAs ya diseñados (e.g., ARM, x86, RISC-V, otros). Debe justificar cada característica del mismo.

4. Los productos finales de esta etapa son los siguientes:

   a) *Instruction reference sheet* o *green card*.

   b) Scripts usados para justificar las características del ISA.

2.2. Requisitos de Microarquitectura

1. La implementación diseñada debe ser correcta respecto a las reglas definidas por la arquitectura, esto quiere decir que el procesador debe ser capaz de ejecutar todas las instrucciones definidas y su especificación respecto a errores y excepciones.

2. El procesador diseñado debe emplear pipelining. Tenga en cuenta las implicaciones respecto a riesgos de dicha técnica, el uso de registros y unidades de ejecución. **No se revisará si no tiene pipeline.**

3. Debe ser implementado usando SystemVerilog.

4. No se permite realizar módulos especializados de hardware, como los de aritmética de punto fijo. Es un curso de Arquitectura de Computadores, no de Diseño de Sistemas Digitales.

5. El procesador debe tener capacidad de segmentación de memoria en datos e instrucciones además debe ser capaz de acceder los dispositivos de entrada y salida del sistema (GPIO, volcado de memoria, switches, etc).

6. Cada unidad funcional del sistema debe ser debidamente probada en simulación, para verificar su funcionamiento correcto (unit tests). Además debe incluir pruebas de integración y sistema. Se le solicita un plan de pruebas donde especifique los objetivos y descripción de las pruebas junto con sus resultados.

7. Los productos finales de esta etapa son:

   a) El código fuente (SystemVerilog) y el bitstream para programar la tarjeta de desarrollo mediante simulación.

   b) Un diagrama de bloques de la microarquitectura y descripción de las interacciones entre ellos.

   c) Simulaciones de las pruebas unitarias y de integración.

   d) Reporte de consumo de recursos del FPGA para el modelo.

Requisitos de Software:

1. Crear una aplicación (software) empleando la arquitectura diseñada, con el fin de implementar las funcionalidades solicitadas para este proyecto.

2. Debe realizar un programa compilador que permita traducir las instrucciones del ISA a binario, con la finalidad de ejecutarlo en el procesador. No es necesario que realice análisis léxico, sintáctico y semántico (este curso no es de Compiladores).

3. Debe desarrollar un programa(s) en alto nivel con la siguiente funcionalidad:

   a) Leer el `.txt` generado por el sistema desarrollado en Quartus y representar la imagen.

4. Los productos finales de esta etapa son:

   a) Programa compilador internamente documentado.

   b) Programa de alto nivel para representación de imagen.

   c) Código ensamblador para la arquitectura diseñada.

El proceso de diseño debe incluir propuestas y comparación de viabilidad de cada una de las tres etapas las mismas.

3. Evaluación y entregables

La defensa será el mismo día de la entrega y todos los archivos (incluyendo código fuente) serán entregados a las 11:59 pm ese mismo día (realícenlo progresivamente y no lo dejen para el final). **Si algo no queda claro o no sabe, no lo asuma, pregúntele al profesor.** Como recomendación se presenta el cronograma de trabajo de la Tabla 1.

Tabla 1: Cronograma sugerido para el proyecto

Semana	Actividades sugeridas
1	Estudio del problema y modelado del programa en alto nivel.
2	Análisis y generación de documentación y scripts para justificar el ISA. En esta parte se va a ir definiendo el *Green Card*.
3	Desarrollo de microarquitectura y pruebas unitarias sobre la misma. Desarrollo del compilador. Se pueden ir desarrollando los diagramas del sistema.
4	Desarrollo final de la microarquitectura. Validación de la arquitectura usando pruebas de integración (arquitectura sobre microarquitectura).
5	Validaciones de integración finales en las tres etapas.

La evaluación del proyecto se da bajos los siguientes rubros contra rúbrica correspondiente:

- Presentación proyecto 100 % funcional (75 %): La defensa se realizará de la siguiente manera: Debido a la situación actual (COVID-19) no se puede tener acceso a hardware u otros instrumentos y medios que requieran presencia y contacto físico tanto entre el profesor como l@s estudiantes. Por esta razón, para la defensa se debe presentar lo siguiente en un espacio de una hora:

    1. Todo el diseño debe ser sintetizable en una tarjeta: **Terasic DE1-SoC-M TL2**. Es decir, debe llegar hasta la generación de un **.sof**. Se garantiza que la tarjeta tenga suficientes recursos para almacenar y ejecutar el diseño según el reporte.
    2. Debe reservar los espacios de memoria para la imagen de salida.
    3. Mediante ModelSim debe crear un *testbench* de los mismos archivos de SystemVerilog que se usaron para sintetizar. Con este *testbench* debe escribir un archivo con el `.txt` binarizado.
    4. Debe generar un script de alto nivel para representar de la imagen de salida generada por el sistema.

    Los entregables adicionales que se revisarán durante la defensa son los siguientes:

    1. Arquitectura:

   a) *Instruction reference sheet* o *green sheet*.
   b) Scripts usados para justificar las características del ISA. **No cuenta como entregable el mismo script proveído por el profesor en este enunciado.**
2. Microarquitectura:
   a) Diagrama de bloques de la microarquitectura.
   b) Reporte de consumo de recursos del FPGA.
3. Software:
   a) Compilador usado.
   b) Código ensamblador para la arquitectura diseñada.
   c) Programa de alto nivel para representar de la imagen de salida generada por el sistema.

- Documentación de diseño (25 %): Este documento se encuentra directamente ligado con el atributo AP. La documentación del diseño deberá contener las siguientes secciones:

  1. Listado de requerimientos del sistema: Cada estudiante deberá determinar los requerimientos de ingeniería del problema planteado, considerando partes involucradas, estado del arte, estándares, normas, entre otros.
  2. Elaboración de opciones de solución al problema: Para el problema planteado deberán documentarse al menos dos opciones de solución. Cada solución deberá ser acompañada de algún tipo de diagrama. **Estas opciones de solución no deben ser fácilmente descartables y deben llevar un análisis objetivo con base en criterios técnicos o teóricos.** Al ser un curso de Arquitectura de Computadores, las opciones deben ser esencialmente basadas en el ISA.
  3. Comparación de opciones de solución: Se deberán comparar explícitamente las opciones de solución, de acuerdo con los requerimientos y otros aspectos aplicables de salud, seguridad, ambientales, económicos, culturales, sociales y de estándares.
  4. Selección de la propuesta final: Se deberá evaluar de forma objetiva, válida y precisa las soluciones planteadas al problema y escoger una solución final.
  5. Archivo tipo README donde especifiquen las herramientas que usaron junto con las instrucciones de uso. **Es un documento README.MD aparte.**

Se seguirán los siguientes lineamientos:

1. Los documentos serán sometidos a control de plagios para eliminar cualquier intento de plagio con trabajos de semestres anteriores, actual o copias textuales, tendrán nota de cero los datos detectados. Se prohíbe el uso de referencias hacia sitios no confiables.

2. No coloque código fuente en los documentos, quita espacio y aporta poco. Mejor explique el código, páselo a pseudocódigo o use un diagrama.