Diseño - Proyecto Final: Computador Básico Basado en Arquitectura ARM para Aplicación Específica

José Bernardo Barquero Bonilla
Carné: 2023150476
Correo: jos.barquero@estudiantec.cr
Escuela de Ingeniería en Computadores
Instituto Tecnológico de Costa Rica

Jose Eduardo Campos Salazar
Carné: 2023135620
Correo: j.campos@estudiantec.cr
Escuela de Ingeniería en Computadores
Instituto Tecnológico de Costa Rica

Jimmy Feng Feng
Carné: 2023060374
Correo: jifeng@estudiantec.cr
Escuela de Ingeniería en Computadores
Instituto Tecnológico de Costa Rica

Alexander Montero Vargas
Carné: 20233166058
Correo: ale_montero@estudiantec.cr
Escuela de Ingeniería en Computadores
Instituto Tecnológico de Costa Rica

***Resumen*—Aquí va el abstract**
***Palabras clave*—Palabras, clave, del, informe, aquí**

I. Introducción

Introduccion

II. Investigación para el Desarrollo del Proyecto

1. Resumen sobre Arquitectura ARMv4

La arquitectura **ARMv4** es una versión de 32 bits de la familia ARM (Advanced RISC Machine), ampliamente utilizada en sistemas embebidos por su eficiencia energética y buen rendimiento [1].

Aspectos clave:

- **Tipos de Instrucciones:**
  - *Data Processing:* Instrucciones que realizan operaciones aritméticas y lógicas [3]. Ejemplo: `ADD R1, R2, R3`.
  - *Load-Store:* Mueven datos entre registros y memoria [3]. Ejemplo: `LDR R0, [R1]`.
  - *Branching:* Instrucciones que alteran el flujo de ejecución del programa [3]. Ejemplo: `B loop_start`.
- **Codificación:** Instrucciones de 32 bits con campos definidos para condición, operación, operandos y desplazamientos [1].
- **Registros:**
  - R0–R12: Registros generales para el programador [3].
  - R13: Stack Pointer (SP) [3].
  - R14: Link Register (LR) [3].
  - R15: Program Counter (PC) [3].
  - *CPSR*: Registra el estado actual del procesador, incluyendo las banderas de condición [3].
  - *SPSR*: Guarda el estado previo del procesador durante interrupciones [3].

Ejemplo de instrucciones:

```
ADD R0, R1, R2     ; R0 = R1 + R2
MOV R3, #0x5       ; R3 = 5
LDR R4, [R5]       ; Cargar en R4 desde dirección
STR R6, [R7]       ; Guardar R6 en dirección R7
```

2. Herramientas para Simulación y Traducción

Simuladores:

- **QEMU:** Emulador general para arquitecturas ARM.
- **DS-5:** IDE con simulador y depurador de ARM.
- **Modelos ARMv4:** Para pruebas sin hardware físico.

Traducción a Lenguaje Máquina:

- **GAS (GNU Assembler):** Ensamblador de GNU usado con GCC.
- **Keil uVision:** IDE con ensamblador y depurador integrado.
- **ARM Compiler:** Herramientas optimizadas de compilación para ARM.

Flujo de Trabajo:

1) Escribir el código en ensamblador.
2) Compilar con `arm-none-eabi-gcc` o similar.
3) Simular o ejecutar con QEMU u otra plataforma.

3. Protocolos y Controladores Utilizados

PS/2 (Teclado):

- Comunicación serie sincrónica (Clock + Data).
- Señales capturadas por el controlador PS/2 en la FPGA.

Diagrama de tiempo (PS/2):

```
Clock    : |--|--|--|--|--|--|--|--|
Data     : |  |  |  |  |  |  |  |  | (datos de tecla)
```

UART (Serial Asíncrono):

- Comunicación serial usando TX/RX.
- Bits de inicio, datos (8 bits), y parada.

Diagrama UART:

```
Start Bit : |
Data Bits : | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 |
Stop Bit  : |
```

SPI (Serial Peripheral Interface):

- Alta velocidad, usa SCLK, MOSI, MISO y SS.

I2C (Inter-Integrated Circuit):

- Comunicación a dos cables (SDA, SCL), ideal para sensores.

Diagrama de tiempo (I2C):

```
SDA : |--|--|--|--|--|--|
SCL : |  |  |  |  |  |  |
```

Diagrama de Estados del Controlador PS/2:

```
[Inicio]
|
v
[Esperando datos]
|
v
[Leer datos de PS/2]
|
v
[Procesar datos]
|
v
[Enviar al CPU]
|
v
[Esperando siguiente señal]
```

III. Requerimientos del sistema

A. Requerimientos de hardware

1) **CPU con datapath compatible con ARMv4:**
   El sistema debe contar con una unidad central de procesamiento (CPU) basada en la microarquitectura ARMv4, que sea capaz de procesar instrucciones del lenguaje de máquina ARMv4. Esto garantiza la compatibilidad con programas previamente desarrollados y asegurará que se pueda utilizar en proyectos colaborativos con otros ingenieros.
   El utilizar una arquitectura estándar ARMv4 permite la reutilización de componentes y la integración con sistemas existentes, lo que contribuye a la reducción de desechos electrónicos.
2) **Módulo de RAM para almacenamiento de datos en tiempo de ejecución:**
   El sistema debe contar con suficiente memoria RAM para ejecutar programas dentro de los límites de la arquitectura ARMv4, sin desperdiciar recursos. Esto asegura que el sistema sea eficiente en términos de uso de memoria, evitando el sobrecosto y el derroche de recursos.Se debe considerar el uso de tecnología de memoria eficiente para minimizar el impacto ambiental y el consumo de energía.
3) **Controladores para dispositivos I/O:**
   El sistema debe ser capaz de interactuar con al menos dos tipos de dispositivos de entrada/salida, como teclado PS2, mouse, conexiónes UART. La elección de estos dispositivos debe basarse en la disponibilidad de hardware accesible, evitando el consumo innecesario de materiales nuevos y favoreciendo el uso de componentes ya disponibles y funcionales.
   Seleccionar dispositivos con interfaces más antiguas para la reutilización de hardware existente reducirá la huella de carbono asociada con la fabricación de nuevos productos electrónicos.
4) **Salida de Video VGA:**
   El sistema debe ser capaz de visualizar la salida en un monitor VGA, un estándar de visualización que sigue siendo ampliamente utilizado y que es compatible con la mayoría de las interfaces de hardware más antiguas.
   La reutilización de monitores VGA existentes reduce la necesidad de producir nuevos dispositivos de visualización, lo que disminuye el desperdicio electrónico y promueve la sostenibilidad.

B. Requerimientos de software

1) **Interpretador de lenguaje de máquina para ARMv4:**
   El sistema debe incluir un interpretador de lenguaje de máquina para decodificar y ejecutar el set de instrucciones ARMv4. Esto garantizará que la CPU pueda ejecutar correctamente los programas escritos en lenguaje ensamblador ARMv4.
   Se debe asegurar que el interpretador sea eficiente para minimizar el uso de ciclos de reloj, lo que reducirá el consumo de energía y mejorará el rendimiento general.
2) **Aplicación específica en software usando lenguaje ensamblador ARMv4:**
   El sistema debe ejecutar una aplicación específica, desarrollada en lenguaje ensamblador compatible con ARMv4, que aproveche al máximo el poder de procesamiento de la CPU y minimice el uso innecesario de memoria y ciclos de reloj.
   El software debe ser eficiente en cuanto al uso del CPU y la memoria, evitando sobrecargar el sistema y optimizando los recursos.
3) **Interfaz intuitiva para facilitar el uso y la curva de aprendizaje:**
   La aplicación debe ser intuitiva para el usuario, de modo que facilite su uso sin una curva de aprendizaje pronunciada. En particular, debe ser diseñada con accesibilidad

en mente, para permitir la adaptación a personas con discapacidades físicas o motoras.
La interfaz debe ser adaptable, internacionalizable, y fácil de comprender para usuarios con diferentes necesidades y antecedentes. Esto garantizará la inclusión y accesibilidad en el uso de la aplicación.

C. Consideraciones Generales

- **Salud y Seguridad Pública:**
  El diseño del sistema debe garantizar que las interacciones físicas con el hardware sean seguras para el usuario. Además, se debe cuidar que no haya riesgos eléctricos ni otros peligros relacionados con la interfaz de hardware.
- **Costo Total de la Vida:**
  A lo largo del ciclo de vida del sistema, desde la producción hasta la posible eliminación del hardware, se debe tener en cuenta no solo el costo inicial de los componentes, sino también el mantenimiento, la eficiencia energética, y los costos de reciclaje al final de la vida útil del hardware. Esto contribuirá a minimizar el impacto económico y ambiental del sistema.
- **Carbono Neto Cero:**
  El diseño debe ser lo más eficiente posible en términos de consumo de energía. A través de la selección de componentes de bajo consumo y la optimización del uso de ciclos de reloj, se puede reducir la huella de carbono asociada al uso de la CPU y otros dispositivos. Además, la reutilización de componentes y la integración de interfaces antiguas ayudará a minimizar el impacto ambiental de la fabricación de nuevos dispositivos.
- **Aspectos Culturales, Sociales y Ambientales:**
  El sistema debe ser diseñado de forma que sea accesible y útil para una amplia variedad de usuarios, respetando las diferencias culturales y sociales. La inclusión de opciones para personas con discapacidades motoras o físicas también es un aspecto importante para fomentar la equidad en el acceso a la tecnología.

IV. Alternativas

A. Alternativa 1: Uso de hardware existente (FPGA DE10-Standard)

Descripción de la Alternativa: Esta alternativa propone usar una FPGA DE10-Standard como plataforma base para implementar la solución. La FPGA es programable mediante el uso de lenguajes de descripción de hardware (HDL) como SystemVerilog, lo que permite adaptar el diseño a la aplicación específica. Además, la FPGA ya cuenta con puertos e interfaces de Entrada/Salida (I/O), lo que facilita la interacción con dispositivos como el teclado PS/2, botones de la FPGA y salida de video VGA.

Ventajas:

- **Facilidad de implementación:** La FPGA ya cuenta con muchos bloques lógicos y módulos predefinidos, lo que facilita la implementación de la CPU ARMv4 y otros componentes del sistema (RAM, ROM, ALU, FLAGS).
- **Adaptabilidad:** La FPGA es programable, lo que significa que puede adaptarse fácilmente a nuevas aplicaciones o modificaciones sin necesidad de rediseñar todo el sistema.
- **Optimización de recursos:** El uso de una FPGA permite un diseño modular, optimizando el uso de bloques lógicos y recursos disponibles. Esto es esencial para maximizar el rendimiento sin desperdiciar recursos, lo que contribuye al costo total de la vida más bajo y una menor huella de carbono.
- **Reutilización de hardware:** Al utilizar una plataforma existente como la FPGA DE10-Standard, se reutilizan dispositivos y recursos previos, reduciendo el consumo de materiales nuevos y el impacto ambiental.
- **Facilita la verificación:** La FPGA permite realizar simulaciones y pruebas más fácilmente antes de la implementación final, asegurando que el sistema funcione correctamente.

Desventajas:

- **Limitaciones de recursos:** Aunque la FPGA DE10-Standard es potente, tiene limitaciones en términos de capacidad de bloques lógicos y memoria en comparación con un sistema más específico.
- **Costo inicial más alto:** La compra de una FPGA puede ser más cara en comparación con un chip dedicado, lo que puede no ser óptimo para proyectos con presupuesto muy limitado.
- **Consumo de energía:** Las FPGAs pueden consumir más energía en comparación con un chip diseñado específicamente para una tarea, ya que tienen una mayor cantidad de transistores y lógica programable.
- **Complejidad del diseño:** Programar una FPGA puede ser más complejo, lo que puede incrementar el tiempo de desarrollo y la posibilidad de errores.

Consideraciones Adicionales:

- **Costo Total de la Vida:** La reutilización de la FPGA reduce el costo total de la vida, aunque el costo inicial puede ser elevado.
- **Carbono Neto Cero:** El diseño modular en FPGA permite optimizar el uso de recursos, lo que ayuda a minimizar el consumo energético.
- **Recursos:** La FPGA DE10-Standard ofrece una variedad de bloques lógicos y recursos que permiten una eficiente utilización de los recursos.
- **Salud y Seguridad Pública:** Un diseño robusto en FPGA permite simular y probar el diseño antes de implementarlo, mejorando la seguridad del sistema.

B. Alternativa 2: Usar hardware a medida (ASIC o procesador ARMv4 preexistente)

Descripción de la Alternativa: Esta alternativa sugiere diseñar un procesador personalizado (ASIC) o utilizar un procesador ARMv4 preexistente y conectar manualmente todos los dispositivos de I/O, como teclado, botones y salida de video. Este enfoque requiere una mayor especialización en el diseño de hardware.

Ventajas:

- **Eficiencia en el uso de recursos:** Un procesador diseñado específicamente para la tarea puede ser más eficiente energéticamente.
- **Posiblemente más económico a gran escala:** Un ASIC puede ser más barato que una FPGA si se produce en masa.
- **Menor consumo de energía:** Un chip dedicado suele ser más eficiente que una FPGA.
- **Mayor control sobre el diseño:** Permite un control total sobre la arquitectura del sistema.

Desventajas:

- **Mayor tiempo de desarrollo:** Diseñar o integrar hardware personalizado toma más tiempo.
- **Costo de desarrollo alto:** El diseño de un ASIC o integración de un procesador puede ser costoso.
- **Flexibilidad limitada:** Una vez fabricado, el chip no puede modificarse.
- **Dificultades en la verificación y pruebas:** Verificar un ASIC es más complejo que una FPGA.

Consideraciones Adicionales:

- **Costo Total de la Vida:** En producciones masivas, el costo por unidad puede ser menor. Para usos únicos, es más alto.
- **Carbono Neto Cero:** Un diseño optimizado reduce la huella de carbono frente a una FPGA.
- **Recursos:** El ASIC optimiza recursos al estar hecho a medida para la aplicación.
- **Salud y Seguridad Pública:** Un diseño dedicado puede enfocarse completamente en la robustez y seguridad.

V. Valoración de Opciones de Solución

En esta sección, realizamos una valoración comparativa entre las dos alternativas de solución (uso de FPGA DE10-Standard vs. uso de hardware a medida), teniendo en cuenta los aspectos clave del proyecto, como la salud y seguridad pública, el costo total de la vida, el carbono neto cero, y consideraciones sobre recursos, culturales, sociales y ambientales.

A. Alternativa 1: Uso de hardware existente (FPGA DE10-Standard)

1) **Salud y Seguridad Pública:** La FPGA DE10-Standard es un sistema probado y robusto, lo que minimiza el riesgo de fallos imprevistos durante el uso. El diseño modular y la capacidad de realizar simulaciones previas en FPGA garantizan que el sistema sea seguro antes de implementarlo en el hardware real. Además, los riesgos asociados con la manipulación de hardware son bajos, ya que el sistema es ampliamente utilizado y tiene un buen historial en términos de fiabilidad.
2) **Costo Total de la Vida:** El uso de una FPGA existente es una opción de bajo costo a corto plazo debido a la reutilización de hardware. Aunque la inversión inicial puede ser más alta, permite desarrollar y probar rápidamente sin diseñar un chip desde cero, lo que reduce el costo de desarrollo y el tiempo de implementación. A largo plazo, este costo se amortiza mediante la reutilización en otros proyectos.
3) **Carbono Neto Cero:** Aunque las FPGAs pueden consumir más energía que los ASICs, un diseño eficiente puede minimizar el uso de recursos y reducir la disipación de calor. Además, el uso de hardware existente evita la fabricación de nuevos componentes, lo que disminuye la huella de carbono.
4) **Recursos:** Se aprovechan los bloques lógicos y módulos de I/O preexistentes, maximizando la utilización de recursos y evitando el desperdicio de materiales. Sin embargo, las limitaciones de capacidad pueden requerir optimización para proyectos de mayor escala.
5) **Consideraciones Culturales, Sociales y Ambientales:** Reutilizar hardware tiene un impacto ambiental positivo al evitar la producción de nuevos componentes. La flexibilidad de la FPGA permite adaptar el sistema para necesidades específicas, como la inclusión de personas con discapacidades mediante interfaces adecuadas (teclado PS/2, botones, etc.).

B. Alternativa 2: Usar hardware a medida (procesador ARMv4 personalizado o preexistente)

1) **Salud y Seguridad Pública:** El diseño a medida ofrece control total sobre la seguridad del sistema, permitiendo incorporar medidas específicas. Sin embargo, la integración manual de I/O aumenta la complejidad, lo que puede elevar el riesgo de errores si no se realizan pruebas exhaustivas.
2) **Costo Total de la Vida:** En producción masiva, el ASIC puede ser más económico por unidad, pero en proyectos pequeños, el costo de diseño y fabricación es considerablemente más alto. Además, la conexión manual de dispositivos I/O puede incrementar costos y tiempo.
3) **Carbono Neto Cero:** El ASIC optimizado puede tener un consumo energético menor que una FPGA. Sin embargo, la fabricación de un nuevo chip genera mayor impacto ambiental debido al uso intensivo de materiales y energía.
4) **Recursos:** Permite un uso muy eficiente de recursos al estar diseñado específicamente para la tarea. Sin embargo, al ser poco reutilizable, puede resultar en desperdicio si no se utiliza en otros proyectos.
5) **Consideraciones Culturales, Sociales y Ambientales:** Un diseño personalizado puede facilitar la inclusión de características específicas. No obstante, su impacto ambiental es mayor por requerir nuevos procesos de fabricación y puede ser socialmente menos accesible por su alto costo y menor flexibilidad.

VI. Selección de la Propuesta Final

Propuesta seleccionada: Alternativa 1: Uso de hardware existente (FPGA DE10-Standard)

Justificación de la Selección:

1) **Facilidad de Implementación y Flexibilidad:** La FPGA DE10-Standard ofrece una solución altamente flexible y programable, facilitando la implementación de la arquitectura ARMv4 y sus módulos asociados (CPU, RAM, ROM, ALU, FLAGS) mediante HDL (SystemVerilog). A diferencia del hardware a medida, donde se requiere diseñar un chip o conectar dispositivos I/O manualmente, la FPGA proporciona una plataforma probada que reduce riesgos y acelera el desarrollo, algo crucial en contextos educativos o con plazos ajustados.
2) **Reutilización de Hardware y Reducción de Costos Iniciales:** El uso de la FPGA existente optimiza la reutilización de recursos y disminuye la necesidad de adquirir nuevos componentes. Esto representa un ahorro significativo frente a la alternativa de desarrollar un ASIC, cuyo diseño y fabricación requieren una alta inversión. Además, la FPGA puede ser reutilizada en futuros proyectos, aumentando la rentabilidad del hardware.
3) **Reducción del Impacto Ambiental:** Al evitar la fabricación de nuevos componentes, el uso de la FPGA reduce la huella de carbono y el desperdicio electrónico. Este enfoque está alineado con los principios de sostenibilidad y carbono neto cero, al disminuir el consumo de recursos materiales y energéticos.
4) **Optimización de Recursos y Diseño Modular:** La FPGA permite aprovechar eficientemente los bloques lógicos, la memoria y los módulos de I/O disponibles. Su diseño modular facilita la integración y adaptación de componentes sin complicar la arquitectura, lo que contribuye a una mejor utilización de los recursos y a un menor costo total de la vida.
5) **Facilidad para Pruebas y Simulaciones:** El entorno de desarrollo de la FPGA permite realizar simulaciones y pruebas detalladas de cada componente antes de la implementación física, mejorando la confiabilidad del sistema y reduciendo los errores en etapas avanzadas del proyecto.
6) **Consideraciones Sociales y Culturales:** La solución basada en FPGA permite la integración de interfaces accesibles como teclado PS/2 y botones físicos, que pueden adaptarse a personas con discapacidades físicas o motoras. Esto hace que el sistema sea más inclusivo y socialmente equitativo.

Conclusión: La **Alternativa 1: Uso de hardware existente (FPGA DE10-Standard)** ha sido seleccionada debido a su facilidad de implementación, reutilización de hardware, reducción de costos y su alineación con principios de sostenibilidad. Además, su flexibilidad para pruebas y modificaciones garantiza un desarrollo más robusto, eficiente e inclusivo.

VII. Diseño de la Alternativa Seleccionada

Diagrama de Bloques del Computador

El sistema diseñado utiliza una FPGA DE10-Standard que emula una CPU ARMv4 para ejecutar la aplicación. Los componentes principales incluyen el procesador, memoria (RAM/ROM), dispositivos de entrada/salida (teclado PS/2, botones, VGA) y módulos de control.

- **CPU:** Implementa el conjunto de instrucciones ARMv4.
  - Registros: Incluye el Program Counter y registros de propósito general.
  - ALU: Unidad Aritmético Lógica para operaciones aritméticas y lógicas.
  - Unidad de Control (CU): Coordina la ejecución de instrucciones.
- **Memoria:**
  - ROM: Almacena el código binario del programa.
  - RAM: Guarda datos en tiempo de ejecución.
- **Dispositivos de Entrada/Salida:**
  - Teclado PS/2: Entrada principal del usuario.
  - Botones: Entradas físicas en la FPGA.
  - Pantalla VGA: Salida visual del sistema.
- **Módulos de Control:**
  - VGA Controller: Gestiona la señal de video.
  - Interfaz de Botones: Lee y filtra entradas de botones.
  - Debounce: Elimina rebotes en las señales de entrada.

Diagrama de Bloques (simplificado):

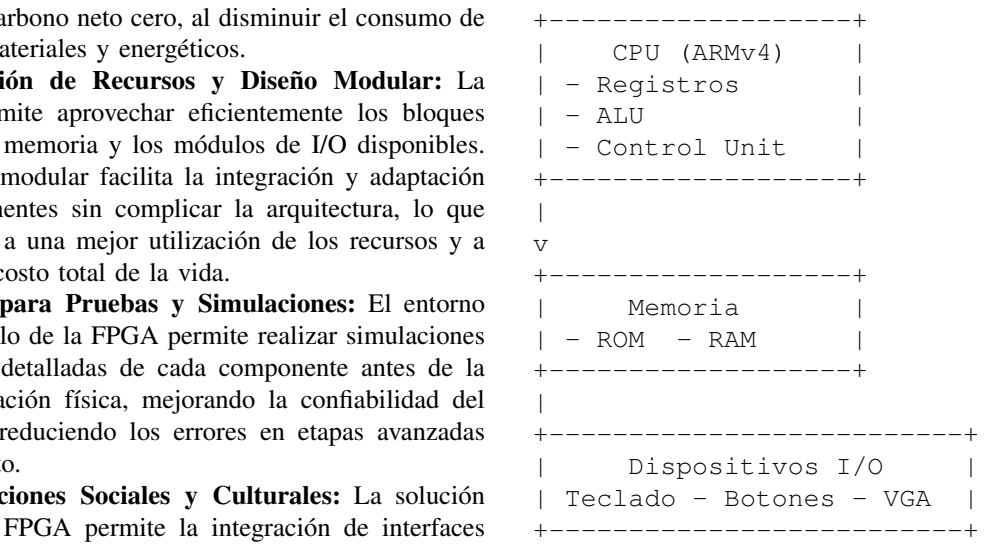
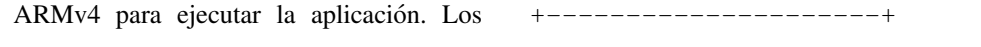


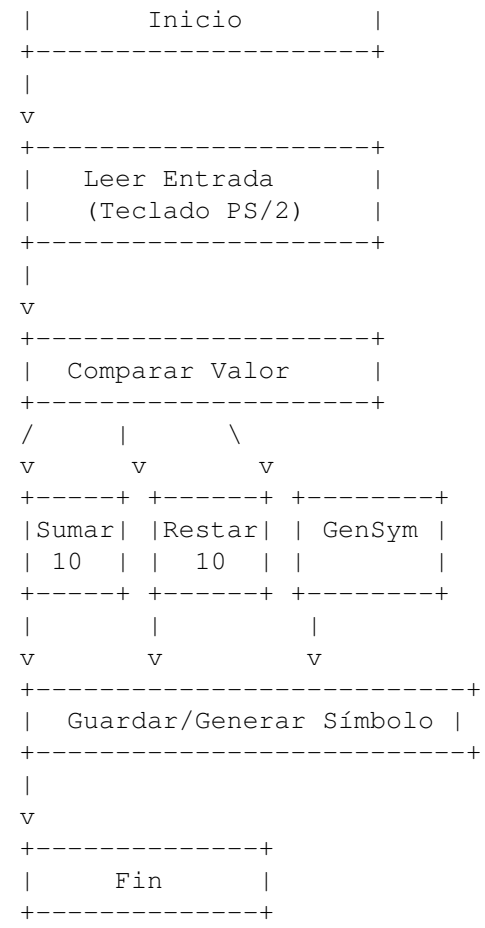
Diagrama de Diseño de Software

El software ejecutado en la CPU ARMv4 es un conjunto de instrucciones en lenguaje ensamblador, responsable de controlar la lógica del sistema.

- **Ciclo Principal:** Controla la ejecución continua del programa.
- **Sumar10:** Suma 1 si la entrada es 0x75.
- **Restar10:** Resta 1 si la entrada es 0x72.
- **GenSym:** Genera un símbolo aleatorio al recibir 0x5A.
- **SaveSym1, ResetSym:** Verifica y almacena símbolos válidos.

Diagrama de Flujo del Algoritmo:





Descripción del Algoritmo en ARMv4

- **Inicialización:** Se configuran registros R0–R3 con valores iniciales.
- **Condiciones:**
  - Si R1 = 0x75 ⇒ incrementar valor en RAM.
  - Si R1 = 0x72 ⇒ decrementar valor en RAM.
  - Si R1 = 0x5A ⇒ generar y guardar símbolo aleatorio si cumple condiciones.
- **Bucle Principal:** Se ejecuta continuamente a la espera de nuevas entradas.

Módulos en SystemVerilog

- **Módulo CPU:** Implementa el procesador ARMv4. Contiene instancias de Mux, ALU, Unidad de Control y Registro.
- **Módulo ALU:** Ejecuta operaciones aritmético-lógicas (suma, resta, comparación).
- **Módulo de Memoria:**
  - **ROM:** Código binario del programa.
  - **RAM:** Variables, símbolos generados y resultados.
- **Módulos de Entrada/Salida:**
  - Teclado PS/2: Entrada principal del usuario.
  - VGA Controller: Genera señal de video para mostrar el estado del juego.
  - Botones: Entradas adicionales para acciones rápidas o reinicio.

VIII. Validación del Diseño

La validación del diseño final se realizará conforme a los requisitos establecidos en el proyecto, incluyendo consideraciones de salud y seguridad pública, costo total de la vida, carbono neto cero, y aspectos relacionados con los recursos, lo social, lo cultural y lo ambiental.

1. Validación de Requisitos Técnicos y Funcionales

Requisito de Hardware: El sistema debe contar con una FPGA DE10-Standard programada con un procesador ARMv4 capaz de ejecutar instrucciones y controlar dispositivos de entrada y salida (teclado PS/2, botones de FPGA, pantalla VGA).

Validación:

- Pruebas funcionales para asegurar la correcta emulación del procesador ARMv4.
- Simulaciones de interfaces I/O (teclado, botones y VGA) para verificar la comunicación efectiva.

Requisito de Software: El software debe ejecutarse de forma eficiente y precisa sobre el procesador ARMv4.

Validación:

- Pruebas del ciclo principal y funciones clave (Loop, Sumar10, Restar10, GenSym, etc.).
- Pruebas de estrés en memoria y registros para asegurar su correcta gestión.

2. Validación de la Salud y Seguridad Pública

Requisito de Seguridad Eléctrica: El sistema debe ser seguro para el usuario.

Validación:

- Uso de componentes certificados y análisis de posibles riesgos eléctricos.
- Pruebas térmicas y de estabilidad bajo diferentes condiciones operativas.

Accesibilidad del Sistema: Debe ser accesible a personas con discapacidades físicas o motoras.

Validación:

- Evaluación de usabilidad del teclado PS/2 y botones en diversos perfiles de usuario.
- Validación de legibilidad y contraste de la interfaz VGA.

3. Validación del Costo Total de la Vida

Requisito de Costo Inicial y Mantenimiento: El sistema debe ser rentable a largo plazo.

Validación:

- Análisis comparativo de costos frente a alternativas ASIC.
- Estimación de reutilización futura del hardware en otros proyectos.

4. Validación del Carbono Neto Cero

Requisito de Eficiencia Energética: El consumo debe ser optimizado.

Validación:

- Medición del consumo energético bajo diferentes cargas.

- Optimización del software para reducir ciclos de reloj innecesarios.

Reutilización de Hardware: Disminución del impacto ambiental.

Validación:

- Plan de reutilización de la FPGA para extender su vida útil.
- Evaluación del ahorro en materiales por evitar fabricación de nuevos chips.

5. Validación de la Optimización de Recursos

Requisito de Uso Eficiente de Recursos: Se debe aprovechar al máximo la FPGA.

Validación:

- Análisis del uso de bloques lógicos y aprovechamiento de RAM/ROM.
- Verificación de que no haya recursos subutilizados o desperdiciados.

6. Validación de Aspectos Culturales, Sociales y Ambientales

Accesibilidad e Inclusividad: El sistema debe ser inclusivo.

Validación:

- Pruebas con usuarios con discapacidades físicas/motoras.
- Evaluación de la interfaz visual con criterios de accesibilidad.

Impacto Ambiental: Debe minimizarse el impacto ecológico.

Validación:

- Análisis del ciclo de vida del sistema (producción, uso, disposición).
- Verificación del potencial de reciclaje de la FPGA y componentes asociados.

REFERENCIAS

[1] S. Furber, *ARM System-on-Chip Architecture*, 2nd. Addison-Wesley, 2000, ISBN: 978-0201675191.

[2] D. Harris y S. Harris, *Digital Design and Computer Architecture*, 2ª ed. Morgan Kaufmann, 2015. dirección: https://www.elsevier.com/books/digital-design-and-computer-architecture/harris/978-0-12-800056-4.

[3] R. Meier y D. M. Harris. "ARMv4 ISA Instructions." Milwaukee School of Engineering. dirección: https://faculty-web.msoe.edu/meier/ce1921/slidesets/isaarm-instructions.pdf.

[4] W. Stallings, *Computer Organization and Architecture: Designing for Performance*, 9th. Pearson Education, 2013, ISBN: 978-0132936330.