Índice

1. Introducción

1.1. Descripción General

Descripción por arriba del trabajo. por ejemplo la industria dispone de herramientas que nosotros usamos para un fin determinado.

1.2. Objetivos

1.2.1. Objetivo General

Implementar un system on chip OpenSource con un microprocesador embebido Soft-core que soporte un sistema operativo libre , con la finalidad de entregar un sitema integral FPGA-SoC-Sistema Operativo completamente funcional y bajo licencia GPL v2.

1.2.2. Objetivo Específico

- Seleccionar, analizar y determinar un microprocesador Sof-Core.
- Establecer un system on chip Open Source donde poder implementar un Soft-Core.
- Determinar sistemas operativo con licencia GPL v2 que tengan las prestaciones funcionales adecuadas.

1.3. Motivación

Existe un grupo de cores Sof-Core de código abierto que no están limitados por la tecnología. Los cores destacados de microprocesadores de 32 bits, son los procesadores SPARC LEON OpenRISC 1200 , y el core de LatticeMico32. Usar cores de codigo abierto, va unido a una serie de conceptos como:

- Flexibilidad. Si el codigo fuente está disponible, los desarrolladores pueden modificar el codigo de acuerdo a sus necesidades.Adémas, se produce un flujo constante de ideas que mejora la calidad del codigo.
- Fiabilidad y seguridad. Con muchos programadores a la vez escrutando el mismo trabajo, los errores se detectan y corrigen antes, por lo que el producto resultante es mas fiable y eficaz que el comercial.
- Rapidez de desarrollo. Las actualizaciones y ajustes se realizan a través de una comunicación constante vía Internet.
- Relación con el usuario. El programador se acerca mucho mas a las necesidades reales de su cliente, y puede crear un producto especifíco para él

Obtener un sistema integral de código abierto en donde se tiene código HDL, assembler y C disponible para adaptarse de acuerdo a los requerimientos del proyecto. Ademas de la de la capacidad de migrar de una plataforma a otra. Logrando menor dependencia entre el código fuente y la plataforma objetivo. La portabilidad del codigo abierto nos permite implementarlo sobre una ASICs (Application-specific integrated circuit) o con modificaciones menores en cualquier FPGA (Field Programmable Gate Array) de Xilinx, Altera, Lattice, etc. Estos tres de los más grandes proveedores de FPGA , Xilinx , Altera y Lattice , ofrecen sus propios micro core RISC de 32bits los dos mayores proveedores de dispositivos FPGA , Altera y Xilinx , proporcionan el micro core Nios y Microblaze, respectivamente. Son micro cores en donde el codigo fuente RTL no se encuera disponible y solo pueden ser implementados en sus respectivas FPGA.

1.4. Importancia del Problema

En el diseño del sistema embebido se usan diferentes procesos depende del tipo de sistema, el hardware disponible y la organización que desarrolle el sistema. Una de las actividades principales en un proceso de diseño de software es la elección del hardware y del sistema operativo que se efectúa antes del comienzo del software. Ante tal situación , se debe diseñar el software par considerar las restricciones impuestas por las capacidades del hardware. Los efectos que influyen dichas elecciones comprenden restricciones de de temporización sobre el sistema, limitación en la energía disponible, experiencia del equipo de desarrollo y limites en el costos del sistema entregable.

Se está explorando una linea donde se busca dar al diseñador del sistema embebido una solución flexible en la primera etapa de la elección de plataforma. Donde a través del análisis de diferentes plataformas de desarrollo OpenSource y privativas pueda elegir la mejor opción para el tipo de sistema a desarrollar y requerimientos de proceso.

Una vez que se ha elegido la plataforma de ejecucion para el sistema, se ha diseñado una arquitectura de proceso y se a determinado una políticas de planeación, es necesario comprobar que el sistema cumplirá sus con sus requerimientos.

1.5. Alcance del Estudio

Debido al plazo estipulado para el desarrollo del proyecto, el mismo involucra tres etapas:

- Especificación y Análisis de requerimientos.
- Implementación.
- Testing.

1.6. Modelo de Desarrollo

El modelo de desarrollo a utilizar es el Modelo en Espiral tipificado por Ian Sommerville [2] . El modelo en espiral de ingeniería de software, mostrado en la Ilustración 1, fue originalmente propuesto por Boehm en año 1988, en su artículo A Spiral Model of Software Development and Enhancement. Propuso un marco del proceso de software dirigido por el riesgo. Aquí, el proceso de software se representa como una es espiral, cada ciclo en la espiral representa una fase del proceso de software. Por ende el, ciclo más interno puede relacionarse con la factibilidad del sistema, el siguiente ciclo con la definición de requerimientos, el siguiente ciclo al diseño del sistema, y así sucesivamente. Cada ciclo del espiral se divide en 4 sectores:

- Establecimiento de objetivo Se definen objetivos específicos para dicha fase del proyecto. Se identifican restricciones en el preceso y el producto, y se traza un plan detallado de gestión. Se identifican los riesgos del proyecto. Dependiendo de estos riegos, se planean estrategias alternativas
- Validación y reducción del riesgo En cada uno de los riesgos identificados del proyecto, se realiza un análisis minucioso. Se dan acciones para reducir el riesgo.
- Desarrollo y validación Despues de una evaluacion del riesgo, se elige un modelo de desarrollo para el sistema.
- Planeción El proyecto se revisa y se toma una decisión sobre si hay que continuar con otro ciclo de la espiral. Si se opta por continuar, se trazan los planes para la siguente fase del proyecto.

Como característica principal de esta metodología es que posee una consideración explícita del riesgo. Informalmente, el riesgo significa sencillamente que algo puede ir mal. Los riegos originan problemas en el proyecto, como los de confección de agendas y excesos en los costos; por lo tanto, la disminución de riegos es una actividad sumamente importante en la gestión del proyecto. Un ciclo en la espiral comienza con la elaboración de objetivos, como el rendimiento y la funcionalidad. Entonces se enumeran formas alternativas de alcanzar estos objetivos y las restricciones impuestas en cada una de ellas. Cada alternativa se evalúa contra cada objetivo y se identifican las fuentes de riegos del proyecto. El siguiente paso es resolver estos riesgos mediante actividades de recopilación de información como la de detallar más el análisis, la construcción de prototipos y la simulación. Una vez que se han evaluado los riesgos se llevará a cabo cierto desarrollo, seguido de una actividad de planificación para la siguiente fase del proceso.

1.7. Metodología

voy a escribir que usamos un desarrollo experimental y de simulación.

2. FPGA y microprocesadores Soft-Core

2.1. FPGAs

problema q resuelven.papel dual que cumplen como prototipo y otro como objetivo final de ejecución

Entre las alternativas para diseñar e implementar un hardware específico encontramos ASICs (Application-specific integrated circuit), FPGAs (Fieldprogrammable gate array), CPLDs (Complex Programmable Logic Device) entre otros. El uso de ASICs posibilita desarrollos con producción a gran escala a bajo costo y es de masiva utilización en este tipo de aplicaciones. Los CPLD y las FPGA son circuitos de de alta densidad programables por el usuario en un tiempo reducido y sin la necesidad de verificación de sus componentes, tarea ya realizada por el fabricante al tratarse de un producto estándar. El procesamiento digital de señales , prototipado de ASICs , tratamiento de imágenes , reconocimiento de voz , glue logic son algunas de las aplicaciones de este tipo de dispositivos. Existen diferentes formas de llevar adelante el diseño e implementación de un sistema digital para FPGA, entre ellas tenemos la realización de un diseño esquemático , herramientas específicas (provistas por el fabricante) y la utilización de un lenguaje de descripción de hardware HDL (Hardware description language) entre los que se encuentran lenguajes como Verilog y VHDL, ambos de gran aceptación en los ambientes industrial y académico. Estos lenguajes proporcionan gran versatilidad para el desarrollo de hardware, permitiendo especificar, diseñar, simular y verificar sistemas digitales complejos, mediante el apoyo de un universo de herramientas EDA (Electronic Design Automation).

Actualmente las FPGA cuentan con una gran cantidad de recursos disponibles (Compuertas lógicas , Bloques de RAM) para implementar diseños digitales complejos. Las FPGA pueden ser usadas para implementar cualquier función lógica que un ASIC pueda realizar. Una de las grandes ventajas del uso de FPGA en la etapa de prototipado es su capacidad de reconfigurar el diseño parcial o totalmente para su actualización o corrección de errores con un costo relativamente bajo a diferencia del prototipado sobre ASICs. Durante la etapa de producción los ASIC resultan de muy bajo costo respecto de la producción de FPGA y esto se traduce en una gran ventaja para desarrollos que deben ser producidos a gran escala. Las arquitecturas reconfigurables combinan parte de la flexibilidad del software con la gran performace del hardware utilizando chips reconfigurables como FPGAs. Una opción de gran potencialidad son las arquitecturas reconfigurables run-time o dinámicas que se sostienen en DRL (Dynamically reconfigurable logic). Un ejemplo de esto es la familia Virtex de Xilinx, que es parcialmente reconfigurable en tiempo de ejecución, este método se conoce como run-time. Claramente este tipo de dispositivos puede ser usado como arquitecturas destino de codiseños hardware/software

(HW/SW) que proveen la flexibilidad de los procesadores software y la eficiencia y rendimiento de los coprocesadores hardware. El desarrollo de aplicaciones de software se ve limitado a los recursos disponibles en los microprocesadores comerciales. El software necesita del soporte de un procesador para su ejecución, así la elección de este elemento conlleva algunas dependencias respecto de las herramientas a utilizar , algunas de estas son : compiladores , ensambladores , depuradores y herramientas de simulación. Con ellas se logra trasladar el software a un entorno de ejecución adecuado dentro del procesador y depurarlo . El lenguaje de programación elegido debe permitir trabajar con el nivel de abstracción necesario para simplificar la tarea de desarrollo de software. El lenguaje C es uno de los más utilizados para desarrollar aplicaciones que requieran ,por ejemplo , de compilación cruzada (Cross-compilling) y se complementa con las herramientas libres de compilación y depuración como son GCC y GDB. Los nuevos proyectos a veces requieren nuevas características de los cores existentes. El proveedor del núcleo puede hacer estas modificaciones (solución comercial) con un incremento sustancial del coste del núcleo. Otra posibilidad (Solución Ad-hoc) es el uso de cores de código abierto con el fin de crear un núcleo de desarrollo adaptable. El enfoque de código abierto tiene varias ventajas: el núcleo posee un costo muy bajo e inclusive cero, los usuarios puede tener acceso al código fuente y hay un grupo de desarrolladores que proporcionan conocimientos para mantener y mejorar el núcleo. Sin embargo, también puede tener varias desventajas como la inestabilidad (el grupo de cambio o de desarrollo desaparece), desarrollo incompleto, deficiente , documentación pobre y una mala metodología de verificación. Los microprocesadores “softcore"(núcleo software

2.1.1. Topología

arquitectura de una fpga

2.1.2. Bloques Lógicos Configurables y Lookup Tables

2.1.3. Bloques de Entrada y Salida

2.1.4. Bloques de Memoria

2.2. Microprocesadores Soft-Core

2.2.1. IP-Core

itro definición y licencias

2.2.1.1. Tipos de IP-Core hard soft y intermedios

2.2.2. Soft-Core

intro, deferencias entre micros soft y hard. fabricantes.ventajas de uno sobre el otro. Ejemplo: En la industria existe un grupo ligeramente diferente de microprocesadores Soft-Core , apuntando principalmente al hardware reconfigurable . Tres de los más grandes proveedores de FPGA , Xilinx , Altera y Lattice , ofrecen sus propios núcleos de microprocesadores RISC de 32 bits. Los dos mayores proveedores de dispositivos FPGA , Altera y Xilinx , proporcionan la Nios y Microblaze núcleos , respectivamente. Ellos se consideran hard-core en los que la fuente RTL no se encuentra disponibles y sólo pueden ser implementadas en sus Tecnologías FPGA .

Existe un grupo de cores opensource que no están limitados por la tecnología, y son cores intrínsecamente soft . Los cores mas destacados en esta categoría son los microprocesadores de 32 bits SPARC LEON OpenRISC 1200 , y el núcleo LatticeMico32 de la empresa Lattice.

Para el desarrollo de una aplicación reconfigurable existe los micro cores de 32bist que ofrece los proveedores de las FPGA y los micro soft-core opensource disponibles de forma gratuita.

Sin embargo , cuando se trata de ser capaz de desarrollar y vender un producto a base de estos núcleos , hay consideraciones adicionales sobre la concesión de licencias de los diseños. Estas cuestiones relacionadas con licencias voluntad se discutirá en una sección posterior.

3. Benchmark

Este capitulo lo puse para darle una intro a el estudio de los test que voy a poner en el estudio de los micor soft-core

3.1. Introducción

explico un poco para q los uso y en que se usan

En informática , un punto de referencia es el acto de ejecutar un programa de ordenador, un conjunto de programas , u otras operaciones , a fin de evaluar el rendimiento relativo de un objeto , normalmente mediante la ejecución de una serie de pruebas estándar y los ensayos en contra de ella . El término ' benchmark ' también se utiliza sobre todo para los fines de los propios programas de benchmarking elaboradamente diseñados.

Benchmarking se asocia generalmente con la evaluación de las características de rendimiento de hardware , por ejemplo , el rendimiento de punto flotante de funcionamiento de una CPU , pero hay circunstancias en que la técnica también es aplicable al software . Puntos de referencia de software están , por ejemplo, van en contra de los compiladores o sistemas de gestión de bases de datos .

3.2. CPU core benchmarking

A pesar de que no se corresponde con la forma en que utilizaría un procesador en una aplicación real , a veces es importante aislar el núcleo de la CPU de los otros elementos del procesador y centrarse en un elemento clave. Por ejemplo , es posible que desee tener la capacidad de hacer caso omiso de la memoria y los efectos de E / S y se centran principalmente en la operación de el pipeline. Este es el dominio de CoreMark . CoreMark es capaz de probar la estructura de pipeline básica de un procesador , así como la capacidad de prueba de lectura / escritura de operaciones básicas , operaciones de enteros y operaciones de control

3.3. CoreMark

CoreMark es un punto de referencia que tiene como objetivo medir el rendimiento de las unidades centrales de procesamiento (CPU) utilizados en sistemas embebidos. Fue desarrollado en 2009 por Shay Gal -On en EEMBC y está destinado a convertirse en un estándar de la industria , en sustitución de la referencia Dhrystone anticuada . El código está escrito en código C y contiene las implementaciones de los algoritmos siguientes : procesamiento de lista (encontrar y ordenar) , Matrix (matemáticas) manipulación (operaciones con matrices comunes) , máquina de estados (determinar si un flujo de entrada contiene números válidos) y CRC

4. Software OpenSource y Libre

la intro al tema contando un poco las comunidades de codigo abierto y por ulitmo de Open-Core

Ejemplo: A medida que la popularidad y la utilidad de Internet ha crecido , también lo han hecho las comunidades de opensource. La comunicación fue el inicio para las grandes comunidades de opensource. Lo que dio como resultado un sinnúmero de comunidades y grupos que contribuye a abrir el desarrollo fuente de casi cualquier cosa.

Sitios web de gran tamaño para que las comunidades se centraron en el desarrollo de software de aplicaciones informáticas , como SourceForce , freshmeat , Ohloh y hacia arriba de acogida CPAN de decenas de miles de proyectos . Un grupo llamado Freenode proporciona Internet Relay Chat (IRC) servidores donde decenas de miles de desarrolladores de código abierto se reúnen para interactuar .

Hay una serie de pequeños servicios de alojamiento de proyectos libres dirigidos a grupos desarrollo de software como Google Code, Launchpad, GitHub, GNU Savannah, así como los sitios de la comunidad antes mencionados.

OpenCores es sitio/ comunidad más grande para el desarrollo de los IP core de hardware como de código abierto del mundo. OpenCores.org proporciona el código fuente de los diferentes proyectos HW digitales (IP-cores, SoC, boards, etc) y apoyar a los usuarios con diferentes herramientas, plataformas, foros y otras informaciones útiles.

4.1. Deferencias

Aca vamos a poner las libertades que permiten uno y otro

4.2. GPL

4.3. LGPL

4.4. OpenSource

intro a opensource

EJ La apertura del código de la propiedad intelectual desarrollada para el proyecto OpenRISC, y otros en OpenCores, ha sido a la vez un obstáculo y una ayuda, pone a la vista el estado de el desarrollo, pero es útil, ya que permite que cualquiera pueda participar en el continuo desarrollo de cores.

4.5. ¿Quien tiene el Hardware?

la idea es poner pro y contra de trabajar con open source EJ El uno de los problemas que enfrentan los desarroladores del opensource que no lo tiene lo desarrolladores de software es la necesidad de usar plataformas privativas para implementar el diseño.Estas plataformas que como elemento base una FPGA, contienen múltiples periféricos ICs , que tienen que ser comprados , así como la depuración y la programación de hardware. Se suma a esto que generalmente las complicadas herramientas de programación de los proveedores,

4.6. OpenRISC

aca vamos a poner el objetivo del proyecto openRisc

EJ: La mayoría de los proyectos de software libre tienen por objeto poner en práctica soluciones relativamente bien conocidas de una manera que permite la apertura y elimine las restricciones que se encuentran en otra implementaciones propietarias .

El objetivo del proyecto OpenRisc de codigo abierto es proporcionar algo útil , productivo y abierto. Los objetivos de estos proyectos de código abierto por lo general están alineados. Por ejemplo , considere el éxito obtenido por el proyecto del kernel de Linux, con miles de colaboradores y decenas de empresas que participan regularmente en el desarrollo . El kernel Linux compite claramente con el propietario variantes de UNIX y de los productos de Microsoft y Apple en todos los casos está experimentando un enorme éxito.

7.4.1.2. Digilent

7.4.1.3. Altera