Uso de RISC-V para la optimización de FPGAs

De Sousa, Loyarte, Set

Abstract—Este artículo intenta detallar el estado del arte de la arquitectura RISC-V y ampliar sobre algunos conceptos. Analizaremos la técnicas de integración hardware-software para obtener mejoras relevantes en performance y eficiencia para el caso de procesadores FPGA. En esas técnicas se utilizará la potencialidad de instrucciones de RISC-V ya visto en ejemplos reales de la industria sobre procesadores convencionales y trasladarlo a la flexibilidad de las FPGA, pudiendo optimizar de forma más profunda escenarios complejos que se presentan cada vez más al incursionar sobre la IoT o el encarecimiento del desarrollo. Primeramente analizaremos las ventajas de RISC-V como arquitectura, tomando su documentación como guía y luego analizando sobre sus puntos fuertes cuales ventajas tiene sobre x86. Luego, profundizaremos sobre FPGA para poder relacionar los beneficios que encontramos en RISC-V a las soluciones FPGA. Por último, ahondaremos sobre las ventajas que aparecen al combinar ambas tecnologías.

Index Terms—RISC-V, FPGA.

1 Introducción

En este trabajo buscaremos analizar la compatibilidad entre RISC-V y las FPGA detallando un poco cada uno y buscando puntos comunes al implementar procesadores RISC-V en FPGAs. En la sección 2 explicaremos un poco sobre la arquitectura RISC-V y sus ventajas y hablaremos sobre como extender la arquitectura. En la sección 3 detallaremos de manera similar las FPGA, como están compuestas y como se configuran. Luego, en la sección 4 estudiaremos como juntar ambas tecnologías. Por último, el trabajo concluye en la 5ta sección.

2 Breve introducción sobre RISC-V

Esta sección detalla los componentes básicos para una arquitectura RISC-V. La primer característica que vemos en RISC-V es la adaptabilidad para usarse en cualquier tipo de sistema. Podemos ver implementaciones de RISC-V en sistemas embebidos de artefactos comunes, tambien podemos encontrar sistemas operativos montados sobre RISC-V para computadoras complejas con virtualización y manejo de entornos de usuario. Por último, existen también emuladores de RISC-V (Spike, QEMU, rv8) e implementaciones sencillas más bien académicas en lenguajes como C o Rust.

2.1 Registros

La arquitectura cuenta con 31 registros de propósito general llamados x1-x31 y un registro adicional, el program counter (pc). A diferencia de otras arquitecturas, la ABI no cuenta con un registro específico que funcione como stack pointer ni uno que se tome como valor de retorno (como `%eax` en x86).Sin embargo, la convención es que se utilice el registro x1 para el valor de retorno, y el registro x2 para el stack pointer.

2.2 Tamaño de instrucciones

Las instrucciones base de la ISA de RISC-V son de tamaño fijo de 32 bits pero el encoding de las instrucciones permite agregar instrucciones de tamaño variable. Parte del espacio de códigos de instrucción ha sido reservado oportunamente para alojar instrucciones de mayor tamaño. Las instrucciones mayores a 32 bits usan una máscara en los bits más bajos para tomar más de una palabra siguiendo el siguiente esquema, por ejemplo, las instrucciones de 32 bits tienen los últimos 2 bits en 1 y los 3 siguientes deben ser diferentes de 111 como se puede observar en la Figura 1.

Fig. 1. Tamaños de instrucción en RISC-V [1]

2.3 Formato de instrucciones

Hay 6 formatos de instrucción, R, I, S, B, U y J, como muestra la figura 2. Todos son de 32 bits y deben estar alineados a 4 bytes. Donde rs1 y rs2 representan los registros de origen y rd el de destino. Los campos opcode, funct3 y func7 sirven para identificar la instrucción que se está ejecutando. Mientras que imm[...] hacen referencia a valores inmediatos.

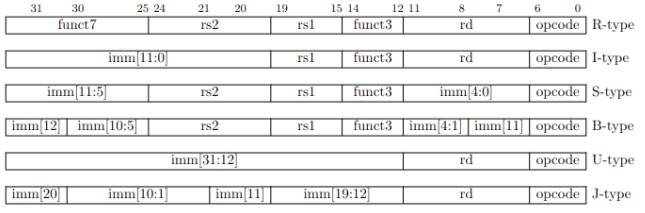

Fig. 2. Formatos de instrucción en RISC-V [1]

De cada formato hay:

1) R-type (Register type): 12 instrucciones
2) I-type (Immediate type): 18 instrucciones
3) S-type (Store type): 6 instrucciones
4) B-type (Branch type): 8 instrucciones
5) U-type (Upper Immediate type): 2 instrucciones
6) J-type (Jump type): 1 instrucción

En resumen, el set de instrucciones de RISC-V cuenta con 47 instrucciones en total mientras que, por ejemplo, la arquitectura x86 de Intel cuenta con varios cientos de instrucciones.

2.4 Ejemplo instrucciones en RISC-V

- Operaciones con Registros (R-Type)
    1) **add x3, x1, x2**, suma los registros x1 y x2 y guarda el resultado en x3.
    2) **sub x4, x5, x6**, hace la resta x5 - x6 y guarda el resultado en x4.
    3) **and x7, x8, x9**, hace el and bit a bit entre x8 y x9 y guarda el resultado en x7.

2.5 Extendiendo RISC-V

Es posible agregar instrucciones a la arquitectura, para ello debemos elegir un formato acorde a la instrucción que querramos implementar. La tabla (fig. 3) muestra la cantidad de instrucciones disponibles en el espacio de instrucciones de la arquitectura.

Size	Usage	# Available in standard instruction length			
		16-bit	32-bit	48-bit	64-bit
14-bit	Quadrant of compressed 16-bit encoding	3			
22-bit	Minor opcode in base 32-bit encoding		2^8	2^{20}	2^{35}
25-bit	Major opcode in base 32-bit encoding		32	2^{17}	2^{32}
30-bit	Quadrant of base 32-bit encoding		1	2^{12}	2^{27}
32-bit	Minor opcode in 48-bit encoding			2^{10}	2^{25}
37-bit	Major opcode in 48-bit encoding			32	2^{20}
40-bit	Quadrant of 48-bit encoding			4	2^{17}
45-bit	Sub-minor opcode in 64-bit encoding				2^{12}
48-bit	Minor opcode in 64-bit encoding				2^9
52-bit	Major opcode in 64-bit encoding				32

Fig. 3. Cantidad de instrucciones restantes para cada tamaño específico de instrucción según los diferentes formatos [Manual de instrucciones de RISC-V

Por ejemplo, si quisiéramos agregar una instrucción 'incr', que incremente el valor de un registro en algún valor inmediato, deberíamos usar una instrucción de formato U-Type. Según la tabla, podríamos usar una instrucción de 32 bits, la cual está disponible en la longitud estándar de instrucción de RISC-V.

Un ejemplo de uso de la instrucción sería:

incr x1, 5

En conclusión, la expansibilidad de la arquitectura nos permite adaptarla a la complejidad que necesitemos en nuestra implementación teniendo un montón de libertad para implementar operaciones que necesitemos resolver de forma atómica.

2.6 ¿Por qué RISC-V?

Tomando gran parte de las especificaciones y detalles sobre RISC-V, las ventajas que encontramos para usar esta arquitectura sobre x86 o AMD64 es la flexibilidad al diseñar su arquitectura y partir de un ISA corto. Este punto es esencial si se desea diseñar soluciones modulares o que necesiten un abanico especifico de instrucciones.

Al poder diseñar instrucciones tomando la base del ISA de RISC-V, eso nos permite modelar nuestros procesadores hacia instrucciones dedicadas para un tipo de procesador, pese a que sea menos óptimo con relación a instrucciones ya implementadas en x86. Un ejemplo es la utilización de la arquitectura SiFive de RISC-V implementada para satélites en módulos donde el rendimiento es clave.

Otra gran ventaja es que, al ser una tecnología de código abierto, la comunidad de RISC-V crece cada vez más, permitiendo explorar diferentes soluciones y aplicaciones de la arquitectura.[5]

Ambas ventajas tienen en común que la optimización de equipos muchas veces se vió limitada por las restricciones de las fabricantes de procesadores por trabajar con una estructura cerrada y no ajustable; y las especificaciones de arquitecturas preexistentes como X86 que no permitían extender o formular una ISA reducida de instrucciones aplicada a las necesidades.

En contrapartida RISC-V no realiza predicción de instrucciones en el pipelining, obteniendo algunas limitaciones al aplicarlo en soluciones más convencionales.

3 FPGAs

Una **FPGA** (*Field Programmable Gate Array*) es un complejo circuito integrado digital programable compuesto por bloques lógicos configurables (*CLB*) y puertos de entrada/salida (*IOB*), cuya interconexión y funcionalidad puede ser programada mediante un lenguaje de descripción especializado. Su principal característica y ventaja es que pueden reprogramarse para un trabajo específico o cambiar sus requisitos después de haberse fabricado. Esto también implica que en muchos casos se pueden hacer cambios físicos sin hacer modificaciones costosas en la placa que lo soporta.[2]

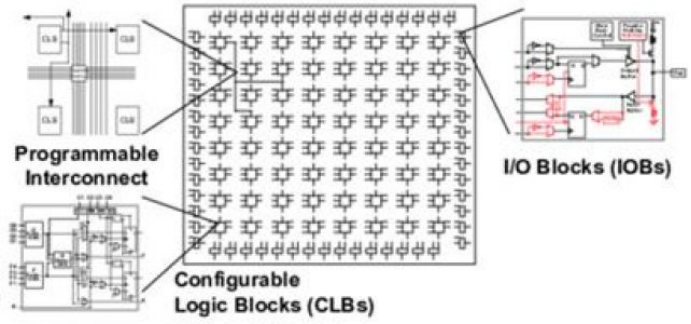

Fig. 4. Una FPGA se compone de Bloques Lógicos Configurables (CLB) o *Logic Cells* (LC) que están interconectados por una serie de uniones programables llamadas Fabric (tejido), que sirven de buses para el intercambio de señales entre CLBs.[8]

En resumen, una **FPGA** es un conjunto de múltiples circuitos (lógicos y de otros tipos) dispuestos matricialmente, cuyas interconexiones son programables por el usuario para la aplicación requerida. En una **FPGA** se programa

su hardware, a diferencia de los microcontroladores / microprocesadores, en los que solo existe un hardware fijo y se programa su *software* (*firmware*).

3.1 Implementando un microprocesador en una FPGA

Podemos separar las etapas de implementación generales para lograr desarrollar un microprocesador de la siguiente forma:

1) Selección del microprocesador: Como primer paso, debemos decidir qué microprocesador vamos a implementar en la FPGA. Podemos elegir entre microprocesadores existentes, como ARM o RISC-V, o incluso optar por diseñar uno desde cero.
2) Arquitectura del microprocesador: Si decidimos diseñar nuestro propio microprocesador, debemos definir su arquitectura. Esto implica especificar el conjunto de instrucciones que el microprocesador debe soportar, la organización de la memoria, la unidad de control, la ALU y otros componentes fundamentales.
3) Diseño RTL: Una vez que definimos la arquitectura, tenemos que traducirla a un diseño de Hardware Description Language (HDL), como VHDL o Verilog. En esta etapa, crearemos el código RTL (Register Transfer Level) que describe el comportamiento de cada componente del microprocesador.
4) Síntesis: Utilizando herramientas de síntesis, como Xilinx Vivado o Intel Quartus Prime, traduciremos el código RTL a una representación lógica a nivel de compuertas (netlist). Esta etapa se encarga de convertir el diseño en un conjunto de bloques lógicos y conexiones dentro de la FPGA.
5) Mapeo de recursos: El siguiente paso es asignar los bloques lógicos generados por la síntesis a los recursos físicos específicos de la FPGA, como LUTs (Look-Up Tables), flip-flops, RAM, multiplicadores, etc.
6) Enrutamiento: Una vez que los bloques lógicos están asignados a los recursos físicos, el siguiente paso es el enrutamiento. El enrutamiento implica conectar los bloques lógicos mediante las interconexiones disponibles en la FPGA. Es una tarea compleja que busca minimizar las interferencias y garantizar que todas las conexiones necesarias estén establecidas correctamente.
7) Carga del bitstream: Una vez completado el enrutamiento, se genera el bitstream, que es el archivo binario que contiene la configuración específica para programar la FPGA. Este archivo se carga en la FPGA, y la configuración se aplica físicamente a los bloques lógicos, lo que configura el microprocesador en el chip FPGA.
8) Integración del sistema: Una vez que el microprocesador está funcionando correctamente en la FPGA, podemos integrarlo con otros componentes del sistema para crear un sistema completo.

4 Mezclando RISC-V con FPGAs

4.1 Ventajas al usar RISC-V en FPGAs

- Personalización:
  Al tratarse de un set de instrucciones de código abierto, RISC-V permite acceder completamente a sus especificaciones y personalizarlas para cada necesidad. Esto hace que se complemente de buena manera con las FPGA, debido a su capacidad para ser programadas, pudiendo así los usuarios implementar un procesador diseñado específicamente para la aplicación en cuestión.
- Flexibilidad:
  Las FPGAs, al ser dispositivos reconfigurables, nos permiten modificar su circuito interno y funcionalidad después de la fabricación del mismo. Al combinar esta propiedad con el diseño modular de RISC-V, se permite al usuario adaptar el procesador a diferentes casos de uso. Por ejemplo, podríamos agregar diferentes módulos y periféricos a la FPGA cuando lo necesitemos, como un acelerador gráfico para cuando se trabaje con algoritmos de procesamiento de imágenes.
- Diseño de hardware y software:
  La naturaleza abierta de RISC-V permite una estrecha colaboración entre el desarrollo de hardware y software. Con una FPGA, podemos diseñar tanto el procesador RISC-V como los aceleradores de hardware o periféricos para satisfacer las necesidades especificas de la aplicación.

4.2 Aplicaciones comunes de RISC-V en FPGAs

- IA Embebida:
  La aplicación más común donde se implementa RISC-V en FPGA. Consiste en ejecutar Inteligencia Artifical en un sistema embebido que requiera poca latencia y bajo consumo eléctrico. El uso de RISC-V permite a los desarrolladores optimizar el softcore en la configuración de su FPGA, reduciendo el conteo de componentes, y optimizar la relación computo vs. consumo. Por ejemplo, Fraunhofer IMS AIRISC que es un procesador RISC-V con implementación diseñada para FPGA optimizado para aplicaciones con IA embebida, permitiendo una personalización de periféricos para diseñar un System on Chip(SoC)[6].
- Fusión de sensores:
  En el área de la fusión de sensores se está utilizando mucho FPGA dada a la gran cantidad de I/Os disponibles en este tipo de procesadores, permitiendo múltiples flujos de información para ser agregados y procesados. RISC-V permite a un desarrollador para que cada núcleo optimice el computo para un grupo especifico de sensores, así como implementar interfaces personalizadas utilizando componentes externos. El Sistema en Modulo RCHD-PF PolarFire utiliza una tecnología FPGA con instrucciones RISC-V para poder operar fusión de sensores en automóviles, automatización industrial, Inteligencia Artificial y otras aplicaciones.[7].
- Visión:
  Esta particularidad de la fusión de sensores implica la suma de múltiple flujos de información, y posiblemente un procesamiento implementado a demanda

de ese flujo entrante. Una implementación con RISC-V en una FPGA para esas tareas ofrece una oportunidad para eliminar funciones sin utilizar y adapta los algoritmos de procesamiento en un solo componente. Resultando en un menor consumo energético y menor latencia que en un cambio de chipset o de la arquitectura del sistema. El chip RCHD-PF PolarFire[7] también se utiliza para el procesamiento de imágenes y vídeo.

5 CONCLUSIÓN

Combinar RISC-V con una FPGA aprovecha la apertura, flexibilidad y versatilidad de ambas tecnologías. Ya sea para desarrollo de proyectos para el mercado o para visualizaciones del funcionamiento básico de procesadores, estas tecnologías nos permiten crear arquitecturas personalizables, escalables y eficientes con rapidez y sin las ataduras que a veces se presentan con el uso de set de instrucciones CISC. A su vez, el ecosistema RISC-V esta cada vez mas maduro y dispone de una amplia gama de recursos y procedimientos para el correcto desarrollo sobre FPGAs.

REFERENCES

[1] *RISC-V Official Documentation*. RISC-V.org. [Online]. Available: https://riscv.org/technical/specifications/

[2] *RISC-V and FPGAs*. Efinix, Inc. [Online]. Available: https://www.efinixinc.com/blog/riscv-and-fpgas.html

[3] *FPGA-Based Infrastructure, With RISC-V Prototype, To Enable Implementation and Evaluation Of Cross-Layer Techniques In Real HW (Best Paper Award)* TECHNICAL PAPER LINK, [Online]. Available: https://semiengineering.com/fpga-based-infrastructure-with-risc-v-prototype-to-enable-implementation-evaluation-of-cross-layer-techniques-in-real-hw-best-paper-award/

[4] *FAC-V: An FPGA-Based AES Coprocessor for RISC-V* FAC-V: An FPGA-Based AES Coprocessor for RISC-V by Tiago Gomes, Pedro Sousa ,Miguel Silva, Ekpanyapong and Sandro Pinto , [Online]. Available: https://www.mdpi.com/2079-9268/12/4/50

[5] *RISC-V vs. ARM vs. x86 – What's the difference?* , Jeff Shepard [Online]. Available: https://www.microcontrollertips.com/risc-v-vs-arm-vs-x86-whats-the-difference/

[6] *AIRISC - The RISC-V Processor for Embedded AI* , Fraunhofer-IMS [Online]. Available: $https : //github.com/Fraunhofer - IMS/airisc_core_complex$

[7] *RCHD-PF SYSTEM ON MODULE* , Conclusive Engineering [Online]. Available: https://conclusive.pl/products/rchd-pf-som/

[8] *FPGA: qué es y cuáles son las características de este componente* Juan Manuel Manchado Ortega y Jorge Antonio García Pérez, Akka Technologies Spain [Online]. Available: https://www.akka-technologies.com/fpga/