Capítulo 2

LAGARTO: UN PROYECTO PARA EL DESARROLLO DE CPU'S Y ACELERADORES HARDWARE CON ISA DE CÓDIGO ABIERTO PARA LA ACADEMIA, LA INVESTIGACIÓN Y LA INDUSTRIA

Marco Antonio Ramírez Salinas[1], Luis Alfonso Villa Vargas[1], Francesc Moll Echeto[2], Lluís Terés Téres[3], Miquel Moretó[4], Adrian Cristal Kestelman[4], Mateo Valero Cortes[4]

[1]Centro de Investigación en Computación-IPN, México

[2]Universidad Politècnica de Catalunya, España

[3]Centro Nacional de Microelectrónica IMB-CNM (CSIC), España

[4]Centro Nacional de Supercomputación-BSC, España

mars@cic.ipn.mx, lvilla@cic.ipn.mx, francesc.moll@upc.edu, lluis.teres@imb-cnm.csic.es, miquel.moreto@bsc.es, adrian.cristal@bsc.edu, mateo.valero@bsc.es

https://doi.org/10.3926/oms.411.2

Ramírez Salinas, M. A., Villa Vargas, L. A., Moll Echeto, F., Terés Térex, Ll., Moretó, M., Kestelman, A. C., & Valero Cortes, M. (2022). Lagarto: Un proyecto para el desarrollo de CPU's y aceleradores hardware con ISA de código abierto para la academia, la investigación y la industria. En M. A. Ramírez Salinas, L. N. Oliva Moreno, L. I. Garay Jimenez y P. Gomez Miranda (Ed.), *Avances 2022: Red de Investigación Computación del Instituto Politécnico Nacional, México* (pp. 21-41). Barcelona, España: OmniaScience.

Resumen

Lagarto, es un proyecto para el diseño de CPU's que inicio en 2010 y que continúa en desarrollo por profesores y estudiantes del laboratorio de Microtecnología y Sistemas Embebidos del Centro de Investigación en Computación del IPN, son varios los proyectos de investigación soportados por la Secretaría de Investigación y Posgrado del IPN, 20211682, 20182219, 20150957 y 20101320, que han servido para ir incrementando el nivel de madurez tecnológica de la familia de procesadores lagarto. Aunque se sigue trabajando en nuevos desarrollos como: Arquitecturas *Multithreading* Multinúcleos, Vectoriales, y Aceleradores de IA, los dos productos de mayor madurez tecnológica son Lagarto Hun, un procesador escalar en orden y Lagarto Ka un procesador superescalar con ejecución fuera de orden, ambos procesadores se han modelado a nivel RTL (*Register Transfer Level*) con el lenguaje de descripción de hardware HDL, Verilog. En los últimos años, en 2019 en alianza con el Centro de Supercomputación de Barcelona BSC (*Barcelona Supercomputing Center*), el Departamento de Ingeniería Electrónica de la Escuela de Ingeniería en Telecomunicaciones de la Universidad Politécnica de Cataluña (UPC) y el Centro Nacional de Microelectrónica, se ha incrementado aún más su madurez tecnológica al fabricarse el primer chip de Lagarto Hun, en una tecnología de TSMC con transistores 65nm a través del programa europeo EURUPRACTICE. El proyecto y sus productos han inspirado a varias generaciones de estudiantes de la Maestría en Ciencias en Ingeniería de Cómputo, Maestría en Ciencias de la Computación y Doctorado en Ciencias de la Computación del CIC-IPN en México y del *Master in Innovation and Research in Informatic* (MIRI) y del Doctorado en Arquitectura y Tecnología de Computadoras de la UPC en Barcelona España, para seguir investigando y proponiendo ideas innovadoras en esta área de la ingeniería

y la computación. A partir de 2021 la Red de Investigación y Posgrado en Computación del IPN, ha decidido como una estrategia institucional poner a disposición de las academias la tecnología desarrollada, como una plataforma para la impartición de cursos de las materias de Diseño de Sistemas Digitales, Diseño VLSI, Microprocesadores, Arquitectura de Computadoras, y Sistemas Operativos.

Palabras clave

SoC, Arquitectura de Computadoras, Microarquitectura de Procesadores, FPGA's, Verilog

1. Introducción

Las tecnologías emergentes relacionadas con las TIC a nivel mundial están basadas en sistemas de cómputo modernos, con alto desempeño y de bajo consumo de energía. Es el caso de los CPU's utilizados para la construcción de Computadoras de propósito general, Dispositivo móviles, Computadoras de Vehículos Autónomos (Aéreos, Terrestres y Submarinos), Robots Inteligentes, Espacios de trabajo inteligentes, Sistemas de interfaz cerebro-computadora, Hogar conectado e Internet de las Cosas (IoT). Una variante reciente son los CPU neuromórficos inspirados por arquitecturas neurobiológicas simples, pero masivamente paralelas con alta interconectividad utilizadas como aceleradores de hardware que utilizan técnicas de IA embebidas para razonamiento probabilístico como aprendizaje de máquina y redes neuronales profundas. Estas tendencias dan cabida a la generación de nuevas ideas para desarrollar CPU's que resuelvan de forma eficiente mediante técnicas innovadoras los retos de las tecnologías emergentes. Existen muchas clases de CPU's para computadoras clasificadas por su segmento de aplicación:

Computadoras personales, son máquinas de propósito general para las cuales se ha desarrollado una gran variedad de software que se ha popularizado porque, a través de sistemas informáticos amigables, facilitan el trabajo y la comunicación en oficinas, escuelas y el hogar.

Servidores de cómputo, son máquinas de alta capacidad de almacenamiento de información y alto desempeño de cálculo. Sus servicios dependen de redes de comunicación y su capacidad computacional se puede configurar en el rango que va desde pequeños servidores de herramientas de cómputo, hasta supercomputadoras construidas por miles de nodos interconectados por redes de alta velocidad. Dentro de sus aplicaciones se pueden considerar servicios bancarios, sistemas de administración tributario de un país, servicios de cómputo en la nube (Amazon, IBM, Microsoft), y en otros casos cómputo científico.

Computadoras embebidas, son máquinas de propósito especifico con características particulares de hardware y software, que forman parte de dispositivo, equipo o herramienta más complejos. Dentro de sus aplicaciones se pueden considerar Equipo médico (Bombas de Infusión, ECG, Tensiómetro, Glucómetro, Oxímetros, etc.), Equipo de oficina y hogar (Teléfonos, Pantallas de TV, Impresoras, etc.), Electrodomésticos (Estufas, Hornos de Microondas, Refrigeradores, Lavadoras, etc.), Computadoras de vehículos aéreos, terrestres y marinos, etc.

Computadoras móviles, son máquinas de uso personal de propósito general, que utilizan técnicas de alto desempeño, bajo consumo de energía; y sus servicios explotan de forma intensa los servicios de redes de comunicación inalámbricas. Dentro de sus aplicaciones se pueden considerar Tabletas de cómputo, Telefonía celular, Cámaras fotográficas, TV portátil, Reproductores de música, Relojes inteligentes, etc.

Un dato relevante es que el núcleo de los procesadores utilizados en la mayoría de las distintas clases o tipos de computadoras, utilizan prácticamente la misma microarquitectura con adición de bloques funcionales y técnicas de desempeño y bajo consumo de energía apropiados para el segmento de mercado que atienden, como aceleradores de hardware, controladores de periféricos y dispositivos de comunicación inalámbrica, así mismo de forma general se usan los mismos componentes para todas las clases de computadoras como dispositivos de interfase con el usuario (pantalla, teclado y ratón), dispositivos de almacenamiento (Disco duro. CD, DVD, Memoria *flash*), Adaptadores de red (Ethernet y Wifi), GPS –*Global Positioning System*– y dispositivos de comunicación (*BlueTooth*, NFC –*Near field communication*–).

2. Métodos y materiales

La CPU (*Central Processing Unit*) es hasta ahora el semiconductor más complejo que se ha diseñado por el ser humano, emplea una aritmética computacional con propiedades matemáticas precisas que deben de conservarse en todo momento a fin de generar resultados consistentes. Para su implementación se utilizan distintos bloques funcionales de software y hardware que se encargan de transformar los códigos de programación de alto nivel en lenguaje de máquina, o incluso de procesos más complejos como el diseño de algoritmos y técnicas de microarquitectura para optimizar su ejecución al interior del chip. Por otro lado, con el desarrollo del Sistema Operativo Linux como una plataforma de código abierto y gratuito, se reduce la brecha tecnológica para la implementación de los SO, facilitando así la incorporación de una gran variedad de versiones de Linux en dispositivos de cómputo disponibles en el mercado. Así mismo el desarrollo las herramientas de diseño electrónico automatizado EDA (Electronic Desing Automation) que utilizan lenguajes de descripción de hardware HDL (Hardware Description Languages) reducen la brecha tecnológica y el esfuerzo de ingeniería que hace algunos años se requería para el diseño de una CPU. Finalmente, los

consorcios académicos para la fabricación de semiconductores como MOSIS y EUROPRACTICE ofrecen distintas opciones asequibles para la producción de semiconductores a nivel de investigación y a escala industrial.

En este capítulo presentamos los avances del diseño de Lagarto Hun, un procesador escalar segmentado con ejecución en orden, que utiliza el conjunto de instrucciones de código abierto RISC-V.

2.1. Aritmética computacional

La aritmética utilizada en el conjunto de instrucciones que las computadoras "entienden "utiliza la numeración binaria, El sistema binario está representado por un subconjunto de dos símbolos {0, 1}, es la base numérica que se adapta de forma eficiente al comportamiento físico de las señales eléctricas, tanto para almacenar información como para transmitirla a través de hilos conductores. Es posible representar cualquier cantidad numérica a partir de un subconjunto de símbolos, el tamaño del subconjunto representa la base. El subconjunto de símbolos y su base implícita, en general representa un sistema numérico. A través de esta base numérica se desarrolló la lógica binaria, el álgebra de Boole y con ellas toda la aritmética computacional que se utiliza en las computadoras modernas. Con la base de esta aritmética computacional se han diseñado las unidades funcionales que forman parte de la etapa de ejecución de la arquitectura Lagarto Hun como son: **ADD** (sumador restador), **MUL** (multiplicador-divisor), **SHIFT** (desplazamientos, lógico y aritmético) y **ALU** (*and, or, xor, not*) para realizar operaciones lógicas bit a bit y sus correspondientes unidades funcionales de punto flotante [1].

2.2. Conjunto de instrucciones RISC-V

Para la familia de procesadores Lagarto se ha utilizado el conjunto de instrucciones RV64I, un conjunto de instrucciones (ISA) de código abierto, en esta versión se utilizan registros de 64 bits de palabra y proporcionan 64-bits de direccionamiento. En el conjunto de instrucciones de RISC-V hay 4 tipos de formatos de instrucciones que conforman el núcleo de instrucciones de números enteros, estos son R/I/S/U. El formato de instrucciones mantiene los campos *Src1, Src2, RDest* de los formatos de instrucciones R/I/S/U en la misma posición

para simplificar el diseño del hardware del decodificador de instrucciones, como se observa en la Figura 1. Hay además un par de variaciones de los formatos de instrucción B/J, basados en el manejo de valores inmediatos. Todas las instrucciones son de 32 bits de longitud y deben estar alineados en memoria en límites de cuatro bytes. Se genera una excepción por una instrucción de salto o salto incondicional tomado, si la dirección destino no está alineada a cuatro bytes (*missaligned*). Esta excepción se reporta en la instrucción de *Branch* o *Jump* no en la instrucción destino y no se generará la excepción si el salto no se toma. Los valores inmediatos tienen siempre extensión de signo y siempre están organizados en la parte de los bits más significativos del formato de la instrucción. En general el bit de signo es siempre el bit 31 de la instrucción.

La diferencia en los formatos S y B es que los 12 bits del campo del valor inmediato se utilizan para codificar el desplazamiento de la instrucción de salto en múltiplos de 2-bytes, por ignorar el bit 0, en el Tipo-B. en vez de desplazar una posición a la izquierda todos los bits del valor inmediato, como se ha hecho tradicionalmente por hardware. Los bits centrales *imm[10:1]* y el bit de signo están en posiciones fijas. Mientras el bit más bajo en el formato de instrucción S *(inst[7])* codifica un bit de orden alto en el formato B. De forma similar la diferencia entre los formatos de instrucción de U y J, es que el bit 20 del valor del inmediato esta recorrido a la izquierda por 12 bits para formar inmediatos sin signo (U), y por un bit para formar

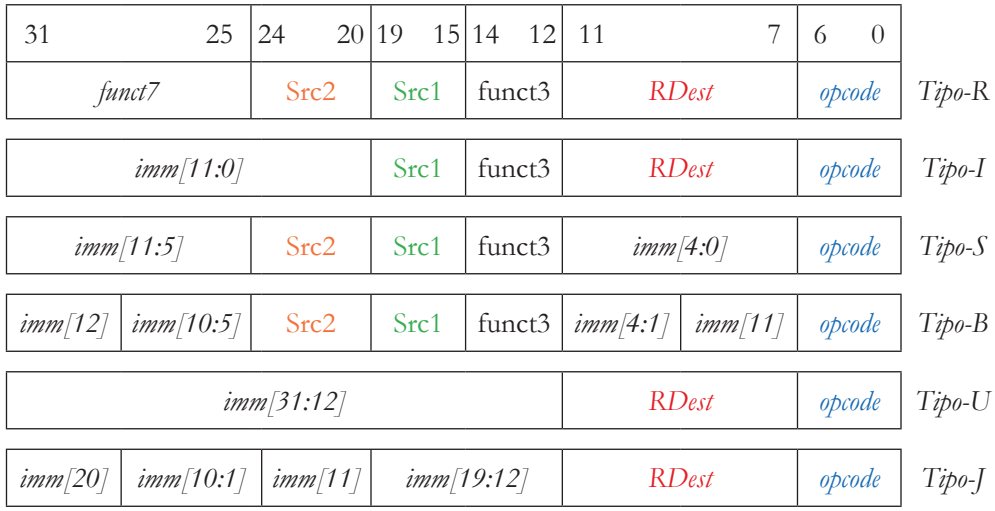

Figura 1. Formato de instrucciones base de RISC-V

saltos inmediatos (J) [2]. Con base en este ISA se ha diseñado, el decodificador de instrucciones y desde luego las estructuras del banco de registros.

2.3. Extracción de instrucciones de memoria cache

El PC calculado ciclo a ciclo, es una entrada para el módulo de predicción de saltos a fin de descubrir si instrucciones previas de saltos condicionales se han decodificado y si se tiene registro de su comportamiento para poder predecir si el salto se toma (*taken=1*) o no se toma (*no taken=0*) antes de que la condición sea calculada ciclos después en la etapa de ejecución, a fin de adelantarse en los próximos ciclos de *fetch* por el camino de ejecución de forma especulativa. Si el predictor de saltos no acierta, se lanza una excepción para recuperar el camino correcto de ejecución, desechando las instrucciones que vienen detrás del salto especulativo, el predictor utilizado para Lagarto Hun es GShare con 1024 entradas [3].

2.4. La jerarquía de memoria cache

La jerarquía de memoria que se ha diseñado para el procesador lagarto, integra memoria de nivel uno separadas (L1: ICache) para instrucciones y para datos (L1: DCache), nivel dos separadas (L2: ICache) para instrucciones y para datos

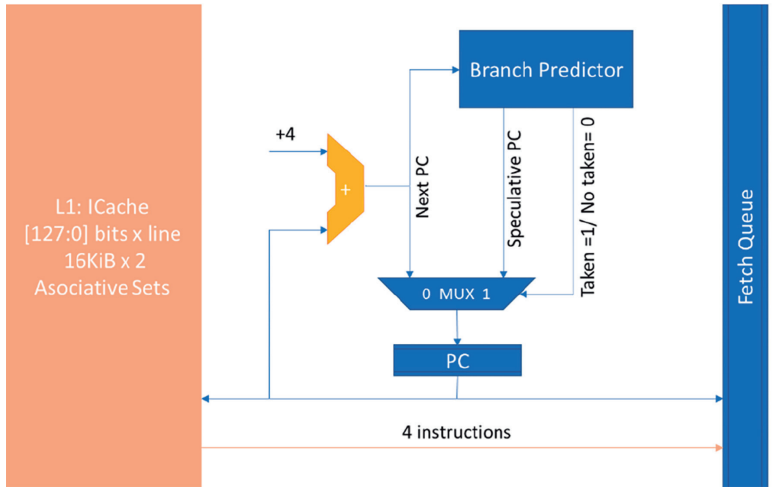

Figura 2. Interfaz de fetch y predictor de saltos de Lagarto Hun RISC-V

(L2: DCache), y nivel 3 con cache unificada (L3: Share Cache) para instrucciones y datos. Para el procesador Lagarto Hun, el contador de programa PC (64 bits) se incrementa por 4 en cada ciclo de reloj, pero realiza una lectura a la memoria cada 4 ciclos de reloj, de esta forma lee de la memoria L1: ICache una línea completa de 128 bits (4 instrucciones de 32 bits). El primer nivel de Cache esta diseñado con la especificación de indexado (*index*) virtual y etiquetado (*tag*) de direcciones físicas. Esta configuración permite el acceso a TLB en paralelo con la memoria Cache en el primer ciclo y permite realizar las comparaciones de Tag (*Etiquetas*) en el segundo ciclo [4] de reloj.

Dentro de la jerarquía de memoria de la Arquitectura Lagarto, se contempla el uso de controladores de memoria DDR, y SDCard, para soportar un S.O. basado en Linux, como un SoC (*System on Chip*).

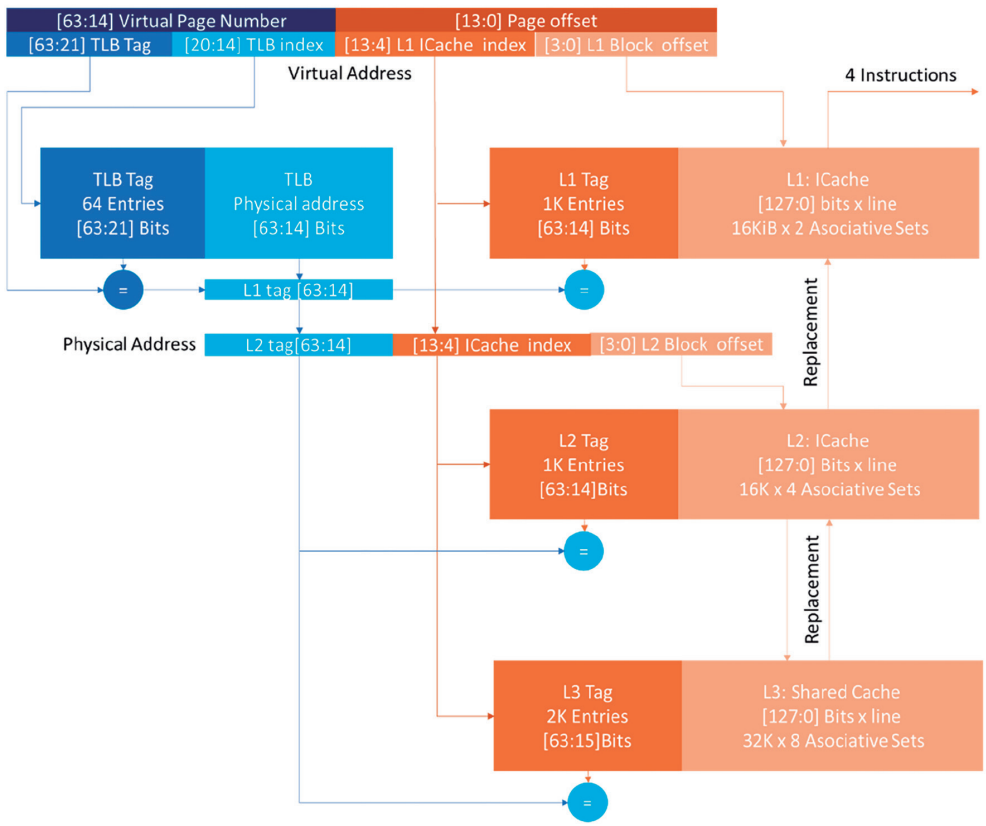

Figura 3. L1-L2-L3 Diseño de Cache de Instrucciones de Lagarto Hun RISC-V

2.5. Decodificador de instrucciones

El mecanismo de *fetch* es el encargado de extraer las instrucciones de la memoria Cache de nivel 1 (L1-ICache), por lecturas de líneas completas de la memoria cache de instrucciones, 4 instrucciones se extraen por ciclo de *fetch*. Estas son almacenadas en un registro inter-etapa tipo *fifo* denominado Cola de Instrucciones Extraídas (*Instruction Fetch Queue*), de donde después se extraen para ser decodificadas.

La máquina de estado del decodificador inicia (en el estado cero) leyendo una entrada de la Cola de Instrucciones Extraídas de la Memoria (IFQ), el estado 1, lee el vector de control de la instrucción, y verifica que no sea una instrucción ilegal o una excepción de sistema, si la instrucción sigue siendo válida, pasa al estado2, en donde se verifica que no exista una solicitud de paro del *pipeline* (*wfp*) o se tenga una interrupción pendiente de atender (*wfi*). de no ser así, pasa al estado 4, en donde la instrucción se envía a ejecución. Si hubiera paro de pipeline o interrupciones en espera de ser atendidas, pasa al estado 3 (*wait*), en espera de recibir el vector de excepción, una vez que es recibida, este se envía para su ejecución. En el estado inicial si la etapa de *fetch* genera una excepción, puede ser un fallo (miss) en L1Cache o un PC con una dirección no alineada en múltiplos de 4 bytes o en el estado 1, si el vector de control

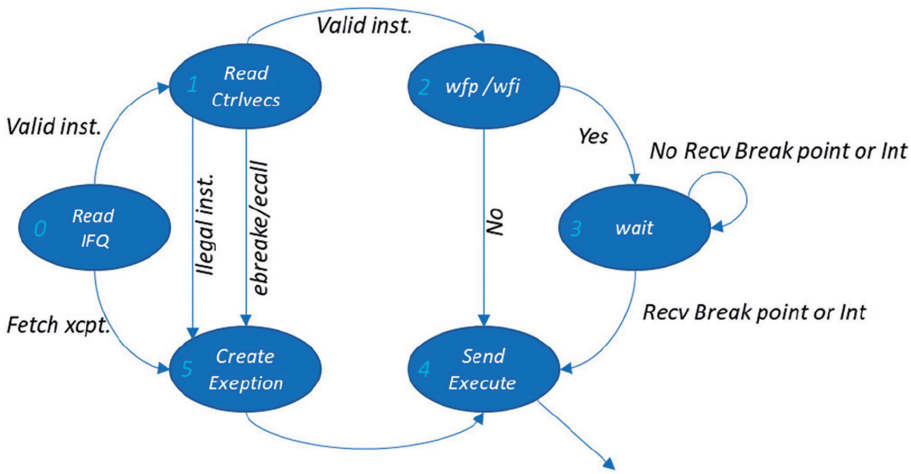

Figura 4. Máquina de estados del decodificador

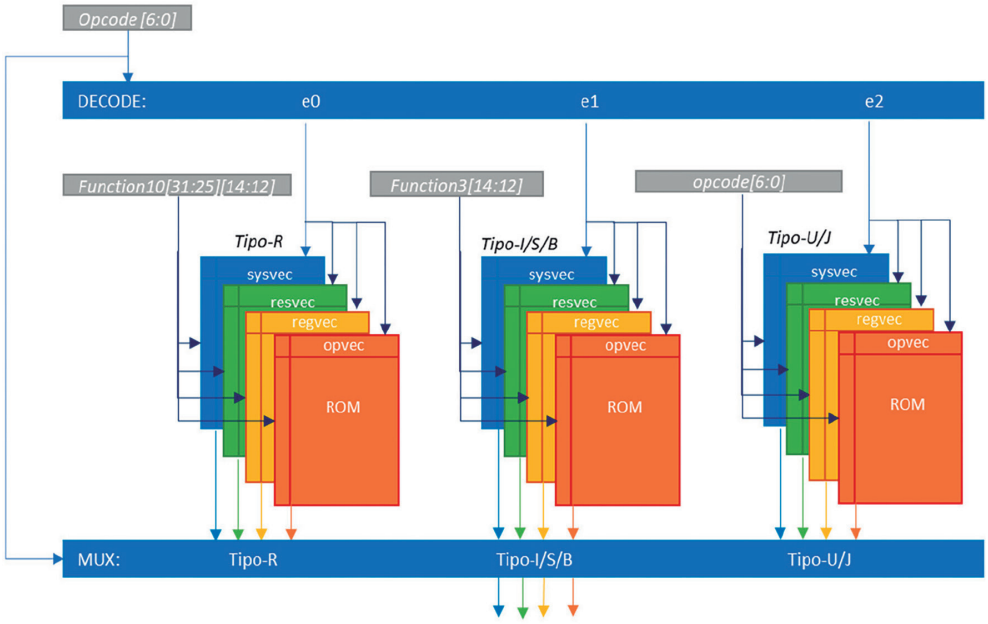

Figura 5. Decodificador de Lagarto Hun

no corresponde a ninguna instrucción valida o corresponde a una instrucción *ecall* o *ebreake*, pasa al estado5, donde se crea el vector de excepción y se envía para su ejecución.

La microarquitectura del decodificador de instrucciones de Lagarto HUn, se muestra en la Figura 5, una vez que la instrucción ha sido leída de la IFQ, se utiliza un decodificador para el campo de *opcode[6:0]* con salidas que habilitan las lecturas de los bancos de memoria ROM que contienen los vectores independientes del vector de ejecución. El vector de ejecución está compuesto por un vector de sistema, un vector de recursos, un vector de registros y un vector de operación, representados en la figura como una flecha en azul, en verde, en amarillo y en naranja respectivamente.

El vector de Sistema: un campo de 6 bits que identifica instrucciones de escritura o *store* (S) y lectura o *load* (L) de memoria, instrucciones de barrera para sincronización de memoria FENCE, instrucciones de pausa (Halt), instrucciones de sistema SYS, instrucciones atómicas AMO de sincronización de cache.

Figura 6. Vector de Sistema

El vector de recursos: un campo de 7 bits que identifica los recursos de hardware necesarios para la ejecución correcta de la instrucción, por ejemplo; la unidad de extensión de signo para operaciones con inmediatos, la detección de instrucciones de salto (*Branch*), las unidades funcionales que ejecutará a la instrucción que está siendo decodifica SLT, SHFT, LOGIC, MUL-DIV y ADD-SUB.

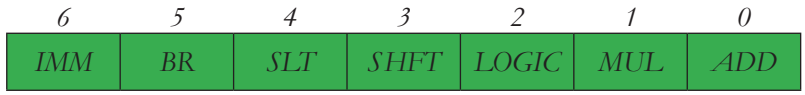

Figura 7. Vector de Recursos

El vector de registros: un campo de 3 bits que identifica los registros disponibles en la instrucción Src1, Src2, RDest y finalmente

Figura 8. Vector de Registros

El vector de operación: un campo de 18 bits que identifica instrucciones especiales como *J, JR,* Saltos condicionales BR*,* instrucciones atómicas de memoria WP *(Wait for Pipeline)* para sincronización*,* instrucciones que operan sobre una palabra W*,* operaciones que hacen uso de la parte alta o baja (HL) de una palabra, instrucciones que requieren que el operando A (*Src1*) sea tratada como un número con signo (SA), instrucciones que requieren que el operando B (*Src2*) sea tratada como un número con signo (SB), instrucciones que realizan cálculo de direcciones relativas al PC, bits reservados para instrucciones futuras (R), instrucciones que definen su operación mediante el campo func3, instrucciones que definen su operación por OPCODE o FUNC7, operaciones de INT, operaciones de MEM, operaciones de manejo de excepciones EH, operaciones de punto flotante FP y operaciones con Vectores (SIMD).

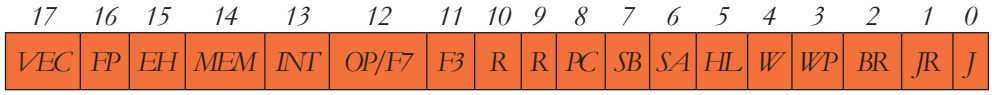

Figura 9. Vector de operación

Finalmente, el mismo campo del código de operación es utilizado para seleccionar el vector de ejecución correspondiente a través del MUX de salida. El vector de operación y el vector de sistema indican el tipo de instrucción decodificada, mientras los bits encendidos del vector de registros son utilizados como habilitadores para la lectura de los operandos fuentes del banco de registros correspondiente para ser enviadas a los puertos de la unidad funcional indicada por el vector de recursos, dependiendo del tipo e instrucción que ha sido decodificada, así mismo el registro destino donde se escribirá el resultado.

2.6. Banco de registros de Enteros y Punto Flotante

Los bancos de registros de Enteros y de Punto Flotante, son estructuras de 32 entradas y 64 bits de palabra multipuerto (Figura 10).

2.7. Ejecución

Una vez que la instrucción es decodificada, el vector de registros se utiliza para habilitar las lecturas de los registros fuente y sus valores se envían a la unidad funcional que corresponde según el vector de recursos decodificado de la propia instrucción y el vector de operación.

Las unidades de ejecución de enteros están diseñadas para resolver en un ciclo de maquina operaciones de desplazamiento de bits (SHFT), operaciones lógicas (LOGIC), operaciones de multiplicación y división (MUL-DIV) y operaciones de suma y resta (ADD-SUB), el sumador da soporte a la instrucción SLT. Mientras las unidades de punto flotante resuelven operaciones de clasificación y conversión de formato en un ciclo de máquina, operaciones de suma y resta (ADD-SUB) en cuatro ciclos de máquina y operaciones de multiplicación y división (MUL-DIV) en doce ciclos de máquina, la ultima etapa de ejecución del procesador es *writeback*, que es donde los resultados de las operaciones se escriben al banco de registros en la dirección indicada por el registro *RDest*,

63 0		63 0	
x0/zero	Alambrado a 0	f0/ft0	Temporales de P.F.
x1/ra	Dirección de retorno	f1/ft1	
x2/sp	Apuntador de *Stack*	f2/ft2	
x3/gp	Apuntador Global	f3/ft3	
x4/tp	Apuntador de *Thread*	f4/ft4	
x5/t0	Temporal /Enlace	f5/ft5	
x6/t1	Temporales	f6/ft6	
x7/t2		f7/ft7	
x8/s0/fp	Salvado de variables/ Apuntador de *frame*	f8/fs0	Salvado de variables de P.F.
x9/s1	Salvado de variables	f9/fs1	
x10/a0	Argumentos de función/ retoro de valores	f10/fa0	Argumentos de funciones PF / Valores de retorno
x11/a1		f11/fa1	
x12/a2	Argumentos de funciones	f12/fa2	Argumentos de funciones de P.F.
x13/a3		f13/fa3	
x14/a4		f14/fa4	
x15/a5		f15/fa5	
x16/a6		f16/fa6	
x17/a7		f17/fa7	
x18/s2	Salvado de variables	f18/fs2	Salvado de variables de P.F.
x19/s3		f19/fs3	
x20/s4		f20/fs4	
x21/s5		f21/fs5	
x22/s6		f22/fs6	
x23/s7		f23/fs7	
x24/s8		f24/fs8	
x25/s9		f25/fs9	
x26/s10		f26/fs10	
x27/s11		f27/fs11	
x28/t3	Temporales	f28/ft8	Temporales de P.F.
x29/t4		f29/ft9	
x30/t5		f30/ft10	
x31/t6		f31/ft11	

pc

Figura 10. Bancos de registros Ent. y F.P. para Lagarto Hun con nemónicos para ensamblador de RISC-V

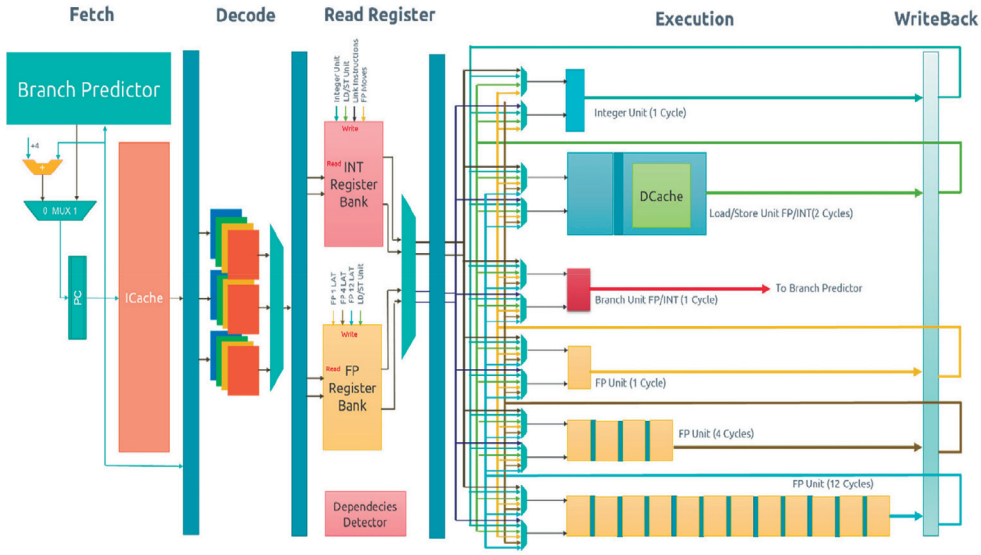

Figura 11. Núcleo del Procesador Lagarto Hun

de la instrucción ejecutada. El valor del resultado es también enviado a la red de *baypass*, un arreglo de multiplexores utilizados para sustituir algún valor obsoleto leído del banco de registros por un valor actual recién calculado, esto sirve solucionar problemas de dependencias de datos entre instrucciones consecutivas en el orden del programa [1].

2.8. Mapa de ruta del proyecto Lagarto

Describir un mapa de ruta del proyecto lagarto, da una visión general del plan trazado para su desarrollo, se definió bajo una perspectiva de 15años para alcanzar productos tangibles e intangibles, dentro de los primeros están la tecnología de semiconductores mexicanos Lagarto Hun y Lagarto Ka, (utilizamos vocablos mayas como identificadores Hun=1, Ka=2, Kan=4) que tiene que ver con el numero de instrucciones que pueden ejecutar por ciclo en cada etapa del procesador segmentado. Con la popularidad de RISC-V como ISA de código abierto, se tomó como estándar para el desarrollo de los núcleos de Lagarto, y para 2022 se estará trabajando con núcleos verificados al 100% para la construcción de Chips multiprocesador, la academia y la investigación es parte fundamental del proyecto, por lo que se ha considerado el desarrollo de *workshops* institucionales

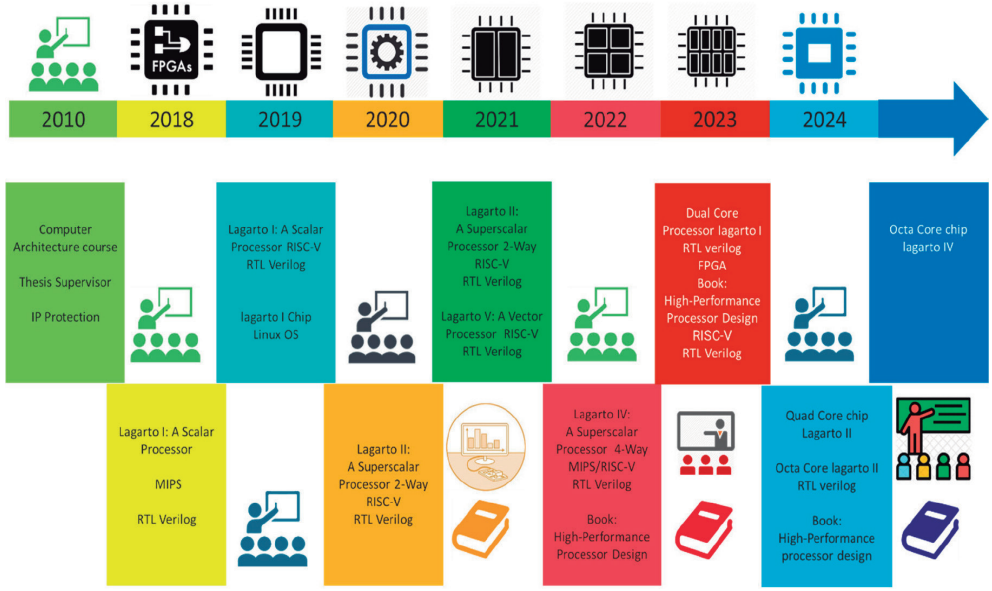

Figura 12. Mapa de ruta del proyecto Lagarto

y la publicación de libros en las diferentes temáticas que aborda este proyecto para la enseñanza en los niveles medio superior, superior y posgrado que ofrece el IPN y su alianza con redes internacionales.

Varias estudios se han abordado a lo largo del desarrollo de este proyecto, de acuerdo con el mapa de ruta planteado desde su origen [5], desde técnicas de microarquitectura para reducir el consumo de energía en las estructuras más complejas del procesador [6], Diseño de unidades de acceso a memoria [7], Diseño de una extensión vectorial para aplicaciones multimedia [8] y otro estudio para aplicaciones no convencionales [9], Arquitecturas *Multithreading* [10], Diseño de unidades funcionales para números enteros y unidades funcionales para números de punto flotante [1], Diseño del bus local para interconectar la jerarquía de memoria y los periféricos [11], la propia Jerarquía de la memoria para un SoC [4]. Diseño de la unidad de manejo de excepciones [12], Análisis y pruebas para el sistema de arranque de un SoC [13], hasta evaluar la posibilidad de construir sistemas multinúcleos utilizando el núcleo de Lagarto I [14], y algunos estudios para verificación funcional de sistemas digitales complejos [15], que nos han dejado con grandes expectativas para seguir haciendo investigación y desarrollo tecnológico con innovación.

3. Resultados y análisis

Un circuito integrado de Lagarto Hun en silicio, capaz de *bootear* Linux, es uno de los resultados alcanzados a la fecha, esto gracias a la colaboración con el Centro de Supercomputación de Barcelona (BSC), el Centro Nacional de Microtecnología (CNM) y la Universidad Politécnica de Cataluña (UPC). El trabajo de esta colaboración comprende el diseño de Lagarto Hun a nivel RTL, la verificación funcional, el diseño VLSI y las pruebas posteriores de validación a nivel eléctrico y funcional de muestras físicas. En este SoC se adaptó la plataforma desarrollada por LowRISC para el núcleo RISC-V denominado Rocket desarrollado por la Universidad de Berkeley USA y remplazando el núcleo de Lagarto Hun, como primera aproximación para su verificación funcional [16].

3.1. El proceso de verificación funcional

El proceso de verificación funcional incluye simulaciones pre-síntesis y post-síntesis, para lo cual se ejecutaron 395 pequeños programas que verifican el ISA instrucción por instrucción y realizan chequeos para determinar si se ejecutaron correctamente. El resultado de estas simulaciones es una lista de verificación de las pruebas realizadas indicando si estas fueron exitosas o no. La simulación a nivel de compuertas, también se utiliza para obtener información de la actividad

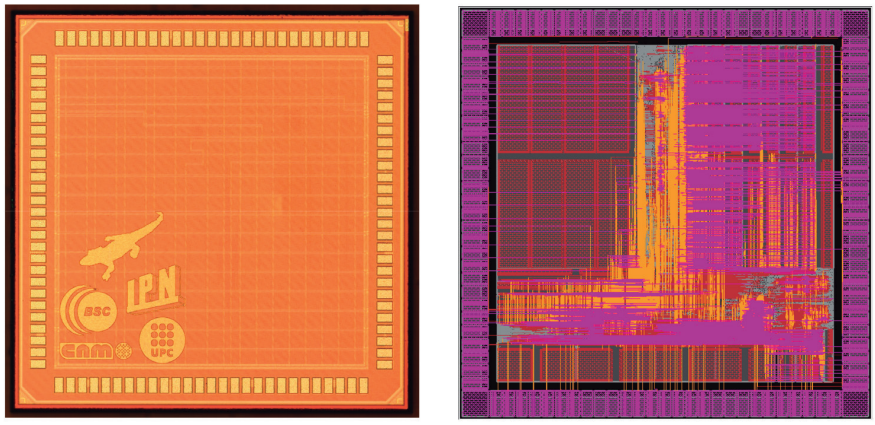

Figura 13. Fotografía del CHIP Lagarto Hun en Silicio con CMOS de 65nm de TSMC Diseño superior (izquierda) y Diseño de todas las capas (derecha)

de los bloques funcionales del C.I. para realizar estimaciones más realistas del consumo de energía del circuito, utilizando el módulo Joules de CADENCE [17].

3.2. Pruebas en FPGA's

La verificación del modelo RTL del SoC en un FPGA, tiene como propósito la verificación del diseño antes de realizar las tareas de flujo de diseño de un ASIC. La estrategia de verificación sigue los siguientes tareas:1) Pruebas simples de instrucciones del ISA RISC-V, 2) Pruebas del SoC, esta incluye la jerarquía de memoria, TLBs -PTW e instrucciones privilegiadas que incluyen los registros de control y estado CSR's, 3) Generación aleatoria de pruebas denominada "*Torture Test*" utilizadas en el entorno de verificación del proyecto LowRISC, y finalmente el arranque del núcleo del Sistema Operativo Linux.

4. Conclusiones

La arquitectura del SoC preDRAC, se construyó con el núcleo de Lagarto Hun, y la infraestructura del SoC de LowRISC, Jerarquía de Memora, TLB's, CSR's, y un bloque controlador para debug en silicio, todo diseñado con celdas estándar de TSMC en tecnología de 65nm vía EUROPRACTCE. Para adaptar el modelo RTL se usaron macros optimizadas de SRAM, para las tablas del Predictor de Saltos, Banco de registros, Memorias Caché, TLB's, y Tablas asociadas. La herramienta de síntesis utilizada es Genus de CADENCE, las etapas de verificación funcional antes de la fabricación, la síntesis física en silicio y las pruebas realizadas al Chip fabricado se realizaron por el grupo coordinado por el Centro de supercomputación de Barcelona (BSC), en Mexico el proyecto Lagarto, como se ha planteado de forma original sigue el mapa de ruta definido desde 2010 y se mantiene la colaboración estrecha con el BSC, la UPC y el IMB-CNM.

Agradecimientos

El proyecto DRAC está financiado por fondos para el desarrollo regional de la Comunidad Europea, en el marco del programa operativo ERDF de Cataluña 2014-2020 y liderado por el BSC, mientras Lagarto es soportado por la Secretaría de Investigación y Posgrado SIP del IPN México y por el CONACyT México

con becas para estudiantes de posgrado del Centro de Investigación en Computacion del IPN.

Referencias

[1] C. Ramírez-Lazo, «Design and implementation of an out of order execution engine of floating point arithmetic operations,» Instituto Politécnico Nacional, Centro de Investigación en Computación, Maestría en Ciencias en Ingeniería de Cómputo | Universidad Politécnica de Cataluña, Departamento de Arquitectura de Computadoras, Master in Innovation and Research in Informatics, Ciudad de México, México | Barcelona, España, 2016.

[2] A. Waterman, Y. Lee and D. A. P. a. K. Asanović, «The RISC-V Instruction Set Manual, Volume I: User-Level ISA, Version 2.1,» 2016.

[3] S. McFarling, «Combining Branch Predictors,» Digital Equipment Corporation, 1993.

[4] G. Mondragón-García, «Diseño de una Jerarquia de Memoria para Sistemas Embebidos,» Maestría en Ciencias en Ingeniería de Cómputo, Ciudad de México, México, 2018.

[5] C. Hernández-Calderón, «SUPERSCALAR PROCESSOR DESIGN FOR EMBEDDED SYSTEMS,» Instituto Politécnico Nacional, Centro de Investigación en Computación, Doctorado en Ciencias de la Computación, Ciudad de México, México, 2021.

[6] J. R. García-Ordaz, «Diseño de un ROB-Distribuido para procesadores superescalares,» IPN-CIC, Maestría en Ciencias en Ingeniería de Cómputio, Ciudad de México, México, 2011.

[7] A. J. Ruíz-Ramirez, «Diseño de una cola de instrucciones de acceso a memoria con emision fuera de orden,» Universidad Politécnica de Cataluña, Departamento de Arquitectura de Computadoras, Master in Innovation and Research in Infiormatics | Instituto Politecnico Nacional, Centro de Investigación en Computación, Maestría en Ciencias en Ingeniería de Cómputo, Barcelona, España | Ciudad de México, México, 2016.

[8] E. J. Martínez-Montes, «Diseño de una unidad de extension multimedia para procesadores RISC,» Instituto Politécnico Nacional, Centro de Investigación en Computación, Maestría en CIencias en Ingeniería de Cómputo | Universidad Politecnica de Cataluña, Departamento de Arquitectura de Computadoras, Master in Innovation and Research in Informatics, Ciudad de México, México | Barcelona, España, 2016.

[9] J. Pavón-Mercado, «Non-conventional Vector Units for Big Data Workloads,» Universidad Politécnica de Cataluña, Departamento de Arquitectura de Computadoras, Master in Innovation and Research in Informatics | Instituto Politécnico Nacional, Centro de Investigación en Computación, Maestría en Ciencias en Ingeniería de Cómputo, Bercelona, España | Ciudad de México, México, 2018.

[10] J. Mendoza-Escobar, «Multitheading RISC-V Implementation for Lagarto Architecture,» Maestria en Ciencias en Ingeniería de Cómputo | Master in Innovation and Research in Informatics, Ciudad de México, México | Barcelona, España, 2020.

[11 J. I. Quiróz-Mercado, «Diseño e implementación de un Bus Local para el Procesador Lagarto I,» Maestría en Ciencias en Ingeniería de Cómputo, Ciudad de México, México, 2017.

[12] D. O. Hernández-Trejo, «Unidad de manejo de Excepciones para procesadores RISC,» IPN-CIC, Maestría en Ciencias en Ingeniería de Cómputo, Ciudad de México, México, 2015.

[13] I. Vargas-Valdivieso, «Sistema de arranque de un SoC RISC-V,» Maestría en Ciencias en Ingeniería de Cómputo, Ciudad de México, México, 2019.

[14] N. I. Leyva-Santes, «Diseño y Simulación de Redes de Interconexión para jerarquia de memoria compartida en microprocesadores multinúcleo, utilizando el núcleo de Lagarto I.,» Mestría en Ciencias en Ingeniería de Cómputo, Ciudad de México, México, 2021.

[15] A. Martínez-Cruz, R. Barrón-Fernández, H. Molina-Lozano, M. A. Ramirez-Salinas, L. A. Villa-Vargas, P.-C. Antonio y Kwang-Ting, «An Automatic Functional Coverage for Digital Systems Through a Binary Particle

Swarm Optimization Algorithm with Reinitialization Mechanism,» *Journal of Electronic Testing,* vol. 33, pp. 431-447, 2017. https://doi.org/10.1007/s10836-017-5665-x

[16] C. Rojas-Morales, «From FPGA to ASIC: A RISC-V Processor Experience,» Master in Innovation and Research in Informatics | Maestría en Ciencias en Ingeniería de Cómputo, Barcelona, España | Ciudad de México, México, 2020.

[17] Barcelona Supercomputing Center (BSC), Universidad Politecnica de Cataluña (UPC), Instituto de Microelectrónica de Barcelona (IMB-CNM), Instituto Politécnico Nacional de México, «An Academic RISC-V Silicon Implementation Based on Open-Source Components,» de *DCIS*, 2020.