ISSN 1870-4069

Procesador *soft-core* de una única instrucción

Fernando Olivera Domingo, Felipe Iturriaga Cortés,
Christian Abraham Almaraz de Horta

Instituto Politécnico Nacional,
Unidad Profesional Interdisciplinaria de Ingeniería Campus Zacatecas (UPIIZ),
Zacatecas, Mexico

{foliverad, fiturriaga}@ipn.mx, crisalmaraz@outlook.com

Resumen. En este trabajo se presenta un procesador *soft-core*, implementado sobre un dispositivo lógico programable FPGA, de tipo OISC (*One Instruction Set Computer*), es decir que cuenta con una única instrucción y que esta es completa Turing. El procesador desarrollado es de 8 bits (aunque se puede modificar fácilmente) y se compara con el procesador de 8 bits de Xilinx (Picoblaze) con el fin de presentar una propuesta sobre sus posibles aplicaciones.

Palabras clave: arquitectura de computadoras, lógica programable, sistemas embebidos.

Single-Instruction Soft-Core Processor

Abstract. This paper presents a processor textit soft-core, implemented on a programmable logic device FPGA, of type OISC (textit One Instruction Set Computer), that is to say that it has a 'unique instruction 'on and that this is complete Turing. The developed processor is 8 bits (although it can be easily modified) and compared with the 8-bit Xilinx processor (Picoblaze) in order to present a proposal about its possible applications.

Keywords: Computer Architecture, Programmable Logic, Embedded Systems.

1. Introducción

Generalmente cuando se utiliza un procesador, este ya esta implementado con circuitos integrados, siendo lo que se denomina un procesador *hard-core*. Sin embargo existen circunstancias donde se necesita mayor flexibilidad para modificar su set de instrucciones o simplemente no existe una arquitectura

comercial adecuada, en esos casos se puede hacer uso de un dispositivo lógico programable, como son las FPGA (*Field Programmable Gate Array*), para sintetizar procesadores mediante lenguaje de descripción de hardware (HDL). A estos dispositivos se les conoce como procesadores *soft-core*. Estos procesadores no son implementados directamente en el silicio por lo que presentan mayores capacidades de configuración a costa de perder velocidad de procesamiento [14]. Debido a los elevados costos de los procesadores *hard-core*, los procesadores *soft-core* se convierten en una alternativa excelente para la investigación, desarrollo y validación de nuevas arquitecturas no comerciales.

2. OISC

En este trabajo se presenta un procesador OISC, es decir, de una sola instrucción. Parecería inútil realizar un procesador así si no consideramos que esta instrucción concreta sea completa Turing [13] (suponiendo un almacenamiento ilimitado y una fiabilidad absoluta), lo que significa que una máquina con solo esta instrucción puede simular una máquina universal de Turing, lo que de manera más coloquial implica que la máquina puede, en principio, hacer cualquier cálculo que cualquier otra computadora es capaz de hacer (en otras palabras, es programable). Obsérvese, sin embargo, que no dice nada sobre el esfuerzo de escribir un programa para la máquina o sobre el tiempo que puede tomar el cálculo.

Los procesadores OISC son también llamados URISC (*Ultimate Reduced Instruction Set Computer*) [8], por llevar al extremo el principio de los procesadores RISC (*Reduced Instruction Set Computer*) de utilizar un set de instrucciones reducido, en este caso compuesto por una sola instrucción. Sin embargo es discutible que los procesadores OISC se puedan incluir dentro de los procesadores RISC o sean una nueva clase con características diferenciadas respecto a los procesadores RISC y CISC [6].

Se pueden separar los procesadores OISC en tres categorías [10]:

- *Transport Triggered Architecture Machines*: Máquinas en las que la computación es un efecto secundario del movimiento de datos (cuentan solo con una instrucción de tipo MOVE)[6].
- *Bit Manipulating Machines*: Máquinas que manipulan bits (o bytes) y saltan incondicionalmente a una dirección especificada por uno de los operandos[9].
- *Arithmethic Based Turing Complete-Machine*: Máquinas que unen la realización de una operación aritmética junto con un salto condicional [6].

La ventaja que presenta este tipo de computadoras es que no requieren un código de operación porque solo tiene una instrucción, lo que simplifica considerablemente el diseo de la CPU.

3. Desarrollo de un procesador OISC en FPGA

En este trabajo se ha desarrollado un procesador OISC del último tipo presentado en la sección anterior, con la instrucción SUBLEQ A B C (SUBtract

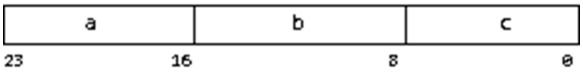

Fig. 1. Formato de la única instrucción del procesador.

and branch if Less or EQual), que significa "resta el valor almacenado en la posición A de la memoria al valor almacenado en la posición B de la memoria, almacena el resultado en la posición B de la memoria y si es menor o igual que cero salta a la dirección de memoria correspondiente al valor almacenado en la posición C de la memoria". Se puede demostrar que está instrucción es Turing completa basándose en el Teorema de Bhm-Jacopini [1] siguiendo la misma linea presentada por Gilreath y Laplante [6].

La máquina presentada en cuestión tiene un tamao de datos de 8 bits, pero las instrucciones son de 24 bits, estando el formato de la instrucción en términos de las posiciones de memoria que direcciona la instrucción (ver figura 1). En este caso se eligió utilizar una arquitectura tipo Harvard con una memoria ROM (asíncrona o síncrona) con un tamao de palabra de 24 bits, que almacena el programa, y una memoria RAM de doble puerto síncrona con un tamao de palabra de 8 bits, que le permite leer dos datos a la vez y escribir uno cuando la seal de habilitación esté activa. La CPU cuenta solo con dos estados pues no necesita decodificar, por lo que la máquina de estados solo tiene un estado donde carga la nueva instrucción y otro donde la ejecuta. No se ha incluido segmentación (*pipeline*) porque la idea principal es realizar un procesador lo más sencillo posible (con un área mínima). Se podría implementar una arquitectura von Neumann, considerando que cada instrucción serían tres posiciones de la memoria RAM (y que por tanto el contador de programa debe ir avanzando 3 posiciones de memoria cada nueva instrucción), sin embargo, aunque se ahorra mucha área en memoria se pierde velocidad del procesador por los accesos a memoria RAM. El procesador no necesita contar con registros, sin embargo es posible tener registros de propósito general o de entrada/salida (I/O) mapeados en memoria. La figura 2 muestra un esquemático de los componentes del procesador.

4. Trabajos relacionados

Desde que fue propuesto el primer procesadores OISC (o URISC) [12] se han desarrollado distintos procesadores OISC como prueba de concepto, pero en muchos casos no son más que simulaciones [3]. También se ha planteado el uso de procesadores OISC en aplicaciones como algoritmos genéticos [5], procesamiento de imágenes [7], computación encriptada [2] [11] o incluso como multi-procesador [10].

Los procesadores OISC, por su sencillez, puede ser útiles en aplicaciones donde se necesita un microcontrolador sencillo integrado con otros circuitos lógicos

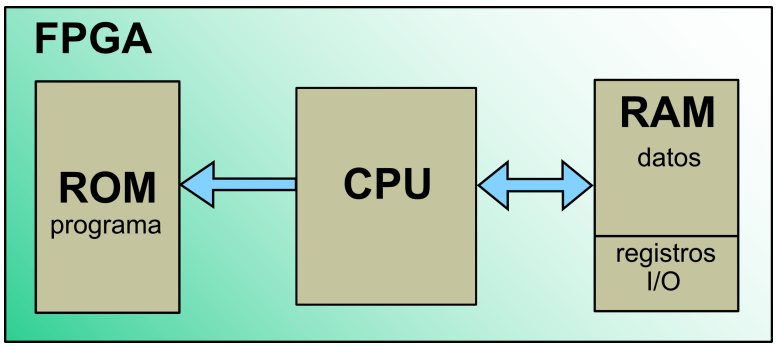

Fig. 2. Esquemático de la arquitectura del procesador.

dentro de una FPGA. Es por ello que el objetivo del trabajo ha sido comparar un procesador OISC basado en la instrucción SUBLEQ con el procesador *soft-core* de 8-bits de Xilinx: Picoblaze [15] para poder encontrar posibles aplicaciones. Creemos que los procesadores OISC pueden adaptarse bien en aplicaciones donde se necesite utilizar pocos recursos.

5. Metodología

Para el desarrollo del procesador se utilizó una metodología Top-Down, donde se realiza una descripción inicial del sistema sin conocer todos los detalles de la implementación y dicho diseo se divide en bloques más sencillos y se va refinando el diseo a medida que se baja en el nivel de diseo, llegando hasta un nivel en que el que el sintetizador mapea el lenguaje de descripción de hardware (HDL) en bloques lógicos. Como lenguaje de descripción de hardware se utilizó VHDL y el procesador se implementó y simuló teniendo como objetivo una tarjeta FPGA Spartan3E XC3S500E de Xilinx.

6. Resultados

Se obtuvo como producto el procesador SUBLEQ diseado. En la tabla 1 se ofrece una comparativa entre el microprocesador *soft-core* SUBLEQ desarrollado y el microprocesador *soft-core* Picoblaze de Xilinx, que es un procesador también de 8 bits, con 16 registros de propósito general y un set de instrucciones (ISA) compuesto por 57 instrucciones.

Se puede observar que la máxima frecuencia de reloj es similar, más aún si consideramos que Picoblaze generalmente no puede trabajar a su frecuencia máxima sin tener algunos problemas de latencia, consiguiéndose en la tarjeta objetivo frecuencias máximas de reloj de entorno a 87 MHz. Sin embargo el área ocupada en la FPGA en el procesador SUBLEQ es de solo de 15 *slices*, 6 veces

Tabla 1. Comparativa entre el procesador desarrollado y el Picoblaze de Xilinx.

Procesador	Bits datos	Bits instrucciones	Máx. frec.	Slices	BRAM	Ciclos/instr.	ISA
Picoblaze	8	18	125.3 MHz	96	1	2	57
SUBLEQ	8	24	101.5 MHz	15	2	2	1

menor que las 96 que requiere Picoblaze. Aunque ambos procesadores necesitan 2 ciclos de reloj por instrucción, hay que recordar que el procesador desarrollado solo tiene una instrucción, por lo que para realizar la misma operación que Picoblaze necesitará realizar dicha instrucción varias veces. Por ejemplo para realizar la instrucción LOAD sX,sY de Picoblaze se necesita realizar la instrucción SUBLEQ varias veces:

SUBLEQ sX sX
SUBLEQ sY Z
SUBLEQ Z sX
SUBLEQ Z Z

Siendo sX la dirección de memoria (que puede ser un registro mapeado en memoria) que recibe el dato que hay en la dirección de memoria sY y Z una dirección de memoria que comienza y termina con un valor nulo. Se omite el tercer operando de la instrucción SUBLEQ porque sería una dirección de memoria con el valor de la posición de la siguiente instrucción, de modo que independientemente del resultado de la substracción la ejecución siga un orden secuencial. De modo que dicha instrucción por ejemplo se ejecutará en 8 ciclos de reloj en el procesador desarrollado, frente a los 2 ciclos de reloj en Picoblaze.

Sin embargo hay que considerar que la sencillez del procesador desarrollado permite implementar 6 procesadores en paralelo en el mismo área de silicio que un único Picoblaze. Es cierto que cada procesador SUBLEQ utiliza dos bloques BRAM (memoria embebida en las FPGA de Xilinx) en vez de uno. Sin embargo utilizando memoria distribuida (memoria implementada en LUTs de la FPGA) para realizar la memoria ROM, todavía se obtiene un procesador de 30 *slices*, 3 veces menos que Picoblaze.

Viendo el pequeo tamao del procesador desarrollado se decidió probar a desarrollar versiones con un tamao de palabra de datos distintos. Inicialmente se trató de dotar a dichos procesadores de una memoria RAM lo más grande posible de acuerdo al tamao de palabra (n): $n \times 2^n - 1$. Con memorias ROM de $3n \times 2^n - 1$. Sin embargo los 20 BRAM con los que cuenta la tarjeta objetivo ya no eran suficientes para implementar un procesador con un tamao de palabra de 15 bits. En las figuras 3 y 4 se muestra la evolución del área utilizada así como de la frecuencia máxima de reloj.

Posteriormente se decidió limitar el tamao de la memoria RAM a $(n \times 2^8 - 1)$, así como de la memoria ROM a$(3n \times 2^8 - 1)$. En las figuras 5 y 6 se muestra la evolución del área utilizada, así como de la frecuencia máxima de reloj.

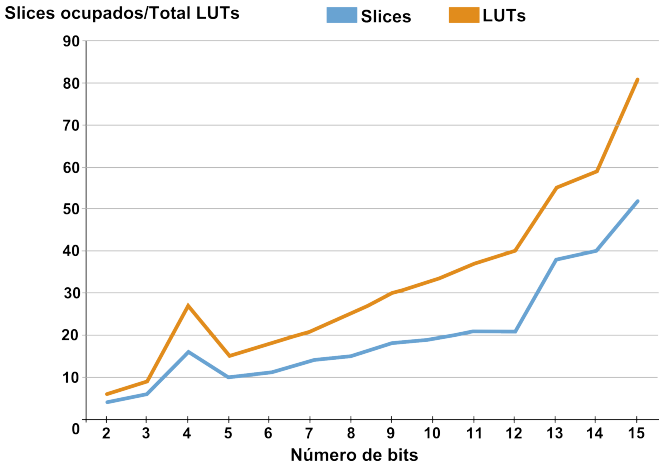

Fig. 3. Evolución del área (en LUTs y *slices*) en función del tamao de palabra del procesador.

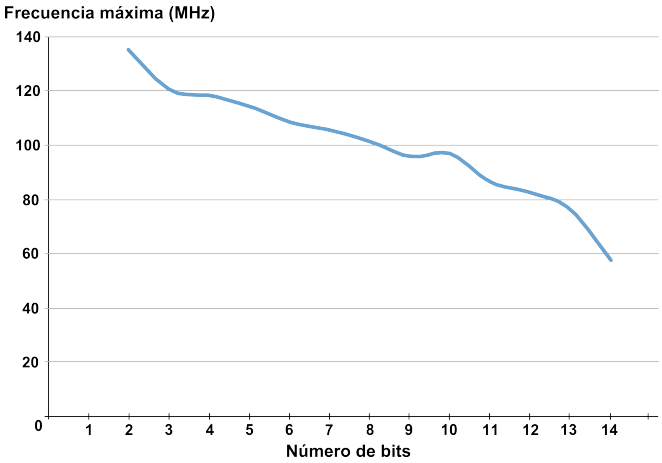

Fig. 4. Evolución de la frecuencia máxima de reloj en función del tamao de palabra del procesador.

Se puede observar que incluso con un tamao de palabra de datos de 128 bits se consigue un procesador de solo 90 *slices* (más pequeo que picoblaze aunque no en bloques BRAM) y una frecuencia máxima de reloj superior a 50 MHz.

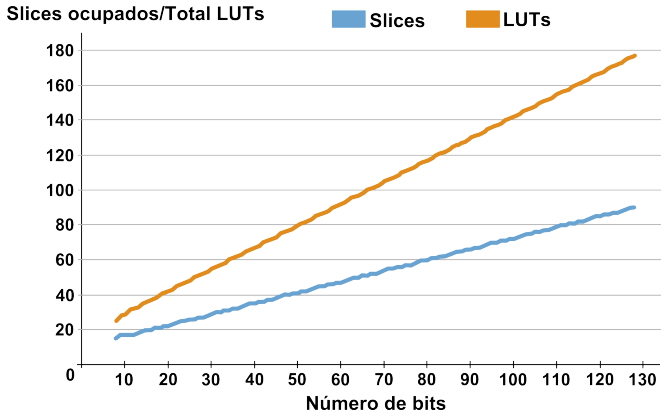

Fig. 5. Evolución del área (en LUTs y *slices*) en función del tamao de palabra del procesador limitando la memoria.

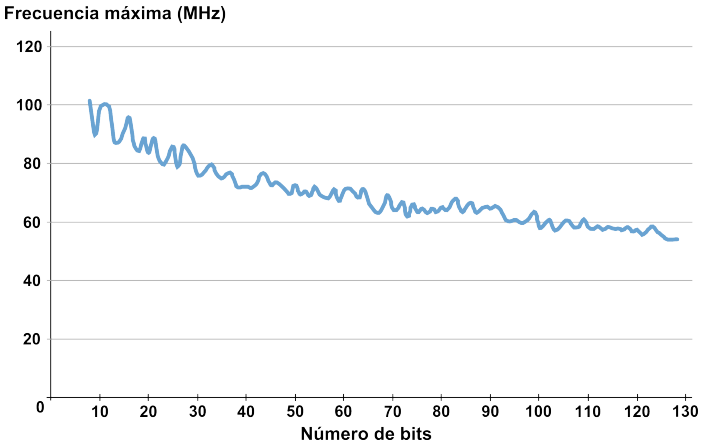

Fig. 6. Evolución de la frecuencia máxima de reloj en función del tamao de palabra del procesador limitando la memoria.

7. Conclusiones y trabajo futuro

Los resultados apuntan a que pueden existir algunas aplicaciones donde el procesador desarrollado pueda tener un buen desempeo. Por un lado parece que en aplicaciones donde limitar el área sea importante, un procesador de este tipo podría funcionar bien; es necesario sin embargo profundizar en el desempeo del procesador SUBLEQ desarrollado, para lo que sería necesario comparar la velocidad de resolución de algunos alg1oritmos tanto en Picoblaze como en el

procesador SUBLEQ. Para ello es necesario contar con herramientas de ensamblado y/o compilación (como las utilizadas en [10]) adaptadas al procesador.

Cabe destacar que limitando el tamao de la memoria se consiguen tamaos de palabra muy altos con pocos recursos. Atendiendo a la taxonomía de Flynn [4] para las arquitecturas paralelas se puede sugerir que una arquitectura de tipo MIMD (en realidad SIMD porque solo existe una instrucción en el procesador desarrollado) como las implementadas en las computadoras VLIW (*Very Long Instruction Word*), también llamadas últimamente EPIC (*Explicitly Parallel Instruction Computers*), podría tener un desarrollo y resultados interesantes en el caso de una computadora SUBLEQ. Aunque una aplicación de ese estilo requeriría esfuerzos adicionales en el compilador para evitar dependencias.

Por otro lado, la sencillez de personalización del procesador desarrollado deja entrever la posibilidad de desarrollar aplicaciones reconfigurables, donde el propio compilador decida en función del software a ejecutar, cuales serán las características del procesador (tamao de la memoria, tamao de palabra ...) y este se configure antes de cargar el software.

Por último decir que es necesario comparar Picoblaze y el procesador SUBLEQ desarrollado en FPGA más potentes como Artix o Kintex.

Agradecimientos. Esta investigación ha sido financiada con el proyecto SIP 20171871 de la Secretaría de Investigación y Posgrado del Instituto Politécnico Nacional.

Referencias

1. Böhm, C., Jacopini, G.: Flow diagrams, turing machines and languages with only two formation rules. Communications of the ACM 9(5), 366–371 (1966)
2. Chatterjee, A., Sengupta, I.: Furisc: Fhe encrypted urisc design. IACR Cryptology ePrint Archive 2015, 699 (2015)
3. Davidar: https://github.com/davidar/subleq (2009), https://github.com/davidar/subleq
4. Flynn, M.J., Rudd, K.W.: Parallel architectures. ACM Computing Surveys (CSUR) 28(1), 67–70 (1996)
5. Gilreath, W., Laplante, P.: A one instruction set architecture for genetic algorithms. In: Biocomputing, pp. 91–113 (2004)
6. Gilreath, W.F., Laplante, P.A.: Computer architecture: A minimalist perspective, vol. 730. Springer Science & Business Media (2003)
7. Laplante, P., Gilreath, W.: One instruction set computers for image processing. Journal of VLSI signal processing systems for signal, image and video technology 38(1), 45–61 (2004)
8. Mavaddat, F., Parhami, B.: Urisc: the ultimate reduced instruction set computer. International Journal of Electrical Engineering Education 25(4), 327–334 (1988)
9. Mazonka, O.: Bit copying: The ultimate computational simplicity. Complex Systems Journal 19, N3, 263–285 (2011)
10. Mazonka, O., Kolodin, A.: A simple multi-processor computer based on subleq. arXiv preprint arXiv:1106.2593 (2011)

11. Mazonka, O., Tsoutsos, N.G., Maniatakos, M.: Cryptoleq: A heterogeneous abstract machine for encrypted and unencrypted computation. IEEE Transactions on Information Forensics and Security 11(9), 2123–2138 (2016)
12. van der Poel, W.L.: The logical principles of some simple computers. Ph.D. thesis, University of Amsterdam (1956)
13. Turing, A.M.: On computable numbers, with an application to the entscheidungsproblem. Proceedings of the London mathematical society 2(1), 230–265 (1937)
14. Xilinx: Microblaze Processor Reference Guide (2006), https://www.xilinx.com/support/documentation/sw_manuals/mb_ref_guide.pdf
15. Xilinx: Picoblaze 8-bit Embedded Microcontroller User Guide (2011), https://www.xilinx.com/support/documentation/ip_documentationug129.pdf