DESCRIPCIONES VHDL PARA CALCULAR LA FUNCIÓN RECÍPROCA EN PRECISIÓN SIMPLE

Oscar Norberto Bria - Javier O. Giacomantone

LIDI, Facultad de Informática - CeTAD, Facultad de Ingeniería

Universidad Nacional de La Plata

Palabras Claves: Evaluación de Funciones Elementales, Función Recíproca, IEEE-754, VHDL.

RESUMEN

En este trabajo presentamos una descripción VHDL de un método para el cómputo de la función recíproca en formato IEEE 754 de precisión simple. El algoritmo básico es el ATA (Add-Table lookup-Add) propuesto por Wong y Goto. Después de analizar y comparar el error del algoritmo original con el de las posibles alternativas, se sugieren mejoras que permiten reducir el tamaño de las tablas requeridas. Las descripciones VHDL implementadas forman parte de una biblioteca de algoritmos para el cálculo de funciones elementales.

1. INTRODUCCION

Son muchas las aplicaciones actuales que demandan cómputo aritmético intensivo: criptografía, corrección de errores, multimedia, telefonía celular y electrónica de consumo masivo en general. Por esta razón es de esperar un próspero crecimiento de la aritmética de computadoras en los próximos años [10].

Un enfoque serio de este campo requiere de un estudio de la teoría subyacente y de los diseños de hardware reales. Además se complementa con la implementación de los algoritmos y la evaluación de su desempeño. De esta forma puede desarrollarse una biblioteca de diseños adecuados para la implementación en hardware o en software de procesadores empotrados [4].

Las implementaciones de algoritmos para el cómputo de funciones se caracterizan por la representación numérica que soportan (incluyendo la exactitud del redondeo) y la latencia de cómputo asociada con los recursos involucrados. Normalmente existe una relación de compromiso entre la latencia y los recursos utilizados, y ciertamente existe una relación de compromiso entre estos dos factores y la representación soportada para datos y resultados [8]. Las distintas implementaciones resolverán con variada eficiencia estas relaciones de compromiso atendiendo a las necesidades numéricas, a las limitaciones de recursos y tecnológicas, y a las características intrínsecas de los algoritmos. El análisis de las relaciones anteriormente citadas puede resolverse

en algunos casos en forma analítica, pero la mayoría de las veces requiere de la implementación o simulación del algoritmo para poder comparar fehacientemente las arquitecturas propuestas (e incluso atender a las características peculiares de un dado soporte tecnológico).

La implementación de funciones elementales comienza con el estudio o desarrollo de los algoritmos con vistas a la validación de sus comportamientos numéricos. En este trabajo se presentan descripciones en VHDL de un algoritmo para el cómputo de la función recíproca de números representados en formato IEEE 754. Los algoritmos que utilizamos para el cálculo de la inversa están basados en el método propuesto por Wong y Goto [13], [14] para el cálculo de funciones elementales en precisión simple. Es un método tabular donde el rango de las direcciones está determinado por el rango de fragmentos de los operandos, más que por el rango de los operandos en sí mismos. Las descripciones VHDL están dirigidas a facilitar la comprobación del correcto funcionamiento numérico de los algoritmos bajo prueba, mediante procesos de simulación y análisis de resultados.

2. El METODO ATA

El formato IEEE 754 [7] para la representación de un número en precisión simple utiliza 4 bytes. Un bit se destina al signo, 8 bits se reservan para la representación sesgada del exponente (el sesgo es 127), y 23 bits se destinan a la representación de la parte fraccionaria del significante. Se asume que el significante tiene un bit a la izquierda del punto (o coma) binaria, que a su vez es el bit implícito de la representación normalizada.

En lo que sigue describiremos en forma breve el método de Wong y Goto, ignorando el tratamiento del signo y del exponente. Además haremos las siguientes distinciones:

- Llamaremos fracción a los 23 bits del significante que tienen representación explícita en la norma.
- Llamaremos mantisa a la representación completa del significante, es decir incluyendo el bit implícito y el punto (o coma) binaria.

Sea X un número en punto fijo que representa la mantisa de la representación de un número de punto flotante en el formato IEEE de precisión simple. Este número tiene una longitud de 24 bits, de los cuales el primer bit representa la parte entera y los restantes representan la fracción. Al ser una representación normalizada, el bit que representa la parte entera está fijo en 1. La mantisa entonces está restringida a valores en el intervalo $[1, 2)$.
Dividamos al número X en los siguientes segmentos:

$$X = x_0 + \lambda x_1 + \lambda^2 x_2 + \lambda^3 x_3 \tag{1}$$

donde $\lambda = 2^{-6}$, y $x_i < 1$ (con $i = 1, 2, 3$) son segmentos de 6 bits de longitud.

Sea f la función elemental que queremos computar (en este caso la función recíproca), y $f^{(n)}$ la derivada n-ésima de f . El desarrollo en serie de Taylor de la función dada se escribe de la siguiente manera:

$$f(X) = \sum_{n=0}^{\infty} \frac{f^{(n)}(x_0 + \lambda x_1)}{n!} (\lambda^2 x_2 + \lambda^3 x_3)^n \tag{2}$$

Expandiendo la ecuación (2) para los primeros términos se obtiene la ecuación (3), donde $|\zeta|$es el residuo, que es dependiente de la magnitud de las derivadas.

Se puede probar ver [14] que $\tilde{f}(X)$ en la ecuación (4) aproxima a $f(X)$ de la ecuación (3) hasta $\mathcal{O}(\lambda^5)$.

La formulación anterior sirve de base para el método ATA propuesto por Wong y Goto para aproximar funciones elementales dentro de un rango limitado. ATA es el acrónimo de Add-Table lookup-Add (sumar, leer tabla, sumar), que son las tres operaciones básicas que se realizan en secuencia para el cálculo de $\tilde{f}(X)$:

1. Sumar o restar para obtener las 4 diferencias centrales de 13 bits, $x_0 + \lambda x_1 + \lambda x_2$, $x_0 + \lambda x_1 - \lambda x_2$, $x_0 + \lambda x_1 + \lambda x_3$,$x_0 + \lambda x_1 - \lambda x_3$ (el número $x_0 + \lambda x_1$ no requiere suma).

2.1 Leer los cinco valores de f de una sola tabla, o de tablas paralelas para aumentar la velocidad. Estas tablas tienen un rango de direcciones de 13 bits.

2.2 Leer el valor del último término, que es una función de x_0 y de x_2, de una tabla diferente. Esta tabla tiene un rango de direcciones de 13 bits.

$$\begin{aligned} f(X) &= f(x_0 + \lambda x_1) + f^{(1)}(x_0 + \lambda x_1)(\lambda^2 x_2 + \lambda^3 x_3) + \\ &\quad \frac{f^{(2)}(x_0 + \lambda x_1)}{2}(\lambda^4 x_2^2 + 2\lambda^5 x_2 x_3 + \lambda^6 x_3^2) + \zeta \end{aligned} \tag{3}$$

$$\begin{aligned} \tilde{f}(X) &= f(x_0 + \lambda x_1) + \frac{\lambda}{2}\{f(x_0 + \lambda x_1 + \lambda x_2) - f(x_0 + \lambda x_1 - \lambda x_2)\} + \\ &\quad \frac{\lambda^2}{2}\{f(x_0 + \lambda x_1 + \lambda x_3) - f(x_0 + \lambda x_1 - \lambda x_3)\} + \lambda^4 \left\{ \frac{x_2^2}{2} f^{(2)}(x_0) - \frac{x_2^3}{6} f^{(3)}(x_0) \right\} \end{aligned} \tag{4}$$

3. Sumar seis operandos, los provenientes de las seis lecturas de las tablas.

Los números almacenados en las tablas provienen de representaciones fraccionarias de doble precisión (52 bits) que pueden ser acotadas a un número menor de bits.

3. DESCRIPCIÓN VHDL DE ATA

VHDL [5], [1], es un lenguaje de descripción de hardware que es estándar IEEE desde 1987. Permite descripciones tanto en el dominio estructural como en el comportamental, y varios niveles de abstracción desde el nivel lógico hasta el nivel de sistema. Una de sus virtudes es su capacidad para hacer convivir de forma natural descripciones mixtas multinivel: distintos estilos y niveles de abstracción para las distintas partes de un diseño [9].

Debido a la creciente complejidad de los sistemas digitales, es frecuentemente deseable empezar el diseño a altos niveles de abstracción, es decir a nivel puramente algorítmico. Las

Nombre	Característica	Nro.
`Adder`	Sumador de 13 bits	2
`Subtr`	Restador de 13 bits	2
`Ti`	Tabla de 13 bits de direc.	6
`Tree`	Sumador de 6 operandos	1

Cuadro 1: Detalle de Componentes

transformaciones que son necesarias para llegar a la implementación final se llevan luego a cabo o bien manualmente o bien con herramientas de síntesis automáticas.

El uso del estándar VHDL para la validación de algoritmos numéricos se fundamenta en su facilidad para especificar y operar con tipos compuestos, y la mencionada posibilidad de usar distintos niveles de abstracción cuando es necesario.

Además, a través del depurado de los modelos puramente funcionales, se puede iniciar un proceso de refinamiento gradual de la descripción del circuito hasta alcanzar un modelo arquitectural-RT que sea sintetizable mediante procesos automáticos guiados por el diseñador. Esta primera fase de refinamiento gradual para pasar de una descripción funcional a un de nivel RT está ahora en vías de automatización y se la conoce como síntesis comportamental [11].

Por otra parte la comunicación entre las descripciones algorítmicas y otros componentes VHDL de igual o más bajo nivel se puede llevar a cabo usando constructores VHDL. Para reducir el tiempo de diseño requerido para la inserción de estos componentes en un diseño más complejo se siguen desarrollando herramientas para analizar la calidad de las descripciones VHDL, modelos de componentes reusables para síntesis comportamental [2], analizadores de la reusabilidad de diseños existentes [3], [12], extensiones del mismo VHDL (OO-VHDL) , y estándares para síntesis [6].

El algoritmo ATA puede describirse en forma estructural en VHDL como se muestra en la Figura 1.

En la descripción de la arquitectura VHDL se ha puesto de manifiesto la secuencialidad implícita, haciendo uso de la identación. Además se han repetido las tablas para aumentar la velocidad. En la descripción se tienen los componentes básicos que muestra el Cuadro 1.

Los componentes pueden implementarse de varias maneras. No se presenta aquí una descripción detallada de las posibles alternativas.

Haciendo un análisis detallado puede determinarse que el rango de direcciones de las tablas es en realidad el mostrado por el Cuadro 2.

Mediante análisis pueden determinarse los datos mostrados en el Cuadro 3 respecto de la precisión de las tablas.

4. VERIFICACIÓN DEL DISEÑO

Particularizando las tablas puede simularse el diseño anterior para la implementación de la función recíproca de mantisas en el rango $[1, 2)$, que es el rango de las mantisas de números normalizados en IEEE-754.

Tablas	Rango de Direc.
T0 y T5	2^{12}
T1 y T3	$2^{12} + 2^{6}$
T2 y T4	$2^{12} + 2^{5}$

Cuadro 2: Rangos de Direcciones para Tablas

Tabla/s	Bits de Precisión
T0	30
T1 y T4	25
T3 y T5	19
T5	6

Cuadro 3: Bits de Precisión para Tablas

Para la verificación del correcto funcionamiento del algoritmo escribimos un programa que examina exhaustivamente todas las posibles entradas (esto es posible porque las entradas son de 24 bits de longitud, haciendo por lo tanto viable el cómputo). Analizando los resultados puede verse que el máximo error absoluto se presenta en el bit ubicado en la posición 27,3:

$$\text{Posición} = -\log_2(\text{Error Absoluto Máximo}) \qquad (5)$$

Para la mantisa donde se produce el error absoluto máximo se puede escribir el conjunto de relaciones (6).

$$\begin{aligned} f(\ 1.\ 000000\ 111111\ 111111\ 000000) &= \texttt{1. F 8 1 F F E 4 2 E 3 E 7 0} \\ f(\ 1.\ 000000\ 111111\ 111111\ 000000) &= \texttt{1. F 8 1 F F E 0 F C 0 8 1 F} \\ f(\cdot) - f(\cdot) &= \texttt{0. 0 0 0 0 0 0 3 3 2 3 6 5 1} \end{aligned} \qquad (6)$$

La Figura 2 ilustra el comportamiento del error en torno del valor de la mantisa donde se produce el máximo valor del mismo.

5. ANÁLISIS DEL ERROR Y VARIACIONES DEL ALGORITMO

La Figura 3 representa el máximo error absoluto cada cien mil números para todo el rango de la mantisa. La curva superior representa dicho error para el algoritmo original. Se ve que el error

mejora casi linealmente aumentando un bit cada aproximadamente 1,6 millones de números, después de pasar por el mínimo para la mantisa número 131.040 (el rango de la mantisa es 2^{23}).

Este comportamiento del error sugiere que se investiguen variaciones del algoritmo orientadas a reducir los recursos en la implementación de las tablas y de los sumadores para valores altos de la fracción. Algunas alternativas son las siguientes:

1. La variante más inmediata es la eliminación del factor de corrección, con el consiguiente ahorro de la tabla T5 en forma total y la eliminación de un operando en Tree.

2. Otra alternativa es introducir modificaciones en la forma en que se realiza la fragmentación de la fracción antes de calcular las diferencias centrales. Manteniendo las definiciones dadas, la siguiente es una posible partición:

$$X = x_0 + 2(\lambda x_1 + \lambda^2 x_2 + \lambda^3 x_3) \tag{7}$$

3. Se pueden combinar las dos alternativas anteriores.

La Figura 3 compara el algoritmo de Wong y Goto original con las tres alternativas sugeridas. En dicha figura, la curva superior representa, como se ha mencionado, los resultados para el algoritmo original. Las curvas siguientes representan los errores para las variantes 2), 1) y 3) respectivamente.

Una posible implementación es usar el algoritmo original para las mantisas hasta la mitad del rango, y de ahí hasta el final del rango utilizar la segmentación sugerida en 2). Del análisis de esta combinación surge que se logra un ahorro de alrededor del 25 % en la dimensión de todas las tablas como muestra el Cuadro 4 (comparar con el Cuadro 2). Esta solución ha sido evaluada exhaustivamente con resultados satisfactorios.

6. CONCLUSIONES

El método de Wong y Goto constituye una alternativa a los métodos polinomiales e iterativos para la imple-

mentación de funciones elementales, como así también a los métodos tabulares puros.

Tablas	**Rango de Direc.**
`T0 y T5`	$2^{11} + 2^{10}$
`T1, T2, T3 y T4`	$2^{11} + 2^{10} + 2^6$

Cuadro 4: Nuevos Rangos de Direcciones

La variante introducida en el algoritmo permite un ahorro significativo de recursos sin afectar la latencia ni comprometer la precisión.

Dadas las descripciones VHDL ya verificadas se podrán comenzar a evaluar distintas alternativas de implementación de los componentes del sistema y se podrá proceder a un refinamiento gradual para pasar a una descripción de nivel RT (registros de transferencia), es decir se podrá hacer la síntesis comportamental como prerrequisito para el mapeo tecnológico.

7. REFERENCIAS

[1] J. Bhasker, *A Guide to VHDL Syntax*, Prentice Hall, 1995.

[2] C. Costi, D. Miller, "A VHDL analysis environment for design reuse," *Virtual Components Design and Reuse*, editado por R.Seepold, N.Matínez, Kluwer, 2001.

[3] C. Hansen, O. Bringmann. W. Rosenstiel, "A VHDL reusable component model for fixed abstraction level simulation and behavioral synthesis," *Virtual Components Design and Reuse*, editado por R. Seepold, N. Martínez, Kluwer, 2001.

[4] J. O. Giacomantone, "Tradeoffs in Arithmetic Architectures for CORDIC Algorithm Design," *Anales VII Workshop IBERCHIP*, Montevideo, Uruguay, 2001.

[5] IEEE, *VHDL Language Reference Manual*, ANSI/IEEE Standard 1076-1994, 1993.

[6] IEEE, *Standard for VHDL Register Transfer Level Synthesis*, IEEE Standard 1076.6, 1999.

[7] W. Kahan, "Lecture notes on the status of IEEE-754, "accesible electrónicamente, `htttp://http.cs.berkeley.edu/~wkahan/`, 1996.

[8] J.-M. Muller, *Elementary Functions - Algorithms and Implementation*, Birkhäuser, 1997.

[9] L. Terés, Y. Torroja, S. Olcoz, E. Villar, *VHDL Lenguaje Estándar de Diseño Electrónico*, McGraw-Hill, 1997.

[10] B. Parhami, *Computer Arithmetic. Algorithms and Hardware Designs*, Oxford University Press, 2000.

[11] Y. Torroja, F. Machado, F. Casado, E. de la Torre, T. Riesgo, J. Uceda, "ARDID: a tool and a model for the quality analysis of VHDL based design," *Virtual Components Design and Reuse*, editado por R.Seepold, N.Matínez, Kluwer, 2001.

[12] H. A.Villagarcía, O. N. Bria, "Diseño de bloques IP: Programabilidad y Reutilización," *Anales de WICC2001*, San Luis, Argentina, Mayo 2001.

[13] W. F. Wong, E. Goto, "Fast hardware-based algorithms for elementary function computations using rectangular multipliers," *IEEE Transactions on Computer*, Marzo 1994.

[14] W. F. Wong, E. Goto, "Fast evaluation of the elmentary function in single precision," *IEEE Transactions on Computer*, Marzo 1995.

```
Entity ATA is
    Port(
           x: in  bit_vector(24-1 downto 0);
           y: out bit_vector(24-1 downto 0);
        );
end ATA;

Architecture Structure of ATA is

.... -- declaration of objects and components

begin

                                        a0 <= x0 & x1;  -- x0 & x1 & x2 & x3 = x
                                        a1 <= Adder(a0, d*x2); -- d = 2^(-6)
                                        a2 <= Subtr(a0, d*x2);
                                        a3 <= Adder(a0, d*x3);
                                        a4 <= Subtr(a0, d*x3);


                          b0 <= T0(a0);  -- Tables
                          b1 <= T1(a1);
                          b2 <= T2(a2);
                          b3 <= T3(a3);
                          b4 <= T4(a4);
                          b5 <= T5(x0, x2);

y <= Tree( b0, -- y = f(x)
           b1,
           b2,
           b3,
           b4,
           b5);

end structure;
```

Figura 1: Diseño VHDL Estructural para ATA.

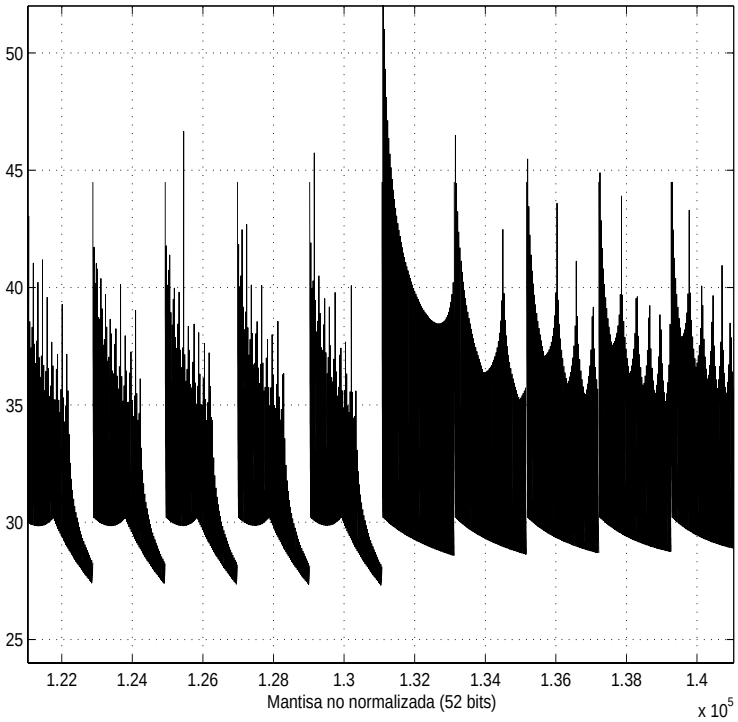

Figura 2: Error en torno al Máximo en Wong-Goto

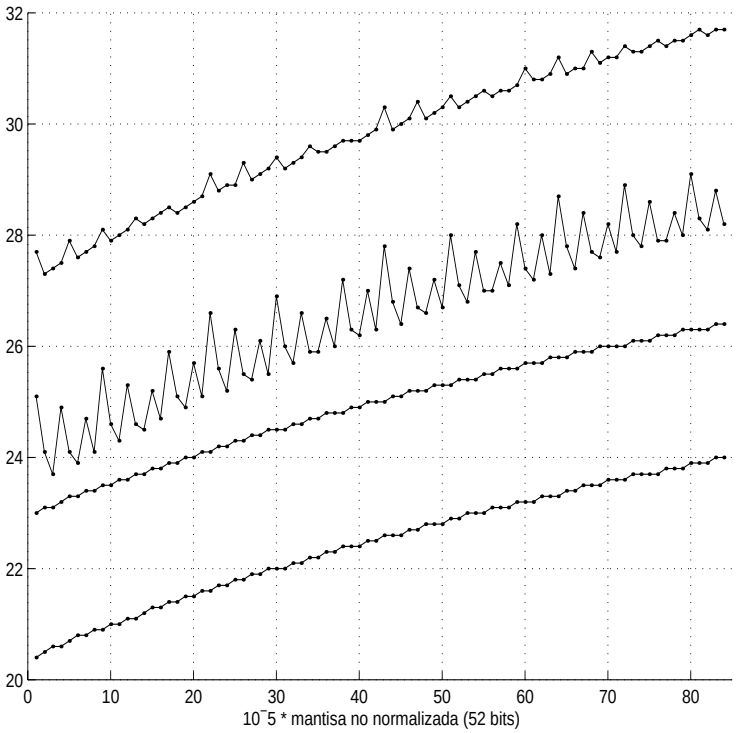

Figura 3: Error en Wong-Goto y sus Variaciones