Diseño de arquitecturas sistólicas para la evaluación de polinomios

Reyes Martínez Héctor Alexis[1], Rivera Domínguez Jorge[1], Ortega Cisneros Susana[1]
[1]Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, Unidad Guadalajara,
Av. Del Bosque 1145, 45019 Zapopan, Jalisco, México.
hector.reyes@cinvestav.mx jorge.rivera@cinvestav.mx susana.ortega@cinvestav.mx

Resumen

En este trabajo se trata el problema de la aproximación de funciones determinísticas del tipo polinomial y trascendental mediante el uso de FPGA. Para ello se propone utilizar los polinomios de Chebyshev como método de aproximación de las funciones matemáticas deseadas, que generan los correspondientes coeficientes de Chebyshev mismos que permitirán la aproximación en algún rango deseado. Para evaluar dichos polinomios, se utilizó el método de recurrencia de Clenshaw, ya que debido a las características de los polinomios de Chebyshev es posible utilizar este método para evaluar la función aproximada. Dichos datos son ingresados a una arquitectura sistólica en cascada que permite en cada iteración realizar cálculos hasta obtener el resultado deseado. Los resultados por simulación mostraron que dependiendo de qué función se desea aproximar y sus características, se obtiene distintos errores. Por último, se aproximó la función $\cos(x)$, $\operatorname{atan}(x)$ y $\sqrt{x}$, las cuales tienen errores aceptables en cada aproximación.

Palabras Clave: arquitectura sistólica, evaluación polinomial, método de Clenshaw, polinomios de Chebyshev.

Abstract

This paper deals with the problem of approximation of deterministic functions of the polynomial and transcendental type using FPGAs. For this purpose, it is proposed to use Chebyshev polynomials as a method of approximation of the desired mathematical functions, which generate the corresponding Chebyshev coefficients that will allow the approximation in some desired range. To evaluate these polynomials, the Clenshaw recurrence method was used, since due to the characteristics of the Chebyshev polynomials it is possible to use this method to evaluate the approximated function. These data are entered into a cascade systolic architecture that allows in each iteration to perform calculations until the desired result is obtained. The simulation results showed that depending on the function to be approximated and its characteristics, different errors are obtained. Finally, the function cos(x),atan(x) and √x were approximated, which have acceptable errors in each approximation.

Key words: systolic architecture, polynomial evaluation, Clenshaw method, Chebyshev polynomials.

Artículo arbitrado	
Recibido: 15 de octubre de 2022	Aceptado: 07 de noviembre de 2022

Introducción

El cálculo matemático en *hardware* es un tema que ha sido investigado a lo largo de la historia de la electrónica. Conforme las necesidades del ser humano avanzaban en conjunto con la tecnología, fue necesario realizar cálculos matemáticos más complejos, ya no solo sumas/restas o productos/divisiones, sino algunos cosenos, senos, tangentes e incluso logaritmos. Este tipo de matemática necesitaría formas especiales para su cálculo.

Algunas opciones han sido investigadas e implementadas en *hardware* para realizar el cálculo de alguna función determinística particular, por ejemplo, las *Look-Up Tables* (LUT, por sus siglas en inglés) son una forma de realizar aproximaciones. Estas consisten en espacios de memoria con valores precargados (Ebrahimi et al, 2020) de la función deseada. El problema que tienen las LUT es que ocupan una cantidad de *hardware* demasiado grande dependiendo de la resolución deseada; además de que solamente podrían aproximar una sola función de acuerdo al diseño realizado. Otro método utilizado es el algoritmo CORDIC, el cual fue desarrollado por Jack E. Volder en 1959 (Kumar, 2019). Este consiste en la rotación de vectores y el uso de LUT (Kumar, 2019), por lo que entre más compleja sea la función, aumenta la latencia, además de que fue optimizado para CPU de baja complejidad y que no tengan módulo dedicado a resolver multiplicaciones. Respecto a implementaciones en FPGA para la aproximación de funciones matemáticas, en Rekha et al. (2018) se implementa un FPGA para aproximar una función exponencial utilizando el bloque de propiedad intelectual CORDIC; en Langhammer et al. (2013), de la misma manera para funciones elementales en punto flotante con un valor de entrada restringido. Finalmente, en Gilliland et al. (2016) se presenta otra implementación en FPGA del algoritmo CORDIC para funciones elementales.

La mayoría de estos métodos enfrentan el problema de la latencia, la exactitud de la aproximación, el tamaño de *hardware* y su adaptabilidad (capacidad de resolver múltiples funciones matemáticas), por lo que es necesario un método en el cual se pueda adaptar a diversas funciones matemáticas sin cambiarlo, y que pueda ser desarrollado para modificar los problemas mencionados de acuerdo con las necesidades de la aplicación. Por lo anterior, el método de Chebyshev evaluado con el método de Clenshaw es una alternativa a estas problemáticas implementadas en FPGA.

Metodología

Arquitecturas sistólicas

Las arquitecturas sistólicas (también llamadas arreglos sistólicos) son una red de elementos de procesamiento (EP). Estos EP regularmente bombean datos desde y hacia otros EP, manteniendo un flujo regular de datos (Cho et al, 2020). Pueden ser implementadas como un procesador en conjunto con un host, que cuando requiere hacer ciertos cálculos matemáticos puede enviar los datos a la arquitectura sistólica y esta los realiza con sus EP. El resultado final es devuelto al host. Los EP calculan y guardan datos independientemente de los demás; cada uno es un procesador independiente y podría contener algunos registros y ALU (Cho et al, 2020).

Hay algunas propiedades que las arquitecturas sistólicas tienen. Estos arreglos son redes de *hardware* con las siguientes características (Chiper et al, 2022):

- Estructuras lineales o de dos dimensiones conectadas con sus vecinos más cercanos.
- I/O externas del host solamente en las orillas de los arreglos.
- Comunicación unidireccional entre celdas.
- Comunicaciones locales solamente.

Sin embargo, con la evolución de la tecnología y *hardware*, se han disminuido las exigencias de

cumplir totalmente esas características para incrementar la utilidad de los arreglos sistólicos. Con base en estas últimas, se puede diseñar arquitecturas sistólicas modulares, regulares y eficientes para aplicaciones DSP.

Polinomios de Chebyshev

Se basa en aproximar una función mediante la sumatoria de la multiplicación de los polinomios por su respectivo coeficiente como se muestra a continuación (Rivera et al, 2018):

$$\bar{f}(x) = \sum_{k=0}^{N-1} c_k T_k(x) \tag{1}$$

Donde $\bar{f}(x)$ es la función que se desea aproximar, c_k los coeficientes de Chebyshev y $T_k(x)$ son los polinomios de Chebyshev. Estos se pueden obtener con la siguiente fórmula de recurrencia (Rivera et al, 2018):

$$\begin{gathered} T_{k+1} = 2xT_k - T_{k-1} \\ \forall k = 2,3,\dots,N-1. \end{gathered} \tag{2}$$

Mientras que los coeficientes de Chebyshev se obtienen con la siguiente fórmula (Rivera et al, 2018):

$$\begin{gathered} c_j = \frac{2}{N}\sum_{k=1}^{N} f(x_k)T_j(x_k) \\ j = 0,1,\dots,N-1 \end{gathered} \tag{3}$$

Donde $f(x_k)$ es la función a aproximar evaluada en los nodos de Chebyshev y $T_j(x_k)$ es el polinomio *j* de Chebyshev evaluado en el nodo *k* de Chebyshev. Los nodos se pueden calcular mediante la siguiente fórmula (Rivera et al, 2018):

$$\begin{gathered} x_k = \cos\frac{\pi\left(k-\frac{1}{2}\right)}{N} \\ k = 1,2,3,\dots,N \end{gathered} \tag{4}$$

En donde N es la cantidad de coeficientes que se desean en la aproximación. En el caso de requerir deun mapeo de los datos de entrada se utiliza la siguiente expresión:

$$u = \frac{b-a}{2}x + \frac{a+b}{2} \tag{5}$$

Donde *a* es el límite inferior y *b* el límite superior deseados en la aproximación, ya que en principio la aproximación es para el dominio en $x \in [-1,1]$.

Fórmula de recurrencia de Clenshaw

Es una forma elegante y eficiente para evaluar una sumatoria de coeficientes que multiplican funciones que obedecen a una fórmula recurrente. Aplicado a la serie de Chebyshev, la recurrencia es (He et al, 2017):

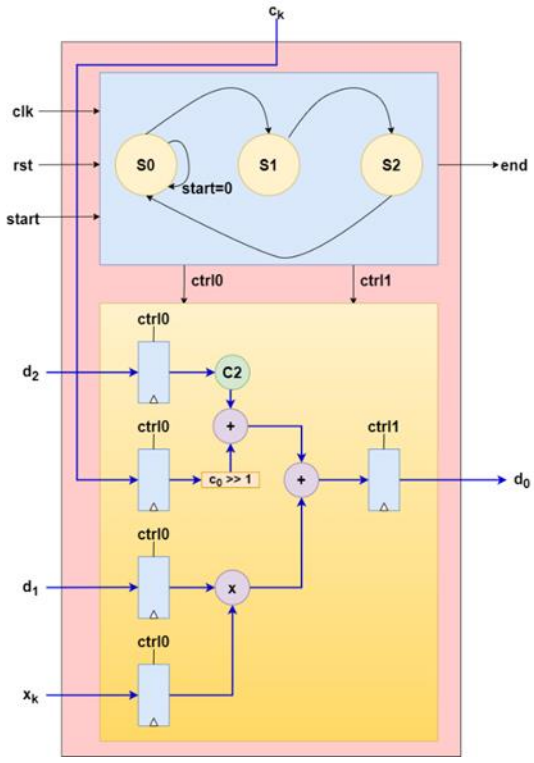
$$\begin{gathered} d_{m+1} \equiv d_m \equiv 0 \\ d_j = 2xd_{j+1} - d_{j+2} + c_j \\ f(x) \equiv d_0 = xd_1 - d_2 + 0.5c_0 \\ j = m-1, m-2, \dots, 1 \end{gathered} \tag{6}$$

Donde d_j son los coeficientes de Clenshaw, d_{m+1} y d_m son coeficientes para las condiciones iniciales, c_j los coeficientes de Chebyshev y *x* el punto donde se desea conocer el valor de la función aproximada.

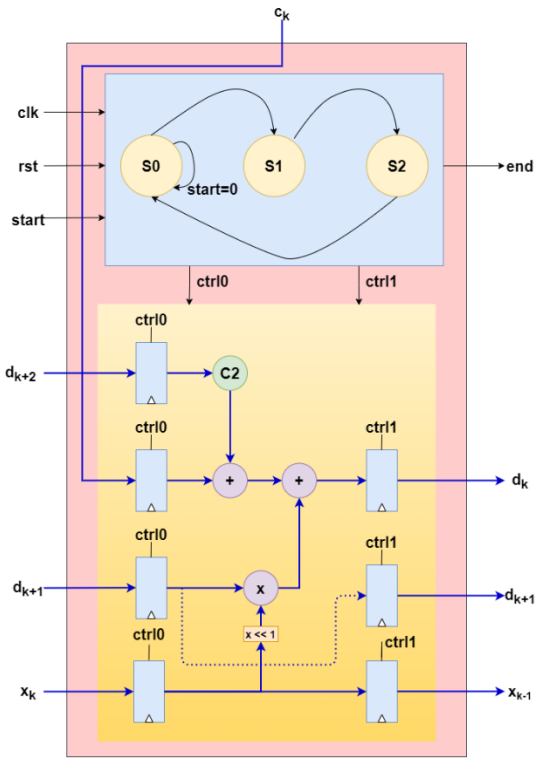
Diseño de arquitectura sistólica para el método de Clenshaw

Al analizar las ecuaciones que se requieren para los polinomios de Chebyshev y su implementación con Clenshaw, es necesario diseñar elementos de procesamiento adecuados para implementarlos en *hardware* como una arquitectura sistólica. Dado que suponemos que los coeficientes de Chebyshev están

pre-cargados en memoria, al analizar la ecuación (6) se observa que las condiciones iniciales son cero, indica como calcular los coeficientes d_j para $j = m-1, m-2, \ldots, 1$ y como es la última iteración del sistema. A partir de estas ecuaciones se concluye que se requieren dos elementos de procesamiento:



a)



b)

Fig. 1. Celdas sistólicas. (a) Celda sistólica para $j = m-2, m-1, \ldots, 1$. (b) Celda sistólica para $j = 0$. Fuente: elaboración propia.

1. Elemento de procesamiento para las iteraciones $j = m-1, m-2, \ldots, 1$ que permita tener las condiciones iniciales para la primera.
2. Elemento de procesamiento para la última iteración $d_0 = xd_1 - d_2 + 0.5c_0$ que solamente realice dicha operación.

Además, tal como se indican las características de las arquitecturas sistólicas (Chiper et al, 2022), se requiere que exista una coordinación síncrona entre las celdas. Para ello, a la lógica combinacional que realizan las operaciones matemáticas se le agregarán registros y una máquina de estados que coordine el flujo de los datos a través de la celda; también su comunicación con las celdas vecinas, con lo que se logre una mejor sincronización de datos y la disponibilidad de interactuar con otro tipo de *hardware*, por ejemplo un procesador host que

mediante señales de control, indique el flujo de coeficientes y resultados.

Desarrollo de soluciones

La Figura 1a muestra la celda diseñada para la **j-ésima** iteración $d_j = 2xd_{j+1} - d_{j+2} + c_j$ con $j = m-1, m-2, \ldots, 1$, en la cual puede observar el cómo se integró una máquina de estados para el control de flujo de datos a través de la lógica combinacional para realizar las operaciones. De la misma forma, se diseñó la segunda celda sistólica para la última iteración $d_0 = f(x) = xd_1 - d_2 + 0.5c_0$ la cual se muestra en la Figura 1b. Estas celdas están conectadas en serie de acuerdo con el número de coeficientes deseados en la aproximación.

La Figura 2 muestra el diagrama de señales de una celda unitaria. Ambas celdas están constituidas por la misma máquina de estados que coordina el flujo de datos. Se puede observar que la señal de reloj está presente en la imagen, y que al activar la señal de control *start* la máquina de estados de esa celda se inicializa. Los datos de entrada deben estar establecidos antes de la señal de control *start*. Después se observa que durante la transición de la máquina de estados la parte combinacional hace los cálculos necesarios y son reflejados en la señal de "salida". Hasta que la señal de control *end* por parte de la máquina de estados sea activada, es cuando se asegura un dato válido a la salida, por lo que esto permite la interconexión de varias celdas para formar un arreglo sistólico con señales de control *start-end*, para que sepan cuándo leer sus datos de entrada y cuándo depositarlos en las salidas. De esta forma se logra un arreglo sistólico coordinado de forma síncrona con sus respectivas dos señales de control: *start y end*.

El número de celdas necesarias para la aproximación es el mismo de acuerdo con el número de coeficientes con que se desee realizar la aproximación. Por ejemplo, en la Figura 2 se observa que una sola celda tiene una latencia de dos ciclos de reloj; debido a los registros de la celda se genera un *pipeline* para optimizar las operaciones de cada celda, por lo que habrá una latencia natural para obtener el primer dato (en ciclos de reloj):

$$latencia_{nat} = 2n \tag{7}$$

Donde *n* es la cantidad de coeficientes deseados en la aproximación. A partir del primer dato, se obtiene los demás datos cada tres ciclos de reloj debido al *pipeline*. Esta implementación fue realizada en el FPGA Cyclone IV EP4CE115F29C7 de Intel.

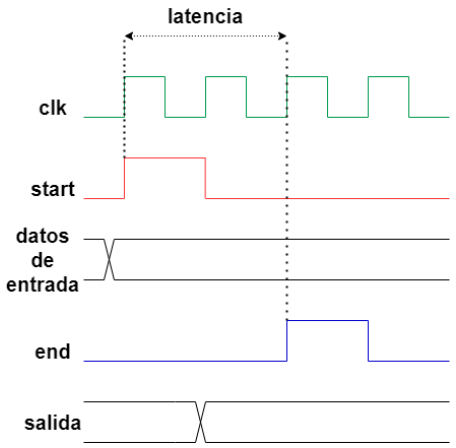


Fig. 2. Diagrama de señales de una celda unitaria.

Resultados

En la Figura 3 se muestra una comparación entre el método de Taylor (Parhi, 2016) y el método de Chebyshev para aproximar la función $f(x) = \cos(x)$, en donde se puede apreciar un mejor desempeño por el método de Chebyshev. Este método realiza aproximaciones en varios puntos de la función dependiendo del grado de aproximación, a diferencia de Taylor que solamente aproxima alrededor de un punto.

En este reporte se muestran tres funciones aproximadas con el método propuesto: $\cos(x)$, $\text{atan}(x)$ y $\sqrt{x}$.

La Figura 4a muestra la aproximación de $\cos(x)$, en la que se tuvo un error máximo de 0.055% aproximadamente a lo largo de toda la función; por

lo que es muy buena aproximación en el cálculo del rango de valores de $x \in [-\pi, \pi]$. Estas funciones fueron seleccionadas debido a que son comúnmente utilizadas en trabajos relacionados con aproximación de funciones.

La Figura 4b muestra la aproximación de $\mathrm{atan}(x)$, en esta aproximación se tuvo un error máximo de 1.6%. La Figura 4c muestra la aproximación de $\sqrt{x}$, la cual tuvo un error máximo de 11% en $x = 0$. Este fenómeno es debido a que existe una asíntota en este valor de x. En general, el error de aproximación es $O(\epsilon^n)$, con $\epsilon \ll 1$. Con esto se entiende que si se desea disminuir el error de aproximación se tiene que aumentar el grado del polinomio n, pero por la ecuación (7) el tiempo de aproximación, esto es, la latencia, crecerá.

En los tres casos, el *hardware* utilizado del FPGA fue del 16%.

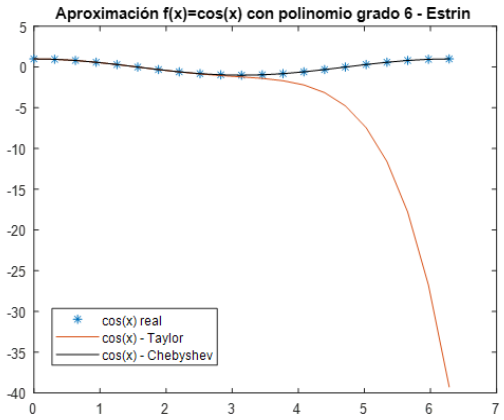
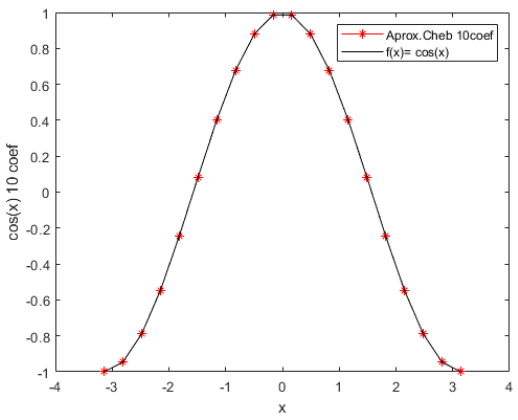


Fig. 3. Aproximación de la función $f(x) = \cos(x)$ con el método de Taylor y método de Chebyshev. Fuente: elaboración propia.

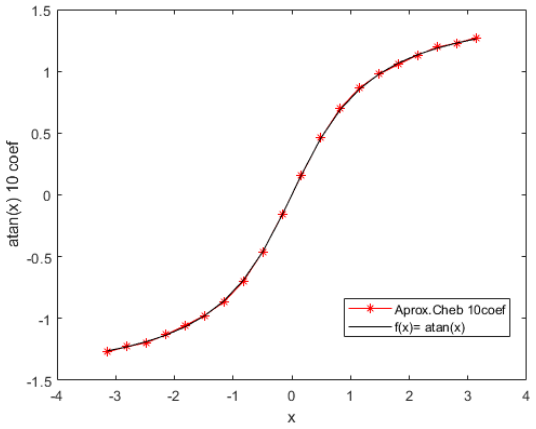
Discusión

La aproximación de funciones matemáticas mediante los polinomios de Chebyshev usando el método de Clenshaw ha mostrado ser muy útil con las arquitecturas sistólicas implementadas en una FPGA. El uso de señales de control mediante una máquina de

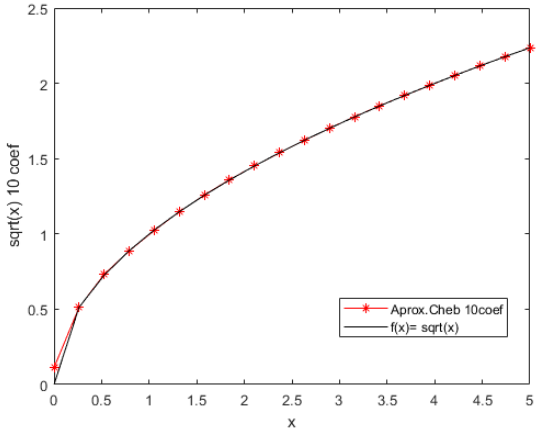
estados permite una mejor coordinación de datos y la posibilidad de interconectar el sistema con un procesador *host*, además de ser utilizado para realizar cálculos complicados. Esto funciona como un acelerador de *hardware* que de forma experimentada fue implementado en punto fijo. Como trabajo futuro, resultaría interesante extender estos resultados con un formato de punto flotante.



(a)



(b)



(c)

Fig. 4. Funciones aproximadas mediante FPGA en un arreglo sistólico de 10 coeficientes de Chebyshev mediante el método de Clenshaw. a) $f(x) = \cos(x)$, b) $f(x) = \text{atan}(x)$, c) $f(x) = \sqrt{x}$. Fuente: Elaboración propia.

Referencias

Ebrahimi, M., Sadeghi, R., & Navabi, Z. (2020). LUT input reordering to reduce aging impact on FPGA LUTs. *IEEE transactions on computers. Institute of Electrical and Electronics Engineers*, *69*(10), 1500–1506. https://doi.org/10.1109/tc.2020.2974955

Kumar, P. A. (2019). FPGA implementation of the trigonometric functions using the CORDIC algorithm. *2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS)*, 894–900.

Chiper, D. F., Cracan, A., & Andries, V.-D. (2022). An overview of systolic arrays for forward and inverse discrete sine transforms and their exploitation in view of an improved approach. *Electronics*, *11*(15), 2416. https://doi.org/10.3390/electronics11152416

Cho, K., Lee, I., Lim, H., & Kang, S. (2020). Efficient systolic-array redundancy architecture for offline/online repair. *Electronics*, *9*(2), 338. https://doi.org/10.3390/electronics9020338

J. Rivera Domínguez & I. E. Dueñas. (2018). Aproximación de funciones trigonométricas con polinomios de Chebyshev para aplicaciones digitales. 2do Coloquio en electrónica analógica y digital.

Parhi, Keshab & Liu, Yin. (2016). Computing Arithmetic Functions Using Stochastic Logic by Series Expansion. IEEE Transactions on Emerging Topics in Computing Vol. 7. DOI: 10.1109/TETC.2016.2618750.

Gilliland, Spenser & Saniie, Jafar & Martinez Vallina, Fernando (2016). Implementation of elementary functions for FPGA compute accelerators. IEEE International Conference on Electro Information Technology. DOI: 10.1109/EIT.2016.7535235.

Langhammer, Martin & Pasca, Bogdan. (2013). Elementary Function Implementation with optimized sub range polynomial evaluation. IEEE 21st Annual International Symposium on Field-Programmable Custom Computing Machines. DOI: 10.1109/FCCM.2013.30.

Rekha, R. Menon & Karunakara P. (2018). FPGA implementation of exponential function using cordic IP core for extended input range. 3rd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT). DOI: 10.1109/RTEICT42901.2018.9012611.