# THIẾT KẾ PHẦN CỨNG TÍNH TOÁN CÁC HÀM TOÁN HỌC DỰA TRÊN XẤP XỈ TUYẾN TÍNH HAI MỨC

Sái Văn Thuận*, Hoàng Văn Phúc, Trần Văn Khẩn

***Tóm tắt:*** *Bài báo này trình bày một phương pháp tính toán các hàm toán học phổ biến dựa trên hai mức xấp xỉ. Trong mức xấp xỉ thứ nhất, hàm được xấp xỉ bằng phương pháp xấp xỉ phân đoạn tuyến tính đều. Sau đó, ở mức xấp xỉ thứ hai các hàm lỗi do mức xấp xỉ thứ nhất sẽ được xấp xỉ bởi phương pháp phân đoạn tuyến tính có nội suy đối xứng nhằm giảm thiểu độ phức tạp của phần cứng. Dựa trên phương pháp đề xuất, kiến trúc phần cứng để tính toán các hàm toán học điển hình được thiết kế và thực thi. Các kết quả thực thi cho thấy kiến trúc phần cứng đề xuất đạt được hiệu quả về tốc độ.*

**Từ khóa**: Số học máy tính; Ước lượng hàm; Xấp xỉ phân đoạn tuyến tính; Xấp xỉ đa thức.

## I. MỞ ĐẦU

Các hàm toán học như hàm sin, logarithm, hàm mũ, hàm nghịch đảo … được sử dụng rộng rãi trong nhiều lĩnh vực như truyền thông, đồ họa máy tính, khoa học tính toán và xử lý tín hiệu số. Thực thi các hàm toán học nói trên có thể thực hiện bằng các chương trình phần mềm. Tuy nhiên, tính toán các hàm toán học bằng phần mềm sẽ có tốc độ tính toán chậm. Vì vậy, nhiều nghiên cứu đã tập trung thực hiện tính toán các hàm toán học bằng các phần cứng chuyên dụng.

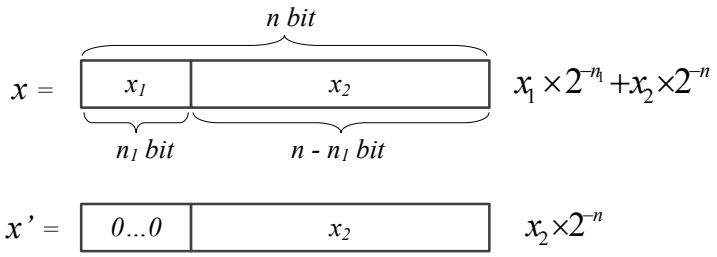
Một số các phương pháp khác nhau đã được nghiên cứu và đề xuất để thực thi phần cứng tính toán các hàm toán học. Các phương pháp này bao gồm: thuật toán CORDIC [1] , xấp xỉ đa thức [2],[3],[4], xấp xỉ hữu tỷ [5] và các phương pháp dựa trên bảng [6],[7],[8],[9]. Thuật toán CORDIC dựa trên kiến trúc lặp, do đó, có độ giữ chậm lớn nên không phù hợp với các ứng dụng thời gian thực. Phương pháp xấp xỉ hữu tỷ có độ chính xác khá cao, tuy nhiên, đòi hỏi độ phức tạp phần cứng cao. Ngày nay, với sự phát triển của công nghệ mạch tích hợp cho phép dung lượng bộ nhớ lớn thì các phương pháp dựa trên bảng được sử dụng khá phổ biến. Tuy nhiên, các phương pháp dựa trên bảng có một nhược điểm là khi độ rộng toán hạng đầu vào lớn đòi hỏi dung lượng lớn, điều đó đòi hỏi nhiều tài nguyên phần cứng cũng như khó khăn trong thực hiện của các công cụ tổng hợp.

Mặt khác, với nhu cầu ngày càng tăng nhanh của các thiết bị điện tử thông minh không dây, các thiết bị cầm tay, điện thoại di động,với nhiều ứng dụng đòi hỏi hệ thống xử lý tín hiệu số tốc độ cao ngày càng đặt ra cấp thiết. Trong khi đó, tính toán các hàm toán học là những thao tác quan trọng trong các ứng dụng xử lý tín hiệu số, việc tính toán các hàm toán học chiếm một phần lớn tài nguyên phần cứng cũng như quyết định đến tốc độ xử lý của các bộ xử lý tín hiệu số. Vì vậy, đòi hỏi có các lõi tính toán không những hiệu quả về tài nguyên mà còn yêu cầu tốc độ xử lý cao.

Trong bài báo này, chúng tôi đề xuất một phương pháp thực thi phần cứng cho tính toán các hàm toán học được sử dụng phổ biến trong các ứng dụng xử lý tín hiệu số. Phương pháp của chúng tôi dựa trên xấp xỉ các hàm toán học bằng hai mức theo phương pháp xấp xỉ tuyến tính phân đoạn. Các kết quả thực thi cho thấy phương pháp đề xuất đạt được hiệu quả về tốc độ thực thi.

## II. ĐỀ XUẤT PHƯƠNG PHÁP XẤP XỈ

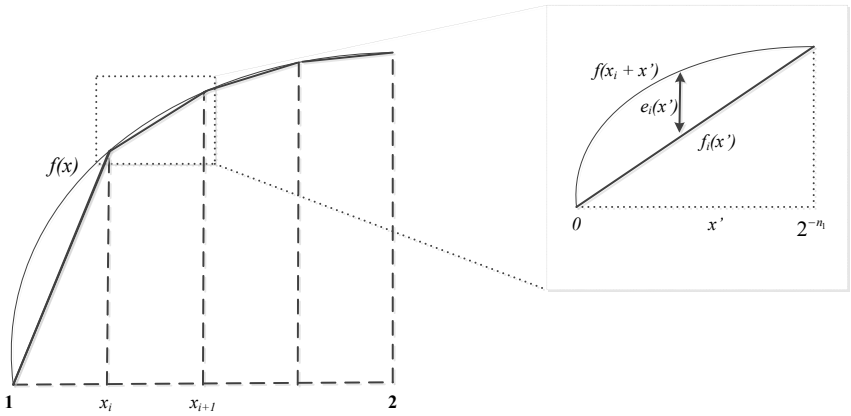
Trong thiết kế của chúng tôi hàm được xấp xỉ bởi hai mức, mức đầu tiên sử dụng xấp xỉ phân đoạn tuyến tính hàm và mức thứ hai thực hiện xấp xỉ lỗi của mức đầu tiên sử dụng phương pháp xấp xỉ phân đoạn tuyến tính đối xứng.




***Hình 1.*** *Phân chia biến đầu vào thành hai phần.*

Trong mức đầu tiên đầu vào $x$ có $n$ bit phần thập phân được chia thành hai phần $x_1$ và $x_2$ với độ dài bit tương ứng là $n_1$ và $n-n_1$ như biểu diễn trên hình 1. Trong bước xấp xỉ đầu tiên khoảng giá trị của $x$ được chia thành $2^{n_1}$ đoạn dựa trên $x_1$ như thấy trên hình 2 cho trường hợp $n_1=2$ và khoảng ban đầu là $[1,2)$. Với đoạn thứ $i$, $[x_i, x_{i+1})$ hai giá trị $f(x_i)$ và $f(x_{i+1})$ được sử dụng để xấp xỉ ở mức thứ nhất để xấp xỉ hàm bằng hàm $f(x')$, ở đây giá trị $x'$ nhận giá trị trong khoảng $[0,\ 2^{-n_1})$. Hàm xấp xỉ mức 1 cho phân đoạn thứ $i$ có dạng như công thức (1).

$$f_i(x') = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i} \cdot x' + f(x_i) \tag{1}$$



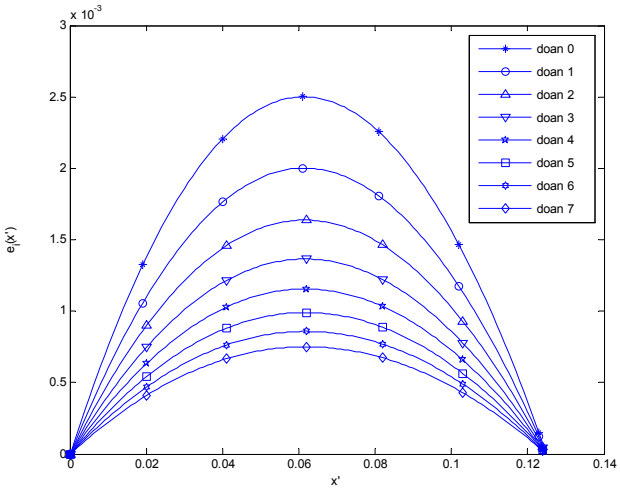

***Hình 2.*** *Xấp xỉ mức thứ nhất của hàm $f(x)$ trong đoạn thứ $i$ $(n_1=2)$.*

Như mô tả trên hình 2, xấp xỉ ở mức 1 sẽ tạo ra lỗi xấp xỉ là độ chênh lệch giữa hàm xấp xỉ ban đầu và hàm thực. Độ chênh lệch này là một hàm lỗi $e_i(x')$, $i=0,1,...,2^{n_1}$ có biểu diễn như công thức (2).

$$e_i(x') = f(x_i + x') - f_i(x'), \qquad 0 \le x' < 2^{-n_1} \tag{2}$$

Trong tầng xấp xỉ thứ hai, các hàm chênh lệch $e_i(x')$ được xấp xỉ bằng phương pháp xấp xỉ phân đoạn tuyến tính sử dụng nội suy đối xứng. Có thể thấy rằng các hàm $e_i(x')$ có dạng như các hàm parabol. Ví dụ, các hàm chênh lệch của hàm $\log_2(x)$ có dạng như trên hình 3 với trường hợp xấp xỉ ban đầu của tám đoạn, tức là $n_1=3$. Các hàm này có dạng parabol vì nó được tạo ra bởi phép trừ của hàm ban đầu với hàm xấp xỉ tuyến tính nên nó sẽ

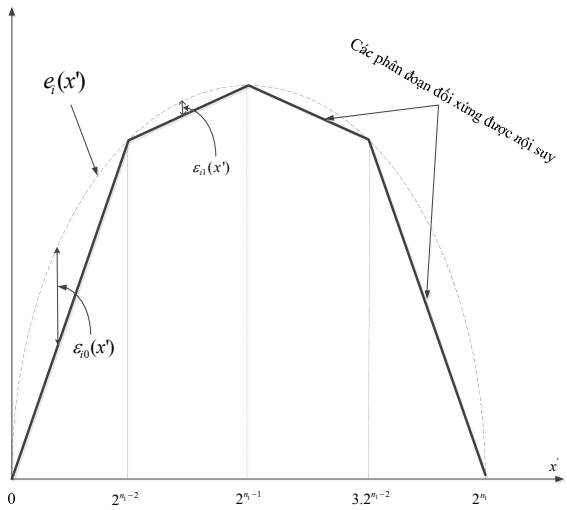
là biểu diễn của các số hạng bậc cao còn lại. Vì vậy, các hàm chênh lệch này có dạng đối xứng qua trục $x' = 2^{-n_1-1}$. Do đó, nếu ta chỉ xấp xỉ nửa khoảng giá trị của $x' \in [0, 2^{-n_1-1})$ bằng các đoạn tuyến tính, nửa còn lại là các đoạn được nội suy đối xứng qua trục $x' = 2^{-n_1-1}$ thì sai số xấp xỉ là không thay đổi. Điều này hứa hẹn cho phép thực thi phần cứng sẽ đơn giản hơn vì khi đó chỉ sử dụng một hoặc một số bit có trọng số lớn nhất (MSB) của dữ liệu đầu vào để nội suy ra các đoạn đối xứng ở nửa còn lại trong khoảng giá trị của $x'$. Hình 4 mô tả xấp xỉ ở mức thứ hai cho hàm $e_i(x')$ với hai cặp đoạn đối xứng.




***Hình 3.*** *Hàm lỗi do xấp xỉ mức 1 đối với hàm* $\log_2(x)$ *với* $n_1 = 3$.

Trong phương pháp đề xuất, nửa khoảng giá trị của $x'$, $x' \in [0, 2^{-n_1-1})$, được chia thành $s$ đoạn con, trong mỗi đoạn con thứ $j, j = 0,1,\ldots,s-1$, hàm lỗi $e_i(x')$ được xấp xỉ bằng một đoạn tuyến tính có các hệ số là $a_{ij}$ và $b_{ij}$, khi đó, lỗi xấp xỉ có dạng như công thức (3).

$$\varepsilon_{ij}(x') = e_i(x') - (a_{ij} \times x' + b_{ij}) \tag{3}$$




***Hình 4***. *Xấp xỉ hàm hàm* $e_i(x')$ *bằng hai cặp đoạn đối xứng.*

Lỗi cực đại cho xấp xỉ trên đoạn $j$ ký hiệu là $\varepsilon_{ij\max}$

$$\varepsilon_{ij\max} = \max\{\varepsilon_{ij}(x')\} \tag{4}$$

Hơn nữa, để đơn giản phần cứng thực thi các hệ số $a_{ij}$ và $b_{ij}$ được ấn định có dạng như công thức (5), trong đó, các hệ số $a_{ij}$ là tổng các số lũy thừa, khi đó, có thể thực thi bằng các phép dịch và cộng tránh việc sử dụng mạch nhân phức tạp.

$$\begin{aligned} a_{ij} &= \sum 2^{-N} \quad ; \; N \in \mathbb{N} \\ b_{ij} &= B \times 2^{-(n-n_1)} \quad ; \; B \in \mathbb{N} \end{aligned} \tag{5}$$

Thuật toán đề xuất tìm các giá trị hệ số $a_{ij}$ và $b_{ij}$ sao cho $\varepsilon_{ij\max}$ nhận giá trị cực tiểu trong đoạn thứ $j$. Thuật toán tìm các hệ số tối ưu được mô tả như bảng 1. Bài báo sử dụng một chương trình Matlab thực hiện thuật toán trên để tính toán các hệ số $a_{ij}$ và $b_{ij}$ tối ưu cho các hàm $\log_2(x)$, $\sin(x)$, $1/x$, $\sqrt{x}$, $1/\sqrt{x}$ và $2^x$ với các khoảng giá trị đầu vào được chuẩn hóa như thể hiện trên bảng 2 và tương ứng với xấp xỉ mức 1 gồm 8 đoạn ($n_1 = 3$) và ở xấp xỉ mức 2 sử dụng xấp xỉ phân đoạn nội suy đối xứng trong đó mỗi hàm $e_i(x')$ được xấp xỉ bởi hai cặp đoạn đối xứng. Bảng 3 thể hiện kết quả các hệ số $a_{ij}$ và $b_{ij}$ tối ưu cho hàm $\log_2(x)$.

**Bảng 1.** *Thuật toán tìm hệ số tối ưu của xấp xỉ mức 2.*

| | |
|---|---|
| 1: | BEGIN: Khởi tạo mảng giá trị $a_{ij}$<br>Khởi tạo mảng giá trị $b_{ij}$<br>Khởi tạo đoạn giá trị $j$ |
| 2: | for i = 1: length($a_{ij}$)<br>for j = 1:length($b_{ij}$)<br>for k = 1:length($x'$)<br>tính $\varepsilon_{ij}(x')$ theo công thức (3)<br>end<br>end<br>end |
| 3: | for i = 1: length($a_{ij}$)<br>for j = 1:length($b_{ij}$)<br>tính $\varepsilon_{ij\max} = \max(\varepsilon_{ij}(x'))$<br>end<br>end |
| 4: | Tính $\min(\varepsilon_{ij\max})$ |
| 5: | END: Xuất giá trị a, b tương ứng với $\min(\varepsilon_{ij\max})$ |

***Bảng 2.*** *Khoảng giá trị đầu vào chuẩn hóa của các hàm.*

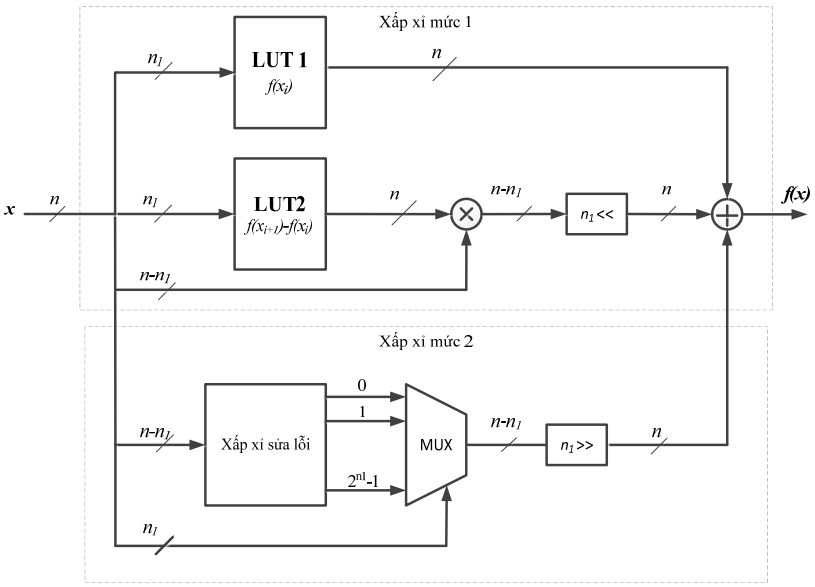
| Hàm | $\log_2(x)$ | $\sin(x)$ | $1/x$ | $\sqrt{x}$ | $1/\sqrt{x}$ | $2^x$ |
|---|---|---|---|---|---|---|
| **Khoảng giá trị đầu vào chuẩn hóa** | [1,2) | [0, 1) | [1, 2) | [1, 2) | [1, 2) | [0, 1) |

***Bảng 3**. Các hệ số $a_{ij}$ và $b_{ij}$ tối ưu ở mức xấp xỉ thứ 2 của hàm $\log_2(x)$.*

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| $e_0(x')$ | $a_{00}$ | $b_{00}$ | $a_{01}$ | $b_{01}$ | $e_4(x')$ | $a_{40}$ | $b_{40}$ | $a_{41}$ | $b_{41}$ |
| | $2^{-4}$ | $58.2^{-20}$ | $2^{-5}$ | $791.2^{-20}$ | | $2^{-5}$ | 0 | $2^{-7}+2^{-10}$ | $672.2^{-20}$ |
| $e_1(x')$ | $a_{10}$ | $b_{10}$ | $a_{11}$ | $b_{11}$ | $e_5(x')$ | $a_{50}$ | $b_{50}$ | $a_{51}$ | $b_{51}$ |
| | $2^{-5}+2^{-6}+2^{-9}$ | $72.2^{-20}$ | $2^{-6}+2^{-9}$ | $1024.2^{-20}$ | | $2^{-6}+2^{-7}$ | $40.2^{-20}$ | $2^{-7}$ | $561.2^{-20}$ |
| $e_2(x')$ | $a_{20}$ | $b_{20}$ | $a_{21}$ | $b_{21}$ | $e_6(x')$ | $a_{60}$ | $b_{60}$ | $a_{61}$ | $b_{61}$ |
| | $2^{-5}+2^{-7}$ | $65.2^{-20}$ | $2^{-7}+2^{-8}$ | $986.2^{-20}$ | | $2^{-6}$ | $88.2^{-20}$ | $2^{-7}$ | $427.2^{-20}$ |
| $e_3(x')$ | $a_{30}$ | $b_{30}$ | $a_{31}$ | $b_{31}$ | $e_7(x')$ | $a_{70}$ | $b_{70}$ | $a_{71}$ | $b_{71}$ |
| | $2^{-5}+2^{-9}$ | $48.2^{-20}$ | $2^{-7}+2^{-9}+2^{-10}$ | $775.2^{-20}$ | | $2^{-6}$ | $51.2^{-20}$ | $2^{-8}$ | $512.2^{-20}$ |

## III. KIẾN TRÚC PHẦN CỨNG VÀ KẾT QUẢ THỰC THI

Kiến trúc tổng quát thực thi các hàm toán học của phương pháp đề xuất có dạng như hình 5. Trong mức xấp xỉ thứ nhất thực hiện tính toán hàm $f_i(x')$ như công thức (1). Trong đó, các giá trị $f(x_i)$ và $f(x_{i+1})-f(x_i)$ được lưu vào trong các bảng tra (LUT: Lookup Table) được sử dụng để tính toán hàm $f_i(x')$. Phép chia cho $x_{i+1}-x_i$ trong công thức (1) được thay thế bởi một phép dịch bởi vì nó là một số lũy thừa của 2. Ở mức xấp xỉ thứ 2, thực hiện xấp xỉ các hàm lỗi bằng phương pháp xấp xỉ phân đoạn tuyến tính có đối xứng. Khối xấp xỉ sửa lỗi sẽ bao gồm $2^{n_1}$ khối xấp xỉ tương ứng với $2^{n_1}$ hàm $e_i(x')$.




***Hình 5.** Kiến trúc phần cứng tổng quát.*

Chúng tôi đã thực thi kiến trúc đề xuất để tính toán các hàm $\log_2(x)$, $\sin(x)$, $1/x$, $\sqrt{x}$, $1/\sqrt{x}$ và $2^x$ với định dạng 23 bit thập phân dữ liệu đầu vào và đầu ra. Về nguyên tắc khi tăng số phân đoạn ở mức xấp xỉ thứ nhất ($n_1$ lớn) thì sẽ đạt được độ chính xác cao hơn, tuy nhiên, điều đó sẽ dẫn đến tăng độ phức tạp phần cứng. Vì vậy, để dung hòa giữa độ phức tạp phần cứng và độ chính xác nhận được chúng tôi thực thi với xấp xỉ mức 1 với 8 phân đoạn ($n_1=3$). Ngoài ra, trong mức xấp xỉ thứ hai, mỗi

hàm lỗi $e_i(x')$ được xấp xỉ bởi phân đoạn tuyến tính của hai cặp đoạn đối xứng. Các kiến trúc đề xuất đã được mô hình trên ngôn ngữ VHDL và thực thi trên thiết bị FPGA Xilinx Virtex6 với công cụ tổng hợp Xilinx ISE 12.4 Suite. Kết quả thực thi trên FPGA của các kiến trúc đề xuất thể hiện trên bảng 4. Bảng 5 thể hiện kết quả tổng hợp bằng Synopsys Design Compiler cho các hàm khác nhau của phương pháp đề xuất với mục đích tối thiểu tài nguyên trên công nghệ CMOS 90 nm. Phương pháp đề xuất cũng được so sánh với kết quả trong [9] với cùng định dạng 23 bit thập phân của dữ liệu đầu vào và đầu ra. Phương pháp đề xuất đã đạt được tốc độ cao hơn đáng kể so với phương pháp trong [9]. Tuy nhiên, tài nguyên chiếm lớn hơn so với kết quả trong [9].

***Bảng 4.*** *Kết quả thực thi của các kiến trúc đề xuất trên FPGA Xilinx Virtex6.*

| Hàm | $\log_2(x)$ | $\sin(x)$ | $1/x$ | $\sqrt{x}$ | $1/\sqrt{x}$ | $2^x$ |
|---|---|---|---|---|---|---|
| Slices | 789 | 755 | 1005 | 897 | 894 | 885 |
| Delay(ns) | 6.79 | 6.76 | 7.28 | 6.78 | 7.33 | 7.73 |
| ADP($\times 10^3$) | 5.42 | 5.10 | 7.32 | 6.08 | 6.55 | 6.84 |

***Bảng 5.*** *Kết quả tổng hợp trên Synopsys Design Compiler.*

| Hàm | Đề xuất | | Trong [9] | |
|---|---|---|---|---|
| | Diện tích ($\mu m^2$) | Độ trễ (ns) | Diện tích ($\mu m^2$) | Độ trễ (ns) |
| $\log_2(x)$ | 26695 | 3.98 | 24023 | 12.69 |
| $\sin(x)$ | 24452 | 3.86 | 22890 | 13.08 |
| $1/x$ | 32736 | 3.99 | 24137 | 12.63 |
| $\sqrt{x}$ | 27213 | 3.83 | 16043 | 12.03 |
| $1/\sqrt{x}$ | 28752 | 4.00 | 22317 | 12.83 |
| $2^x$ | 30887 | 3.86 | 18913 | 12.16 |

## IV. PHÂN TÍCH LỖI

Để đánh giá độ chính xác của phương pháp đề xuất, chúng tôi phân tích lỗi gây ra bởi các thành phần phần cứng trong sơ đồ thực thi và tác động của chúng tới đầu ra. Các lỗi này bao gồm lỗi lượng tử do độ rộng bit hạn chế của các từ nhớ trong ROM (ký hiệu là $\varepsilon_{ROM}$), lỗi gây ra bởi bộ nhân rút gọn ($\varepsilon_{MUL}$), lỗi xấp xỉ do xấp xỉ mức hai ($\varepsilon_{apx2}$) và lỗi làm tròn của bộ cộng cuối cùng ($\varepsilon_{ADD}$). Vì vậy, lỗi tổng cộng sẽ là:

$$\varepsilon_{total} = \varepsilon_{ROM} + \varepsilon_{MUl} + \varepsilon_{ADD} + \varepsilon_{apx2} \tag{6}$$

***Bảng 6.*** *Lỗi của các modul phần cứng.*

| Modul | ROM | Bộ nhân rút gọn | Bộ cộng |
|---|---|---|---|
| Độ rộng bit | $B_F$ | $B_M$ | $B_A$ |
| $\varepsilon_{Modul}$ | $2^{-B_F-1}$ | $2^{-B_M+1}$ | $2^{-B_A-1}$ |
| $\varepsilon_{ROM}$ | $2^{-B_F-1}$ | | |
| $\varepsilon_{MUL}$ | $2^{-B_F}+2^{-B_M+1}$ | | |
| $\varepsilon_{ADD}$ | | | $2^{-B_A-1}$ |

Ký hiệu $B_F$, $B_M$ và $B_A$ là độ rộng bit tương ứng của các bộ nhớ ROM, bộ nhân rút gọn và bộ cộng, chúng ta có thể biểu diễn lỗi do các thành phần phần cứng này gây ra như trên bảng 6.

Lỗi xấp xỉ ở mức 2, $\varepsilon_{apx2}$, được xác định là lỗi tuyệt đối lớn nhất do xấp xỉ ở mức 2 của tất cả các hàm $e_i(x')$. Trong thực thi phần cứng chúng tôi thực hiện xấp xỉ mức 1 với 8 phân đoạn ($n_1 = 3$) tương ứng tạo ra 8 hàm lỗi $e_i(x')$ và mỗi hàm lỗi này được xấp xỉ bởi phân đoạn tuyến tính của hai cặp đoạn đối xứng. Độ rộng bit của ROM và bộ cộng là 23 bit ($B_F = B_A = 23$) và độ rộng bit của bộ nhân rút gọn là 20 bit ($B_M = 20$). Khi đó, các thành phần lỗi và lỗi tổng cộng tương ứng với hàm khác nhau được biểu diễn như trên bảng 7.

**Bảng 7.** *Các lỗi thành phần và lỗi tổng cộng của thực thi các hàm.*

| Hàm | $\log_2(x)$ | $\sin(x)$ | $1/x$ | $\sqrt{x}$ | $1/\sqrt{x}$ | $2^x$ |
|---|---|---|---|---|---|---|
| $\varepsilon_{ROM}$ | $2^{-24}$ | $2^{-24}$ | $2^{-24}$ | $2^{-24}$ | $2^{-24}$ | $2^{-24}$ |
| $\varepsilon_{MUL}$ | $2^{-23}+2^{-19}$ | $2^{-23}+2^{-19}$ | $2^{-23}+2^{-19}$ | $2^{-23}+2^{-19}$ | $2^{-23}+2^{-19}$ | $2^{-23}+2^{-19}$ |
| $\varepsilon_{ADD}$ | $2^{-24}$ | $2^{-24}$ | $2^{-24}$ | $2^{-24}$ | $2^{-24}$ | $2^{-24}$ |
| $\varepsilon_{apx2}$ | $9.85\times10^{-5}$ | $7.95\times10^{-5}$ | $7.93\times10^{-5}$ | $2.04\times10^{-5}$ | $6.17\times10^{-5}$ | $8.79\times10^{-4}$ |
| $\varepsilon_{total}$ | $1.01\times10^{-4}$ | $8.16\times10^{-5}$ | $8.14\times10^{-5}$ | $2.25\times10^{-5}$ | $6.38\times10^{-5}$ | $9.01\times10^{-5}$ |

## V. KẾT LUẬN

Bài báo này đã trình bày một phương pháp tính toán các hàm toán học được ứng dụng phổ biến trong xử lý tín hiệu số dựa trên xấp xỉ hai mức, mức xấp xỉ thứ nhất dựa trên phương pháp xấp xỉ phân đoạn tuyến tính đều và mức xấp xỉ thứ hai là bước xấp xỉ hàm lỗi gây ra bởi mức đầu tiên theo phương pháp xấp xỉ phân đoạn tuyến tính có đối xứng. Các thực thi một số hàm toán học điển hình theo phương pháp đề xuất đã cho thấy hiệu quả về tốc độ xử lý. Vì vậy, kiến trúc đề xuất là phù hợp với các ứng dụng đòi hỏi tốc độ xử lý cao.

## TÀI LIỆU THAM KHẢO


[1]. J. E. Volder, "*The CORDIC Trigonometric Computing Technique,*" IRE Transactions on Electronic Computers, vol. EC-8, pp. 330-334, 1959.

[2]. A. S. Noetzel, "*An interpolating memory unit for function evaluation: analysis and design*," IEEE Transactions on Computers, vol. 38, pp. 377-384, 1989.

[3]. G. H. Garcia and W. J. Kubitz, "*Minimum Mean Running Time Function Generation Using Read-Only Memory,*" IEEE Transactions on Computers, vol. C-32, pp. 147-156, 1983.

[4]. P. T. P. Tang, "*Table-lookup algorithms for elementary functions and their error analysis,*" in Proceedings 10th IEEE Symposium on Computer Arithmetic, 1991, pp. 232-236.

[5]. I. Koren and O. Zinaty, "*Evaluating elementary functions in a numerical coprocessor based on rational approximations,*" IEEE Transactions on Computers, vol. 39, pp. 1030-1037, 1990.

[6]. J. A. Pineiro, S. F. Oberman, J. M. Muller, and J. D. Bruguera, "*High-speed function approximation using a minimax quadratic interpolator,*" IEEE Transactions on Computers, vol. 54, pp. 304-318, 2005.
[7]. M. J. Schulte and E. E. Swartzlander, "*Hardware designs for exactly rounded elementary functions,*" IEEE Transactions on Computers, vol. 43, pp. 964-973, 1994.
[8]. E. G. Walters and M. J. Schulte, "*Efficient function approximation using truncated multipliers and squarers,*" in 17th IEEE Symposium on Computer Arithmetic (ARITH'05), 2005, pp. 232-239.
[9]. S. F. Hsiao, H. J. Ko, and C. S. Wen, "*Two-Level Hardware Function Evaluation Based on Correction of Normalized Piecewise Difference Functions,*" IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 59, pp. 292-296, 2012.

**ABSTRACT**

A HARDWARE DESIGN FOR FUNCTION EVALUATION
BASED ON TWO LINEAR APPROXIMATION LEVELS

*In this paper, a common method for computing math functions based on two approximation levels is presented. At the first approximation level, these math functions are approximated by the equal linear segment approximation approach. After that, at the second approximation level, the error functions in the first step will be approximated by the linear segment method, having symmetric interpolation, to reduce hardware complexity. Based on the proposed method, the hardware structure to compute typical math functions are designed and implemented. The performance results show that the proposed hardware achieves speed efficiency.*

**Key words:** Computer arithmetic; Function evaluation; Piecewise polynomial approximation.

*Nhận bài ngày 09 tháng 03 năm 2017*
*Hoàn thiện ngày 22 tháng 5 năm 2017*
*Chấp nhận đăng ngày 20 tháng 6 năm 2017*

*Địa chỉ:* Học viện Kỹ thuật quân sự.
*Email: saivanthuan@gmail.com