Mở đầu

1. Bối cảnh và động lực nghiên cứu

Hiện nay các hệ thống xử lý tín hiệu số (DSP: Digital Signal Processing) có mặt trong hầu hết các hệ thống điện tử và truyền thông. Cùng với các nhu cầu ngày càng tăng của các ứng dụng DSP, thị trường các thiết bị DSP đã có những tăng trưởng mạnh mẽ. Bên cạnh đó, sự phát triển nhanh chóng của các thiết bị và hạ tầng cho truyền thông không dây là một lý do quan trọng cho sự tăng trưởng của thị trường DSP toàn cầu.

Các hàm toán học cơ sở, là rất phổ biến trong nhiều ứng dụng xử lý tín hiệu số và giữ vai trò quan trọng trong quyết định tốc độ xử lý và tài nguyên tiêu tốn của các bộ xử lý. Các bộ xử lý hiện đại với yêu cầu ngày càng cao về tốc độ, chiếm ít tài nguyên và công suất thấp đặt ra yêu cầu phải có các giải pháp thiết kế các đơn vị phần cứng tính toán các hàm toán học trên một cách hiệu quả.

Hơn nữa, cùng với xu hướng phát triển mạnh mẽ của thị trường DSP các hệ thống và thiết bị DSP sẽ được sử dụng ngày càng nhiều trong các ứng dụng. Cùng với đó là càng nhiều yêu cầu đặt ra đối với các lõi phần cứng chuyên dụng cho tính toán các hàm toán học cả về hiệu năng và ứng dụng, nhất là trong các hệ thống xử lý ảnh 3-D và hệ thống đa phương tiện tốc độ cao.

Vì vậy, cần có các giải pháp khác nhau để đạt được kiến trúc tính toán với độ phức tạp thấp trong khi vẫn đạt được hiệu năng tính toán cần thiết. Do đó, nhiều nghiên cứu đã tập trung tìm kiếm các giải pháp khác nhau để đạt được các kiến trúc tính toán hiệu quả.

Thiết kế phần cứng các lõi tính toán dựa trên cơ sở bảng tra LUT kết hợp với một số mạch logic đơn giản là một giải pháp hứa hẹn đạt được hiệu quả cao. Trong đó, phương pháp kết hợp bảng LUT với một vài mạch logic đơn giản dựa trên các thuật toán tối ưu là một giải pháp hứa hẹn đạt được các kiến trúc phần cứng đơn giản và hiệu quả về tốc độ, công suất thấp, tài nguyên tiêu tốn ít. Vì vậy, luận án sẽ tập trung nghiên cứu thực thi một vài hàm toán học ứng dụng trong DSP dựa trên phương pháp nêu trên.

Ngoài ra, trong một vài năm gần đây kỹ thuật tính toán ngẫu nhiên (SC: Stochastic computing) thu hút được sự quan tâm trong thiết kế phần cứng các mạch số học có độ phức tạp rất thấp. Đó cũng là cơ sở để tạo động lực cho nghiên cứu trong luận án này nhằm đạt được các kiến trúc tính toán hiệu quả dựa trên SC.

2. Đối tượng và phạm vi nghiên cứu

Luận án này tập trung nghiên cứu tìm phương pháp để đạt được kiến trúc phần cứng hiệu quả cho thực hiện các hàm toán học phổ biến ứng dụng trong xử lý tín hiệu số. Những đặc điểm của các ứng dụng DSP có đặc điểm riêng, nó khác với các ứng dụng tính toán nói chung nên phương pháp thực thi và kiến trúc cũng phải phù hợp. Một số đặc điểm quan trọng của các ứng dụng DSP được xem xét như cho phép lỗi, đòi hỏi độ phức tạp thấp, công suất thấp…Vì vậy đối tượng và phạm vi nghiên cứu của luận án là tìm các phương pháp hiệu quả phù hợp với các ứng dụng xử lý tín hiệu số theo các đặc điểm như trên.

3. Các đóng góp chính của luận án

Đề xuất phương pháp xấp xỉ hàm logarithm nhị phân và hàm sin dựa trên xấp xỉ phân đoạn tuyến tính đều kết hợp với LUT bù sai số xấp xỉ. Các kiến trúc phần cứng dựa trên phương pháp đề xuất đạt được độ phức tạp thấp. Đóng góp của đề xuất này được trình bày trong các công trình số 1, số 2 và số 3.

Đề xuất một phương pháp xấp xỉ và các hàm toán học phổ biến trong DSP dựa trên xấp xỉ tuyến tính hai mức. Phương pháp đề xuất đã được áp dụng cho thực thi phần cứng 6 hàm toán học điển hình. Các kết quả thực thi cho thấy phương pháp đề xuất đạt được hiệu quả về tốc độ. Đóng góp của đề xuất này được trình bày trong công trình số 4.

Đề xuất phương pháp thực thi phần cứng tính toán các hàm toán học dựa trên tính toán ngẫu nhiên kết hợp với xấp xỉ các hàm toán học phức tạp bằng các hàm tuyến tính phân đoạn đều. Các kết quả thực thi theo phương pháp đề xuất đã cải thiện được hiệu năng của các lõi tính toán. Đóng góp của đề xuất này được trình bày trong công trình số 5 và số 6.

4. Cấu trúc của luận án

Cấu trúc luận án, ngoài phần mở đầu, kết luận gồm có 5 chương. Chương 1 trình bày cơ sở và phương pháp nghiên cứu, Chương 2 thiết kế phần cứng tính toán hàm logarithm nhị phân, Chương 3 thiết kế phần cứng tính toán hàm sin, Chương 4 thiết kế phần cứng tính toán các hàm toán học dựa trên xấp xỉ tuyến tính hai mức và Chương 5 thiết kế phần cứng tính toán các hàm toán học sử dụng logic ngẫu nhiên và xấp xỉ phân đoạn tuyến tính đều.

CHƯƠNG 1
CƠ SỞ VÀ PHƯƠNG PHÁP NGHIÊN CỨU

1.1. Các hàm toán học trong xử lý tín hiệu số

Các hệ thống xử lý tín hiệu số thực hiện các chức năng khác nhau đều dựa trên một tập hợp các thao tác tính toán cơ bản được thực hiện trên phần cứng. Các thao tác tính toán cơ bản bao gồm phép cộng, phép nhân cho đến các hàm cơ sở như các hàm lượng giác, hàm mũ, khai căn, logarithm…Các hàm cơ sở đóng vai trò quan trọng trong nhiều ứng dụng bao gồm khoa học tính toán, đồ họa máy tính, các ứng dụng đồ họa 3D, xử lý tín hiệu số và thiết kế có hỗ trợ máy tính. Thiết kế phần cứng hiệu quả cho tính toán các hàm toán học này có ý nghĩa quan trọng trong nâng cao hiệu năng các hệ thống xử lý tín hiệu số.

Tính toán các hàm lượng giác là thao tác thường xuyên trong nhiều ứng dụng xử lý tín hiệu số. Ngoài ra, tính toán hàm lượng giác có thể được sử dụng trong bộ tổ hợp tần số số, bộ trộn tần, bộ điều chế và giải điều chế và nhiều ứng dụng trong các hệ thống truyền thông số.

Hàm logarithm là thao tác chính trong các ứng dụng chuyển đổi tỷ số công suất và phổ công suất sang biểu diễn dạng đề-xi-bel, trong các ứng dụng xử lý âm thanh. Tính toán hàm logarithm còn là thao tác không thể thiếu trong các hệ thống số logarithm (LNS:Logarithmic Number System) để chuyển đổi dữ liệu đầu vào sang miền logarithm.

Trong kỹ thuật đồ họa 3D, hơn 80% thời gian xử lý của bộ xử lý đồ họa là để thực hiện dựng ảnh (render), hơn nữa hơn 90% thời gian xử lý dựng ảnh là thực hiện tính toán các hàm toán học cơ sở. Vì vậy, các hàm toán học cơ sở được sử dụng thường xuyên trong ứng dụng xử lý đồ họa 3D và nó quyết định tốc độ xử lý của các đơn vị xử lý đồ họa.

Như vậy, tính toán các hàm toán học là yêu cầu đặt ra trong hầu hết các ứng dụng DSP. Với các yêu cầu ngày càng cao về tài nguyên và tốc độ của các bộ xử lý DSP, cần có các giải pháp thiết kế phần cứng hiệu quả cho tính toán các hàm toán học.

1.2. Các phương pháp thực thi phần cứng cho tính toán các hàm toán học

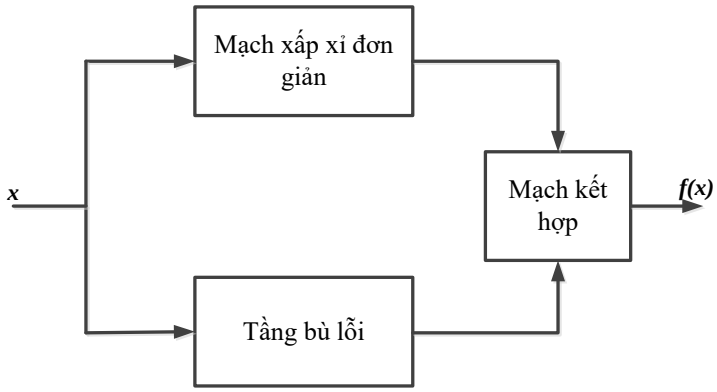
Thực thi các hàm cơ sở bằng phần cứng có ưu điểm đáng kể về mặt tốc độ cũng như khả năng tăng thông lượng bằng việc sử dụng song song nhiều mô-đun phần cứng. Các yêu cầu ngày càng lớn về tốc độ và thông lượng của nhiều ứng dụng là một động lực cho việc phát triển các phương

pháp phần cứng để tính toán các hàm cơ sở với tốc độ cao. Một số phương pháp thực thi các hàm toán học bằng phần cứng điển hình là:

- Phương pháp xấp xỉ bằng đa thức
- Phương pháp sử dụng bảng
- Phương pháp sử dụng thuật toán CORDIC
- Phương pháp xấp xỉ hữu tỷ

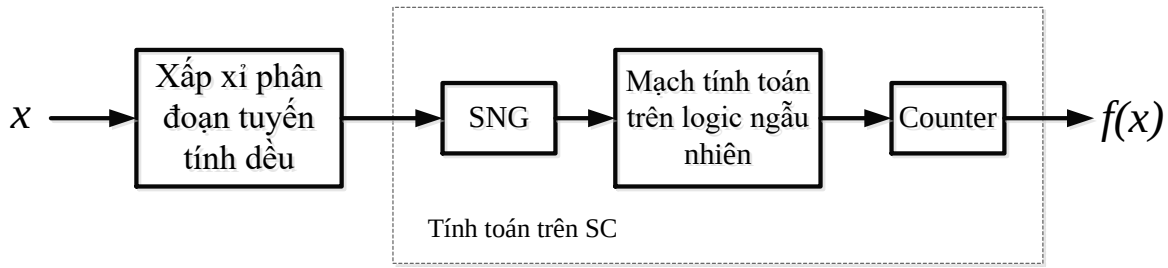
1.3. Phương pháp thực thi các hàm toán học sử dụng trong luận án

Với mục đích giảm độ phức tạp phần cứng của các lõi tính toán các hàm toán học và với mức sai số có thể chấp nhận của các ứng dụng DSP, trong luận án này sử dụng phương pháp xấp xỉ tuyến tính phân đoạn đều. Sau đó, việc bù sai số xấp xỉ có thể thực hiện bằng cách sử dụng một LUT có kích thước nhỏ hoặc tiếp tục sử dụng xấp xỉ tuyến tính các hàm sai số của xấp xỉ ban đầu có lợi dụng các đặc trưng của hàm sai số để giảm chi phí phần cứng và tăng tốc độ. Hình 1.1 mô tả phương pháp này.



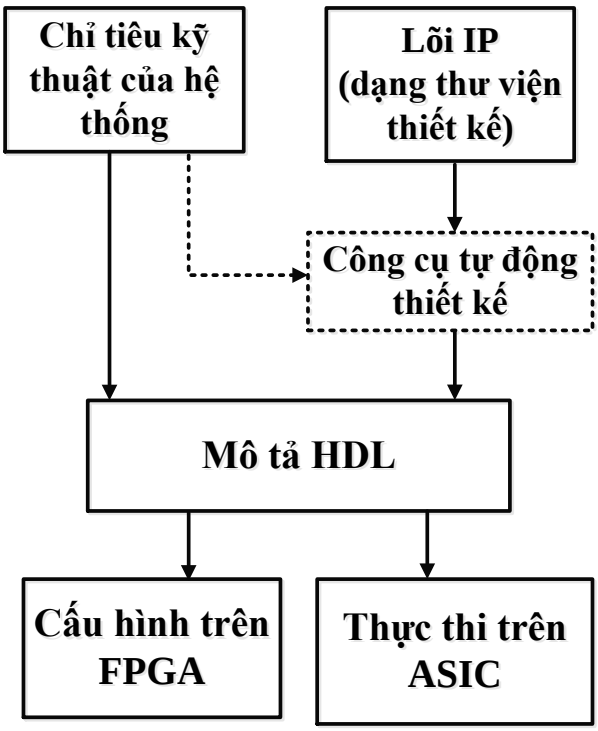
Hình 1.1: *Phương pháp độ chênh lệch.*

Ngoài ra, lý thuyết về tính toán ngẫu nhiên (SC: Stochastic Computing) cho phép ứng dụng để thực thi các đơn vị số học với chi phí rất thấp và lỗi chấp nhận được. Trong luận án này, còn sử dụng phương pháp thực thi các hàm toán học dựa trên xấp xỉ hàm phức tạp bằng các hàm tuyến tính phân đoạn đều. Sau đó, các hàm xấp xỉ sẽ được thực thi dựa trên logic ngẫu nhiên. Hình 1.2 mô tả phương pháp thực thi các hàm dựa trên kết hợp xấp xỉ phân đoạn tuyến tính đều và sử dụng SC.



Hình 1.2: *Phương pháp tính toán sử dụng SC.*

Hình 1.3 mô tả qui trình thiết kế tổng quát thực hiện xấp xỉ hóa các hàm phức tạp trong DSP sử dụng thư viện lõi IP cứng tính toán các hàm phức tạp cho các hệ thống DSP theo phương pháp và thuật toán tối ưu tham số được đề xuất.



Hình 1.3: *Qui trình tạo lõi IP tính toán.*

1.4. Kết luận chương 1

Chương 1 trình bày khái quát các vấn đề cơ bản liên quan đến các giải pháp đề xuất trong luận án bao gồm việc sử dụng các hàm toán học trong các ứng dụng xử lý tín hiệu số, các phương pháp điển hình trong thiết kế phần cứng tính toán các hàm toán học. Đồng thời cũng trình bày phương pháp chung sẽ được sử dụng trong các đề xuất để thiết kế phần cứng tính toán các hàm toán học trong luận án.

CHƯƠNG 2
THIẾT KẾ PHẦN CỨNG TÍNH TOÁN HÀM LOGARITHM NHỊ PHÂN

2.1. Đặt vấn đề

Nhiều ứng dụng của xử lý tín hiệu số hiện đại đòi hỏi nhiều phép tính phức tạp như phép nhân, chia, khai căn, lũy thừa …Bằng việc sử dụng tính toán trên miền logarithm các phép toán trên có thể được đơn giản đi một cách đáng kể, ví dụ như các phép nhân có thể được thực hiện bằng các phép cộng, trong khi các phép tính căn bậc hai có thể được thay thế bằng các phép dịch trên miền logarithm.

Hơn nữa, trên thực tế, tính toán các hàm logarithm là một yêu cầu đặt ra trong nhiều ứng dụng như: xử lý ảnh, trong các hệ thống truyền thông, hệ thống y sinh…Trong nhiều trường hợp, việc tính toán các hàm logarithm bằng phần mềm là không đủ nhanh và do đó sử dụng các phần cứng chuyên dụng để thực hiện là cần thiết.

2.2. Đề xuất phương pháp xấp xỉ hàm logarihtm

Về mặt toán học một số nhị phân không dấu N

$$N = z_k ... z_2 z_3 z_0 . z_{-1} z_{-2} z_{-3} ... z_j = \sum_{i=j}^{k} 2^i z_i, \;\; z_i = \{0,1\}. \tag{2.1}$$

$$N = 2^k + \sum_{i=j}^{k-1} 2^i z_i = 2^k (1 + \sum_{i=j}^{k-1} 2^{i-k} z_i) = 2^k (1+x), \;\; x = \sum_{i=j}^{k-1} 2^{i-k} z_i \tag{2.2}$$

Ở đây x trong khoảng [0, 1). Như vậy, logarithm cơ số 2 của N sẽ là:

$$\log_2(N) = k + \log_2(1+x) \tag{2.3}$$

Ở đây $k = \lfloor \log_2(N) \rfloor$ và $\log_2(1+x)$ tương ứng là phần nguyên và phần thập phân của $\log_2(N)$. Giá trị của k có thể nhận được bằng việc sử dụng một bộ phát hiện bit ‘1’ đầu tiên (LOD: Leading-One Detector) trong dữ liệu đầu vào, khi đó tính toán $\log_2(N)$ sẽ tùy thuộc vào tính toán $\log_2(1+x)$.

Phương pháp đề xuất xấp xỉ hàm $\log_2(1+x)$ bằng 4 phân đoạn tuyến tính đều như sau:

$$\log_2(1+x) \approx \begin{cases} (2^0 + 2^{-2})x \; + \; 2^{-7}, & 0 \le x < 0,25 \\ (2^0 + 2^{-4})x \; + \; 2^{-4}, & 0,25 \le x < 0,5 \\ (2^0 - 2^{-3})x + \; 77 \times 2^{-9}, & 0,5 \le x < 0,75 \\ (2^{-1} + 2^{-2})x \; + \; 2^{-2}, & 0,75 \le x < 1 \end{cases} \tag{2.4}$$

2.3. Phân tích sai số và so sánh

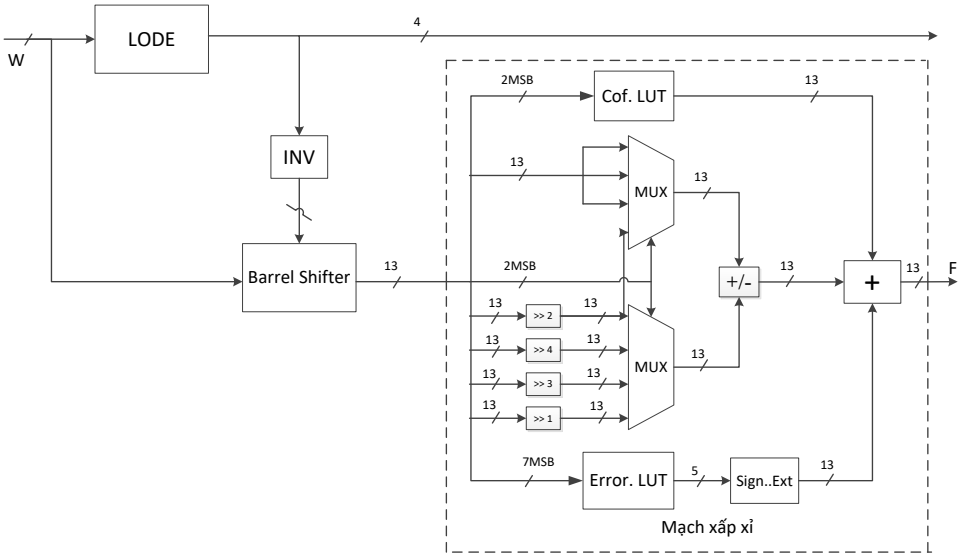
Phương pháp đề xuất được tiến hành phân tích sai số và so sánh với các kết quả trong [34], [41] và [48] (cũng xấp xỉ tuyến tính 4 đoạn). Các tham số được phân tích và so sánh lỗi gồm các tham số sai số (MAE, MPE, MNE), trung bình sai số, các tham số phần trăm sai số (MPEP, MNEP) và trung bình phần trăm sai số.

Bảng 2.1: *Phân tích sai số và so sánh của phương pháp đề xuất.*

Phương pháp	E.L.Hall [34]	De caro[41]	H.Phuc [48]	Đề xuất
MPE	$8{,}1\times10^{-3}$	$9{,}8\times10^{-3}$	$10{,}1\times10^{-3}$	$\mathbf{6{,}3\times10^{-3}}$
MNE	$-2{,}3\times10^{-3}$	$-6{,}8\times10^{-3}$	$-10{,}1\times10^{-3}$	$\mathbf{-8{,}8\times10^{-3}}$
MAE	8,1×10-3	9,8×10-3	10,1×10-3	**8,8×10-3**
Trung bình lỗi	$1{,}2\times10^{-3}$	$2{,}1\times10^{-3}$	$2{,}1\times10^{-3}$	$\mathbf{1{,}76\times10^{-4}}$
MEP	0,78%	0,43%	0,4%	**0,78%**
MPEP	0,13%	0,18%	0,31%	**0,18%**
MNEP	-0,78%	-0,43%	-0,4%	**-0,78%**
Trung bình phần trăm lỗi	$5{,}81\times10^{-5}$	$2{,}03\times10^{-4}$	$2{,}48\times10^{-4}$	$\mathbf{4{,}98\times10^{-5}}$

2.4. Kiến trúc phần cứng và kết quả thực thi

Dựa trên phương pháp đề xuất, một kiến trúc phần cứng của bộ chuyển đổi logarithm cho số nguyên N có 16 bit đầu vào tương ứng với 4 bit phần nguyên và 13 bit phần thập phân đầu ra được thực hiện. Kiến trúc phần cứng đề xuất cho tính toán hàm logarithm cơ số hai của một số N thể hiện như Hình 2.1.



Hình 2.1: *Kiến trúc đề xuất cho bộ tính toán logarithm 16 bit.*

Kiến trúc đề xuất đã được mô hình hóa trên ngôn ngữ VHDL và thực thi trên thiết bị Xilinx FPGA Spatan 3E. Tài nguyên tiêu tốn trong thực thi

trên FPGA được tính thông qua số FPGA LUT được sử dụng. Bảng 2.2 thể hiện các kết quả thực thi trên FPGA của phương pháp đề xuất và so sánh với các kết quả trong [34], [40] và [41]. Ngoài ra, kết quả thực thi cho bộ chuyển đổi logarithm theo phương pháp đề xuất trên công nghệ ASIC SOTB CMOS 65nm thể hiện trên Bảng 2.3.

Bảng 2.2: *Kết quả thực thi và so sánh trên FPGA*

Phương pháp	Slices	FPGA LUTs	Delay (ns)	ADP ($\times 10^3$)
Hall [34]	98	188	28,574	5,371
Guitierz [40]	86	163	24,479	3,989
De Caro [41]	86	162	24,479	3,965
Đề xuất	**81**	**149**	**23,066**	**3,436**

Bảng 2.3: *Kết quả thực thi và so sánh trên ASIC*

Phương pháp	Gutierez [40]	Hall [34]	De Caro [41]	Đề xuất
Diện tích ($\times 10^3 \mu m^2$)	42,9	54,1	40,3	27,2
Độ giữ chậm (ns)	12,2	13,2	12,0	11,5
ADP($\times 10^3$)	523,4	714,1	483,6	312,8
Công suất (µW)	17,70	23,09	14,98	9,40

2.5. Kết luận chương 2

Chương 2 đề xuất một phương pháp xấp xỉ hàm logarithm sử dụng phương pháp xấp xỉ phân đoạn tuyến tính đều, với các hệ số góc của đoạn là có dạng là tổng của các số lũy thừa của hai. Để tăng độ chính xác xấp xỉ, một bảng LUT 5×128 bits được sử dụng. Trên cơ sở đó một kiến trúc thực hiện cho bộ chuyển đổi một số nhị phân 16 bit nguyên cũng được đề xuất và thực thi trên FPGA và ASIC. Các kết quả thực thi và kiểm chứng cho thấy bộ chuyển đổi logarithm đề xuất cải thiện về tài nguyên phần cứng.

CHƯƠNG 3
THIẾT KẾ PHẦN CỨNG TÍNH TOÁN HÀM SIN

3.1. Đặt vấn đề

Tính toán hàm sin là một yêu cầu đặt ra trong nhiều ứng dụng DSP chẳng hạn trong các bộ tổ hợp tần số trực tiếp (DDFS: Direct Digital Frequency Synthesizer), trong các mạch trộn tần, các mạch điều chế số. Tính toán hàm sin có thể thực hiện bằng phần mềm, tuy nhiên tốc độ tính toán là chậm và không phù hợp với các ứng dụng đòi hỏi tốc độ xử lý cao. Vì vậy, với các ứng dụng yêu cầu về tốc độ cao và các DSP thời gian thực, thực thi bằng phần cứng là cần thiết. Một vài phương pháp thực thi hàm sin đã được nghiên cứu, chẳng hạn như các kiến trúc tính toán hàm sin dựa trên thuật toán CORDIC Tuy nhiên, các kiến trúc này dựa trên giải pháp lặp do vậy không phù hợp với các ứng dụng yêu cầu tốc độ tính toán cao và các ứng dụng DSP thời gian thực. Ngoài ra, một số phương pháp xấp xỉ hàm sin như xấp xỉ bằng hàm bậc cao, xấp xỉ sử dụng chuỗi Taylor thường có độ phức tạp phần cứng cao. Vì vậy, với các ứng dụng yêu cầu về cao về tốc độ tính toán và đơn giản về phần cứng thực thi đòi hỏi phải có các giải pháp hiệu quả cho thực thi tính toán hàm sin.

Chương 3 của luận án nghiên cứu thiết kế phần cứng tính toán hàm sin ứng dụng cho DDFS, trong đó tập trung vào cải thiện tỷ lệ nén sin-LUT của bộ chuyển đổi pha thành biên độ (PAC: Phase to Amplitude Converter) trong DDFS.

3.2. Đề xuất phương pháp cải tiến nén biên độ sin ứng dụng trong DDFS

3.2.1. Cơ sở và ý tưởng đề xuất

Trên cơ sở phân tích các công trình nghiên cứu trước đó (công trình [68] và [69]) thấy rằng vẫn có những vấn đề có thể cải thiện để đạt được kiến trúc hiệu quả hơn. Ý tưởng đề xuất là tiếp tục lợi dụng ưu điểm tính đối xứng ¼ của hàm sin và kết hợp với kỹ thuật phân chia góc. Đồng thời đề xuất một phương pháp nén biên độ sin mới để khắc phục những hạn chế trong [68] và [69]. Cụ thể, để đạt được tỷ số nén sin-LUT cao hơn và đồng thời kiến trúc phần cứng đơn giản hơn, Chương 3 của luận án sẽ đề xuất một phương pháp xấp xỉ dựa trên xấp xỉ phân đoạn tuyến tính đều với số đoạn là một số lũy thừa của 2 và các hệ số góc là các số có dạng là tổng của các số lũy thừa của hai.

3.2.2. Đề xuất phương pháp nén biên độ sin

Để thuận tiện cho phân tích về mặt toán học, khoảng giá trị pha của tín hiệu đầu vào được qui ước là trong khoảng [0, 1] thay vì là $[0,\pi/2]$ cho góc phần tư đầu tiên. Giả thiết là chỉ các giá trị pha của góc phần tư đầu tiên được lưu trữ trong sin-LUT. Công thức biểu diễn xấp xỉ như sau:

$$\sin(\frac{\pi}{2}x) \approx a(x) + \varepsilon(x) \tag{3.1}$$

Trong đó $a(x)$ là hàm xấp xỉ đề xuất và $\varepsilon(x)$ biểu diễn sai số xấp xỉ và nó được chứa trong một sin-LUT có kích thước nhỏ. Trong phương pháp đề xuất hàm $a(x)$ là hàm phân đoạn tuyến tính có dạng như sau:

$$a(x) = \begin{cases} a_0 x + b_0 & 0 \le x < \dfrac{1}{s} \\ a_0 x + b_0 & \dfrac{1}{s} \le x < \dfrac{2}{s} \\ \quad\vdots & \\ a_{s-1} x + b_{s-1} & \dfrac{s-1}{s} \le x < 1 \end{cases} \tag{3.2}$$

Để đơn giản hơn nữa cấu trúc phần cứng thực thi, a_i có dạng là tổng của các số lũy thừa của 2 được biểu diễn như công thức (3.3), khi đó tránh việc sử dụng các mạch nhân trong cấu trúc phần cứng.

$$a_i = \sum_{k=0}^{M} p_k . 2^{-k}, \quad p_k = \{0,1\} \tag{3.3}$$

Các phân tích về số phân đoạn xấp xỉ và sai số tương ứng cho thấy lựa chọn xấp xỉ 8 phân đoạn là phù hợp. Các hệ số tối ưu cho xấp xỉ 8 phân đoạn thể hiện trên Bảng 3.1.

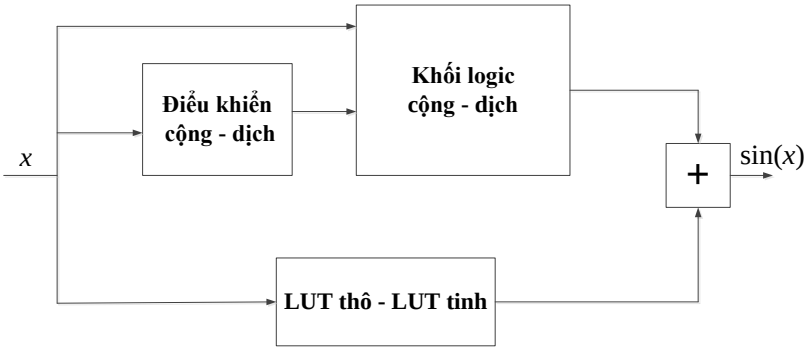
Bảng 3.1: *Giá trị các hệ số tối ưu của hàm xấp xỉ tuyến tính 8 phân đoạn*

Đoạn	**Giá trị pha đầu vào chuẩn hóa**	a_i	b_i	$\varepsilon_{i\max}$
1	$0 < x \le 0{,}125$	$2^0 + 2^{-1}$	0	0,0076
2	$0{,}125 < x \le 0{,}25$	$2^0 + 2^{-1}$	0,007	0,0021
3	$0{,}25 < x \le 0{,}375$	$2^0 + 2^{-2} + 2^{-3}$	0,038	0,0038
4	$0{,}375 < x \le 0,5$	$2^0 + 2^{-2}$	0,082	0,0059
5	$0,5 < x \le 0,625$	$2^0 + 2^{-5}$	0,186	0,0073
6	$0,625 < x \le 0{,}75$	$2^{-1} + 2^{-2}$	0,360	0,0063
7	$0,75 < x \le 0{,}875$	$2^{-2} + 2^{-3} + 2^{-4}$	0,595	0,0065
8	$0,875 < x \le 1$	$2^{-3} + 2^{-4}$	0,812	0,0076

3.3. Kết quả tổng hợp và thực thi

Dựa trên phương pháp đề xuất, kiến trúc bộ chuyển đổi pha – biên độ cho tính toán hàm sin thể hiện trên Hình 3.1.

Bộ DDFS theo phương pháp đề xuất được thiết kế với bộ tích lũy pha có các tham số tương tự như công trình [68] và [69]. Bảng 3.2 thể hiện sự so sánh kết quả thực thi của bộ DDFS đề xuất với các nghiên cứu trong [68] và [69].



Hình 3.1: *Kiến trúc bộ chuyển đổi pha-biên độ của phương pháp đề xuất.*

Bảng 3.2: *So sánh phương pháp đề xuất với các phương pháp trước đó.*

Phương pháp	Sunderland	Độ chênh lệch sin-pha	Độ chênh lệch truyền thống	Trong [68]	Trong [69]	Đề xuất
Kích thước LUT (bits)	4096	3854	2688	2560	1536	**1408**
Tỷ số nén Sin-LUT	44:1	50:1	67:1	70:1	117,3:1	**128:1**
Số Slices	144	146	133	176	136	**85**

3.4. Kết luận chương 3

Chương này đã trình bày một phương pháp tính toán hàm sin ứng dụng cho DDFS dựa trên phương thức độ chênh lệch tuyến tính với hàm tuyến tính phân đoạn đều có các hệ số tối ưu nhằm đạt được tỷ số nén LUT cao và cấu trúc phần cứng đơn giản. Một kiến trúc DDFS dựa trên phương pháp đề xuất đã được tổng hợp và thực thi. Kiến trúc đề xuất đã đạt được tỷ số nén LUT là 128:1 với độ phức tạp phần cứng nhỏ. Vì vậy, kiến trúc đề xuất là phù hợp với các ứng dụng đòi hỏi độ phức tạp phần cứng thấp.

CHƯƠNG 4.
THIẾT KẾ PHẦN CỨNG TÍNH TOÁN CÁC HÀM TOÁN HỌC DỰA TRÊN XẤP XỈ TUYẾN TÍNH HAI MỨC

4.1. Đặt vấn đề

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, xấp xỉ đa thức, xấp xỉ hữu tỷ, và các phương pháp dựa trên bảng. 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.

4.2. Đề xuất phương pháp xấp xỉ

Trong phương pháp đề xuất hàm được xấp xỉ bởi hai mức. Trong mức đầu tiên sử dụng xấp xỉ phân đoạn tuyến tính đều và mức thứ hai thực hiện xấp xỉ hàm sai số do xấp xỉ của mức đầu tiên sử dụng phương pháp xấp xỉ phân đoạn tuyến tính đối xứng.

4.2.1. Xấp xỉ mức 1

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_2. Trong bước xấp xỉ đầu tiên khoảng giá trị của x được chia thành 2^{n_1} đoạn dựa trên x_1. Với đoạn thứ i, $x_i \in [x_i, x_{i+1})$ hai giá trị $f(x_i)$ và $f(x_{i+1})$ được sử dụng trong xấp xỉ ở mức thứ nhất để xấp xỉ hàm bằng một hàm $f_i(x')$, ở đây x' nhận giá trị trong khoảng $[0, 2^{-n_1})$.

$$f_i(x') = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i} \cdot x' + f(x_i) \quad (4.1)$$

Độ chênh lệch giữa hàm xấp xỉ mức 1 và hàm thực là các hàm $e_i(x')$ có biểu diễn như công thức (4.2)

$$e_i(x') = f(x_i + x') - f_i(x') \quad (4.2)$$

4.2.2. Xấp xỉ mức 2

Phân tích về mặt toán học cho thấy các hàm $e_i(x')$ là các hàm có dạng parabol và có dạng đồ thị đối xứng qua trục $x' = 2^{-n_1-1}$. Do vậy, ở mức xấp xỉ thứ hai, các hàm lỗi này sẽ được xấp xỉ bằng các hàm phân đoạn tuyến tính có sử dụng nội suy đối xứng sẽ giảm chi phí về phần cứng.

Trong phương pháp đề xuất, ở mức xấp xỉ thứ 2 nửa khoảng giá trị của $x', x' \in [0, 2^{-n_1-1})$, được chia thành s đoạn con, trong mỗi đoạn con thứ $j, j = 0,1...s-1$ hàm lỗi $e_i(x')$ được xấp xỉ bằng một đoạn tuyến tính có các hệ số là a_{ij} và b_{ij}. Trong đó, để đơn giản hơn nữa về phần cứng các hệ số a_{ij} và b_{ij} được gán có dạng như công thức (4.3)

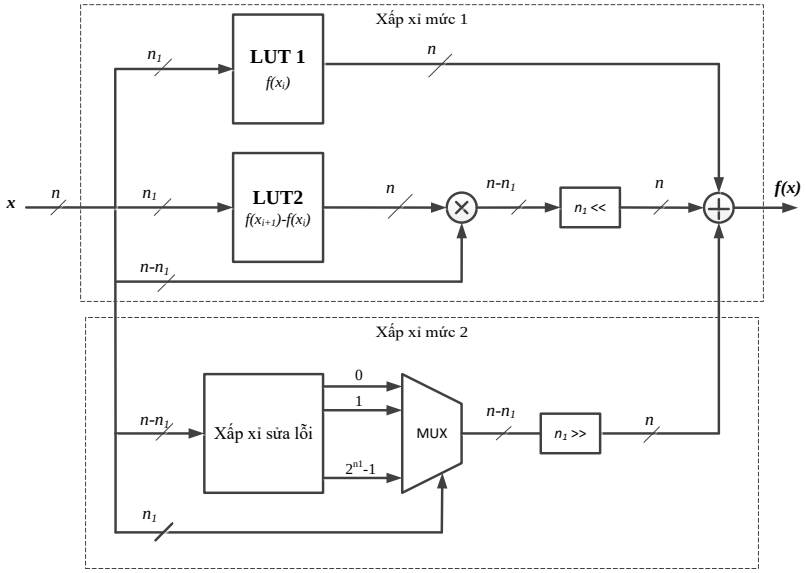
$$\begin{aligned} a_{ij} &= \sum 2^{-N} \quad ; \; N \in \mathbb{N} \\ b_{ij} &= B \times 2^{-(n-n_1)} \quad ; B \in \mathbb{N} \end{aligned} \quad (4.3)$$

Thuật toán đề xuất tìm các giá trị hệ số a_{ij} và b_{ij} sao cho sai số xấp xỉ trong từng phân đoạn là cực. Trong phương pháp đề xuất sử dụng một chương trình Matlab 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 cho trường hợp 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.

4.3. 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 4.1. Trong mức xấp xỉ thứ nhất thực hiện tính toán hàm $f_i(x')$ như công thức (4.1). Trong đó, các giá trị $f(x_i)$ và $f(x_{i+1}) - f(x_i)$ được lưu vào trong các bảng LUT, 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 4.1: *Kiến trúc phần cứng tổng quát.*

Dựa trên kiến trúc tổng quát được đề xuất, các thực thi phần cứng để tính toán các hàm 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. Các kiến trúc đề xuất đã được mô hình trên ngôn ngữ VHDL và thực thi trên thiết bị FPGA Xilinx Virtex 6. Kết quả thực thi trên FPGA của các kiến trúc đề xuất thể hiện trên Bảng 4.1. Bảng 4.2 thể hiện kết quả tổng hợp bằng Synopsys Design Compiler cho các hàm khác nhau của phương pháp đề xuất và so sánh với công trình [75] trên thư viện công nghệ CMOS 90 nm.

Bảng 4.1: *Kết quả thực thi của các kiến trúc đề xuất trên FPGA.*

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

4.4. 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 ε_{ROM}), lỗi gây ra bởi bộ nhân rút gọn (ε_{MUL}), lỗi xấp xỉ do xấp xỉ mức hai (ε_{apx2}) và lỗi làm tròn của bộ cộng cuối cùng (ε_{ADD}). Vì vậy, lỗi tổng cộng sẽ là:

$$\varepsilon_{total} = \varepsilon_{ROM} + \varepsilon_{MUl} + \varepsilon_{ADD} + \varepsilon_{apx2} \quad (4.4)$$

Lỗi xấp xỉ ở mức 2, ε_{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 đề xuất thực hiện xấp xỉ mức 1 với 8 phân đoạn ($n_1 = 3$). Độ rộng bit của ROM và bộ cộng là 23 bit và độ rộng bit của bộ nhân rút gọn là 20 bit. Khi đó, các thành phần lỗi và lỗi tổng cộng tương ứng với hàm khác nhau được biểu diễn như trên Bảng 4.3.

Bảng 4.2: *Kết quả tổng hợp và so sánh trên ASIC.*

Hàm	Đề xuất		Trong [75]	
	Diện tích (µm^2)	Độ trễ (ns)	Diện tích (µm^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

Bảng 4.3: *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
ε_{ROM}	2^{-24}	2^{-24}	2^{-24}	2^{-24}	2^{-24}	2^{-24}
ε_{MUL}	$2^{-23}+2^{-19}$	$2^{-23}+2^{-19}$	$2^{-23}+2^{-19}$	$2^{-23}+2^{-19}$	$2^{-23}+2^{-19}$	$2^{-23}+2^{-19}$
ε_{ADD}	2^{-24}	2^{-24}	2^{-24}	2^{-24}	2^{-24}	2^{-24}
ε_{apx2}	$9{,}85\times10^{-5}$	$7{,}95\times10^{-5}$	$7{,}93\times10^{-5}$	$2{,}04\times10^{-5}$	$6{,}17\times10^{-5}$	$8{,}79\times10^{-4}$
ε_{total}	$1{,}01\times10^{-4}$	$8{,}16\times10^{-5}$	$8{,}14\times10^{-5}$	$2{,}25\times10^{-5}$	$6{,}38\times10^{-5}$	$9{,}01\times10^{-5}$

4.4. Kết luận chương 4

Chương 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 sai số 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 thực thi. Vì vậy, kiến trúc đề xuất là phù hợp với các ứng dụng đòi hỏi tốc độ tính toán cao.

CHƯƠNG 5
THIẾT KẾ PHẦN CỨNG TÍNH TOÁN CÁC HÀM TOÁN HỌC SỬ DỤNG LOGIC NGẪU NHIÊN VÀ XẤP XỈ PHÂN ĐOẠN TUYẾN TÍNH ĐỀU

5.1. Đặt vấn đề

Tính toán ngẫu nhiên (SC: Stochastic computing), được đề xuất đầu tiên từ năm 1967 và gần đây được quan tâm trở lại do việc ứng dụng vào thực thi các đơn vị số học với chi phí rất thấp và lỗi chấp nhận được. Đặc tính cơ bản của SC là một số được biểu diễn bởi một chuỗi bit và được xử lý bởi các mạch rất đơn giản. Một số thực x được biểu diễn bởi một chuỗi ngẫu nhiên X. Chuỗi ngẫu nhiên được biểu diễn dưới hai định dạng, đơn cực và lưỡng cực. Trong định dạng đơn cực, $x = p(X = 1) = p(X)$, vì x là một giá trị xác suất nên trong định dạng đơn cực cần thỏa mãn $0 \le x \le 1$. Trong định dạng lưỡng cực $x = 2p(X = 1) - 1 = 2p(X) - 1$, ở đây $-1 \le x \le 1$.

Để chuyển số x biểu diễn dạng số sang một chuỗi bit ngẫu nhiên X cần một bộ tạo số ngẫu nhiên (SNG: Stochastic Number Generator). SNG bao gồm một mạch so sánh và một thanh ghi dịch hồi tiếp tuyến tính (LFSR: Linear Feedback Shift Register). SNG sẽ tạo ra một bit của chuỗi ngẫu nhiên X trong mỗi chu kỳ đồng hồ. Chuyển một chuỗi bit ngẫu nhiên sang số nhị phân thông thường được thực hiện bằng một bộ đếm.

Gần đây, tính toán ngẫu nhiên đang trở thành chủ đề nghiên cứu được quan tâm cho thực thi các mạch tính toán hiệu năng cao. Tính toán các hàm toán học sử dụng logic ngẫu nhiên đã được thực hiện trong một số nghiên cứu. Ý tưởng chính của các nghiên cứu này là xấp xỉ các hàm toán học dựa trên đa thức Bernstien trong, sử dụng phương pháp máy trạng thái hữu hạn (FSM: Finite State Machines) và sử dụng khai triển Maclaurin kết hợp với qui tắc Horner trong. Trong đó, phương pháp sử dụng khai triển Maclaurin thực hiện trong [97] được các tác giả so sánh với các phương pháp sử dụng đa thức Berntien và FSM và cho thấy đạt được hiệu quả cao hơn. Chương 5 của luận án đề xuất một phương pháp mới thực thi các hàm toán học dựa trên logic ngẫu nhiên và so sánh với các kết quả nghiên cứu trong [97]. Phương pháp đề xuất dựa trên xấp xỉ các hàm toán học bằng phương pháp xấp xỉ phân đoạn tuyến tính đều. Sau đó, thực thi các hàm xấp xỉ sử dụng logic ngẫu nhiên. Các kết quả thực thi sẽ được so sánh với các nghiên cứu trước đó.

5.2. Phương pháp xấp xỉ tuyến tính phân đoạn đều các hàm cho tính toán trên SC

Phương pháp xấp xỉ đề xuất thực hiện xấp xỉ các hàm toán học bằng các hàm phân đoạn tuyến tính đều với số phân đoạn là 8. Các biến đầu vào có định dạng đơn cực trong SC. Trong phân đoạn thứ i hàm xấp xỉ được viết như công thức sau:

$$f(x) \approx a_i x + b_i, \quad \frac{i}{8} \le x < \frac{i+1}{8},\ i = 0 \div 7. \tag{5.1}$$

Lỗi xấp xỉ trên phân đoạn thứ i sẽ là:

$$\varepsilon_i(x) = f(x) - (a_i x + b_i), \quad \frac{i}{8} \le x < \frac{i+1}{8};\ i = 0 \div 7. \tag{5.2}$$

Ngoài ra, để đảm bảo độ chính xác thì các hệ số a_i và b_i có dạng biểu diễn như công thức (5.3) trong đó B biểu diễn độ rộng bit của phần cứng chứa các hệ số.

$$a_i, b_i = N \times 2^B; \quad N, B \in \mathbb{Z} \tag{5.3}$$

Một chương trình được thực hiện để tính toán các hệ số a_i và b_i tối ưu với tiêu chí tối thiểu sai số xấp xỉ cho các hàm toán học khác nhau. Kết quả thể hiện trên Bảng 5.1. Tham số trên Bảng 5.1 là lỗi cực đại xảy ra do xấp xỉ.

5.3. Kiến trúc phần cứng đề xuất

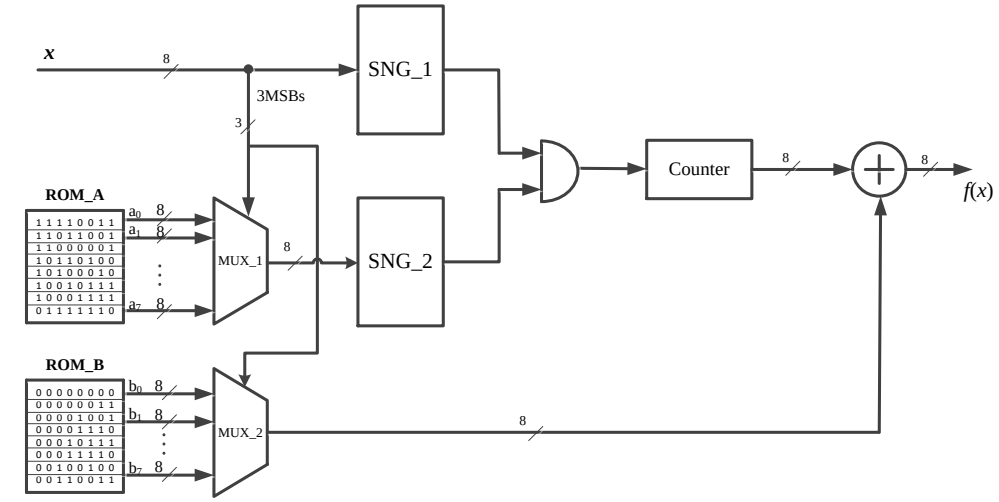
Theo phương pháp đề xuất thì các hàm được xấp xỉ bằng phân đoạn tuyến tính đều và trong mỗi phân đoạn hàm được xấp xỉ bởi một đoạn tuyến tính. Do vậy khi thực thi đòi hỏi một mạch nhân và một mạch cộng. Việc thực thi phép nhân trong các kiến trúc truyền thống sẽ yêu cầu khá nhiều chi phí về phần cứng. Trong logic ngẫu nhiên phép nhân này chỉ cần thực thi bởi một cổng AND, do đó cho phép giảm chi phí phần cứng và tăng tốc độ. Tuy nhiên đòi hỏi các bộ tạo số ngẫu nhiên SNG. Dựa trên phương pháp đề xuất, kiến trúc thực hiện các hàm thể hiện như các Hình 5.1, Hình 5.2, Hình 5.3 và Hình 5.4. Ngoài ra, phương pháp đề xuất cho phép thực thi một kiến trúc chung cho tính toán nhiều hàm toán học như thể hiện trên Hình 5.5.

Trong các kiến trúc phần cứng, các khối ROM_A và ROM_B có kích thước 8×8 bit được sử dụng để chứa các hệ số a_i và b_i. Đầu vào x sẽ được biến đổi thành các số ngẫu nhiên thông qua bộ tạo số ngẫu nhiên SNG_1, đầu ra ROM_A qua một bộ tạo số ngẫu nhiên SNG_2, đầu ra của hai bộ tạo số ngẫu nhiên sẽ được đưa tới đầu vào của cổng AND hai đầu vào để thực hiện phép nhân $a_i \times x$ sau đó đầu ra của mạch AND sẽ được đưa vào bộ đếm để chuyển đổi từ các số ngẫu nhiên sang dạng nhị phân truyền

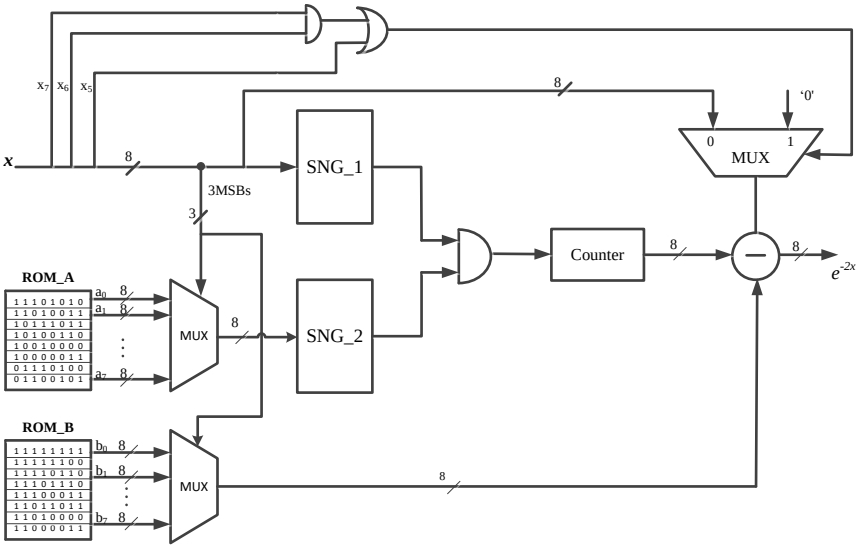
thống. Đầu ra của mạch đếm sẽ được cộng với giá trị của ROM_B để tạo ra kết quả cuối cùng. Ba bit có trọng số lớn nhất của đầu vào x (3MSBs) được sử dụng để điều khiển các bộ ghép kênh MUX_1 và MUX_2 để lựa chọn các giá trị hệ số trong các ROM tương ứng với các phân đoạn.

Bảng 5.1: *Các hệ số tối ưu cho xấp xỉ các hàm bằng 8 phân đoạn.*

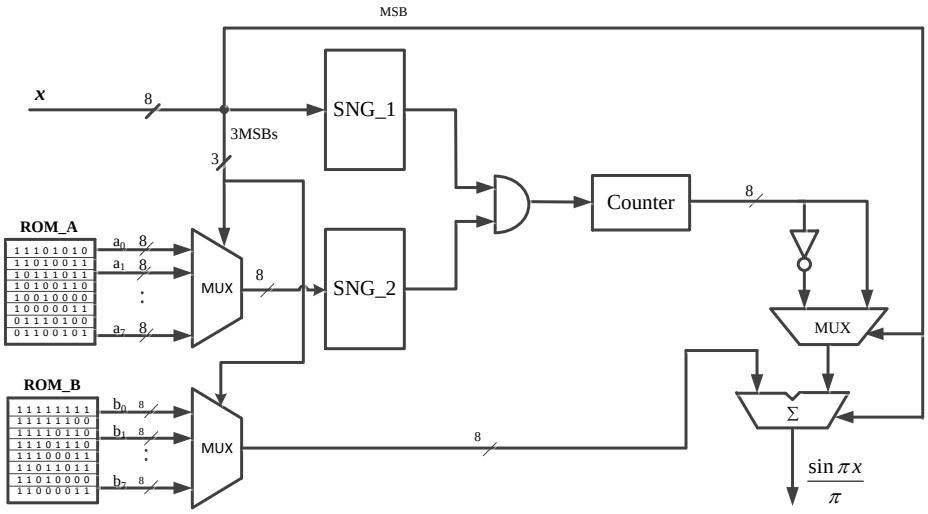
$\ln(1+x)$			$\tanh(x)$			sigmoid		
Đoạn	a_i	b_i	Đoạn	a_i	b_i	Đoạn	a_i	b_i
0	243×2^{-8}	0	0	255×2^{-8}	0	0	64×2^{-8}	128×2^{-8}
1	217×2^{-8}	3×2^{-8}	1	247×2^{-8}	2×2^{-8}	1	64×2^{-8}	128×2^{-8}
2	193×2^{-8}	9×2^{-8}	2	232×2^{-8}	5×2^{-8}	2	63×2^{-8}	128×2^{-8}
3	180×2^{-8}	14×2^{-8}	3	213×2^{-8}	12×2^{-8}	3	63×2^{-8}	128×2^{-8}
4	162×2^{-8}	23×2^{-8}	4	189×2^{-8}	24×2^{-8}	4	57×2^{-8}	131×2^{-8}
5	151×2^{-8}	30×2^{-8}	5	165×2^{-8}	39×2^{-8}	5	57×2^{-8}	131×2^{-8}
6	143×2^{-8}	36×2^{-8}	6	141×2^{-8}	57×2^{-8}	6	52×2^{-8}	135×2^{-8}
7	126×2^{-8}	51×2^{-8}	7	117×2^{-8}	78×2^{-8}	7	50×2^{-8}	137×2^{-8}
$\varepsilon_{\max}=0,0018$			$\varepsilon_{\max}=0,0012$			$\varepsilon_{\max}=7,1\times10^{-4}$		
$\sin x$			$\cos x$			e^{-x}		
Đoạn	a_i	b_i	Đoạn	a_i	b_i	Đoạn	a_i	b_i
0	255×2^{-8}	0	0	-13×2^{-8}	1	0	-234×2^{-8}	256×2^{-8}
1	254×2^{-8}	0	1	-47×2^{-8}	260×2^{-8}	1	-211×2^{-8}	252×2^{-8}
2	245×2^{-8}	2×2^{-8}	2	-79×2^{-8}	268×2^{-8}	2	-187×2^{-8}	246×2^{-8}
3	232×2^{-8}	7×2^{-8}	3	-106×2^{-8}	278×2^{-8}	3	-166×2^{-8}	238×2^{-8}
4	214×2^{-8}	16×2^{-8}	4	-138×2^{-8}	294×2^{-8}	4	-144×2^{-8}	227×2^{-8}
5	201×2^{-8}	24×2^{-8}	5	-162×2^{-8}	309×2^{-8}	5	-131×2^{-8}	219×2^{-8}
6	178×2^{-8}	41×2^{-8}	6	-186×2^{-8}	327×2^{-8}	6	-116×2^{-8}	208×2^{-8}
7	144×2^{-8}	71×2^{-8}	7	-211×2^{-8}	349×2^{-8}	7	-101×2^{-8}	195×2^{-8}
$\varepsilon_{\max}=0,002$			$\varepsilon_{\max}=0,0015$			$\varepsilon_{\max}=0,0013$		
e^{-2x}			$\sin(\pi x)/\pi$			$\log_2(1+x)$		
Đoạn	a_i	b_i	Đoạn	a_i	b_i	Đoạn	a_i	b_i
0	-450×2^{-8}	255×2^{-8}	0	255×2^{-8}	0	0	351×2^{-8}	0
1	-355×2^{-8}	243×2^{-8}	1	254×2^{-8}	0	1	310×2^{-8}	5×2^{-8}
2	-273×2^{-8}	223×2^{-8}	2	245×2^{-8}	2×2^{-8}	2	282×2^{-8}	12×2^{-8}
3	-213×2^{-8}	200×2^{-8}	3	232×2^{-8}	7×2^{-8}	3	258×2^{-8}	21×2^{-8}
4	-168×2^{-8}	178×2^{-8}	4	-214×2^{-8}	16×2^{-8}	4	234×2^{-8}	33×2^{-8}
5	-128×2^{-8}	153×2^{-8}	5	-201×2^{-8}	24×2^{-8}	5	217×2^{-8}	44×2^{-8}
6	-100×2^{-8}	132×2^{-8}	6	-178×2^{-8}	41×2^{-8}	6	205×2^{-8}	53×2^{-8}
7	-73×2^{-8}	108×2^{-8}	7	-144×2^{-8}	71×2^{-8}	7	190×2^{-8}	66×2^{-8}
$\varepsilon_{\max}=0,0039$			$\varepsilon_{\max}=0,002$			$\varepsilon_{\max}=0,0018$		



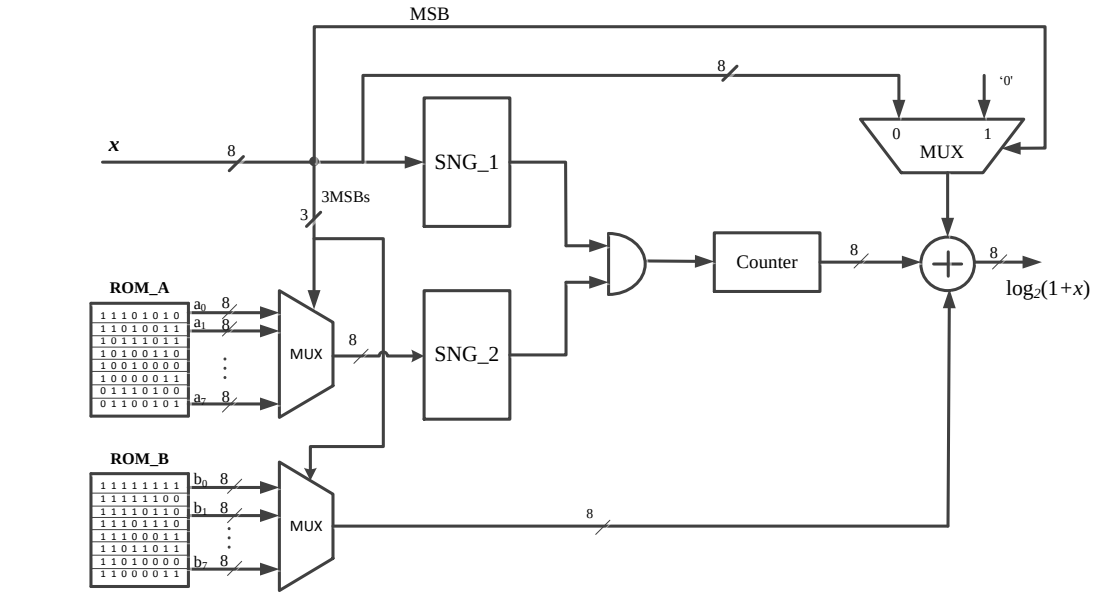
H**ình 5.1:** *Kiến trúc phần cứng thực thi hàm* $\ln(1+x), \tanh x, sigmoid$ *và* $\sin x$



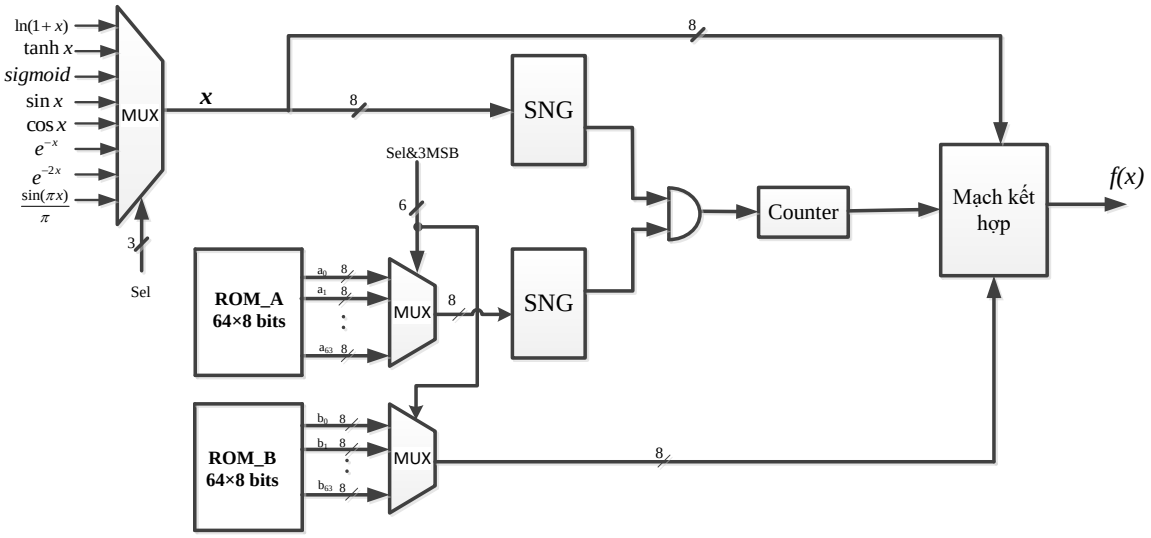
Hình 5.2: *Kiến trúc phần cứng thực thi hàm* e^{-2x}.



Hình 5.3: *Kiến trúc phần cứng thực thi hàm* $\sin(\pi x)/\pi$.



Hình 5.4. *Kiến trúc phần cứng thực thi hàm* $\log_2(1+x)$.



Hình 5.6: *Kiến trúc phần cứng thực thi nhiều hàm toán học.*

5.4. Kết quả thực thi và so sánh

Các kiến trúc đề xuất tính toán các hàm toán học khác nhau được tổng hợp trên FPGA và ASIC. Tất cả các hàm được thực thi đều có đầu vào và đầu ra đều có định dạng đơn cực và được biểu diễn bằng 8 bit. Các kiến trúc được đề xuất trong [97] cũng được thực thi lại để so sánh. Bảng 5.2 thể hiện kết quả thực thi trên FPGA Virtex6. Bảng 5.3 thể hiện kết quả tổng hợp bằng Synopsys Design Compiler cho các hàm khác nhau của phương pháp đề xuất và các kiến trúc so sánh trên thư viện SAED công nghệ CMOS 90 nm. Các kết quả thực thi cho thấy phương pháp đề xuất đạt được hiệu quả cao hơn so với các kiến trúc theo phương pháp trong [97] về cả về tài nguyên, tốc độ và công suất tiêu thụ.

Bảng 5.2: *Kết quả tổng hợp trên FPGA.*

Hàm	Đề xuất			Trong [97]			
	FPGA LUT	Độ trễ (ns)	f_{max} (MHz)	Bậc	FPGA LUT	Độ trễ (ns)	f_{max} (MHz)
$\ln(1+x)$	79	1,885	530,504	5	86	2,287	437,192
$\tanh x$	78	1,885	530,504	7	87	2,290	436,624
sigmoid	75	2,288	437,006	5	96	2,287	437,192
$\sin x$	77	2,288	437,006	7	88	2,290	436,624
$\cos x$	78	2,288	437,006	8	96	2,287	437,192
e^{-x}	77	1,885	530,504	5	87	2,287	437,192
e^{-2x}	94	2,288	437,006	7	99	2,290	436,624
$\frac{\sin(\pi x)}{\pi}$	81	2,288	437,006	9	64	2,288	437,006
$\log_2(1+x)$	87	2,288	437,006	-	-	-	-
Đa hàm	133	2,288	437,006	-	-	-	-

Bảng 5.3: *Kết quả thực thi trên ASIC.*

Hàm	Đề xuất			Trong [97]			
	Diện tích (μm^2)	Độ trễ (ns)	Công suất (μW)	Bậc	Diện tích (μm^2)	Độ trễ (ns)	Công suất (μW)
$\ln(1+x)$	3459	1,69	80,95	5	5381	1,74	101,24
$\tanh x$	3449	1,69	80,34	7	4933	1,74	83,51
sigmoid	3589	1,70	97,89	5	5350	1,74	189,17
$\sin x$	3741	1,70	60,67	7	4933	1,74	83,51
$\cos x$	3746	1,70	65,87	8	5387	1,74	103,80
e^{-x}	3469	1,69	78,43	5	5870	1,74	121,73
e^{-2x}	4227	1,70	135,86	7	6430	1,74	256,55
$\frac{\sin(\pi x)}{\pi}$	4188	1,70	70,30	9	4881	1,74	82,17
$\log_2(1+x)$	4072	1,70	132,32	-	-	-	-
Đa hàm	6949	1,77	298,65	-	-	-	-

5.5. Khảo sát và hiệu chỉnh sai số

Để đánh giá sai số trong tính toán các hàm trên phần cứng đề xuất, một chương trình kiểm tra được viết để nhận được dữ liệu đầu ra cho từng hàm. Với các kiến trúc đề xuất có dữ liệu đầu vào và đầu ra có định dạng 8 bit, dữ liệu đầu ra gồm 256 giá trị tương ứng với 256 giá trị đầu vào. Sau khảo sát bằng cách điều chỉnh các hệ số xấp xỉ để đạt được sai số nhỏ hơn. Bảng 5.4 thể hiện sai số của phương pháp đề xuất và có so sánh với các kết quả trong [97].

Bảng 5.4: *So sánh lỗi của phương pháp đề xuất với các phương pháp khác.*

Hàm		Đề xuất	Khai triển Maclaurin			Đa thức Bernstien			FSM
$\sin x$	Bậc	-	3	5	7	3	5	7	8 trạng thái
	MAE	0,0026	0,0016	0,0033	0,0034	0,0136	0,0088	0,0066	0,0025
$\cos x$	Bậc	-	2	4	6	2	4	6	8 trạng thái
	MAE	0,0035	0,0082	0,0025	0,0023	0,0356	0,0178	0,0120	0,0053
$\ln(1+x)$	Bậc	-	5	6	7	5	6	7	8 trạng thái
	MAE	0,0026	0,0141	0,0109	0,0081	0,0090	0,0076	0,0066	0,0186
$\tanh x$	Bậc	-	3	5	7	3	5	7	8 trạng thái
	MAE	0,0029	0,0178	0,0175	0,0140	0,0182	0,0110	0,0082	0,0351
e^{-x}	Bậc	-	4	5	6	4	5	6	8 trạng thái
	MAE	0,0027	0,0018	0,0008	0,0008	0,0130	0,0103	0,0086	
e^{-2x}	Bậc	-	5	6	7	6	7	8	8 trạng thái
	MAE	0,0027	0,0019	0,0011	0,0009	0,0195	0,0170	0,0875	0,0432
sigmoid	Bậc	-	5			-			8 trạng thái
	MAE	0,0024	0,0046			-			0,0198
$\sin(\pi x)/\pi$	MAE	0,0027	-			-			-
$\log_2(1+x)$	MAE	0,0025	-			-			-

5.6. Kết luận chương 5

Chương 5 của luận án trình bày một phương pháp thiết kế các lõi phần cứng tính toán các hàm toán học dựa trên các kỹ thuật xấp xỉ phân đoạn tuyến tính đều và sử dụng các logic ngẫu nhiên. Kiến trúc phần cứng tính toán các hàm toán học phổ biến đã được thiết kế dựa trên phương pháp đề xuất. Các kết quả thực thi cho thấy, các kiến trúc đề xuất đã cải thiện hiệu năng so với các phương pháp trước đó

KẾT LUẬN VÀ HƯỚNG NGHIÊN CỨU TƯƠNG LAI

Trong luận án này, nghiên cứu sinh đã tiến hành nghiên cứu những kiến thức cơ bản về các phương pháp thực thi phần cứng tính toán các hàm toán học cơ sở. Đồng thời xem xét đặc tính của các ứng dụng xử lý tín hiệu số. Trên cơ sở đó, luận án này tập trung vào nghiên cứu các lõi phần cứng hiệu quả cho tính toán các hàm toán học điển hình ứng dụng trong xử lý tín hiệu số. Một số kết quả đạt được của luận án có thể tóm tắt như sau:

- Đề xuất phương pháp xấp xỉ hàm logarithm nhị phân và hàm sin dựa trên xấp xỉ phân đoạn tuyến tính đều kết hợp với LUT bù sai số xấp xỉ. Các kiến trúc phần cứng dựa trên phương pháp đề xuất đạt được độ phức tạp thấp.

- Đề xuất 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. Các thực thi một số hàm toán học điển hình theo phương pháp đề xuất đã cho thấy hiệu quả về tốc độ. Vì vậy, kiến trúc đề xuất là phù hợp với các ứng dụng đòi hỏi tốc độ thực thi cao.

- Đề xuất một phương pháp tính toán các hàm toán học dựa trên tính toán ngẫu nhiên kết hợp với xấp xỉ phân đoạn tuyến tính đều. Dựa trên phương pháp này 9 hàm toán học sử dụng phổ biến trong ứng dụng xử lý tín hiệu số và mạng nơ-ron đã được thực thi. Đồng thời một kiến trúc chung cho tính toán nhiều hàm toán học dựa trên phương pháp đề xuất cũng được thiết kế và thực thi. Các kết quả tổng hợp trên FPGA và ASIC cho thấy hiệu quả của phương pháp đề xuất so với các nghiên cứu tương tự.

Dựa trên các kết quả nghiên cứu trong luận án này, một số hướng nghiên cứu có thể phát triển trong tương lai là:

- Các hệ thống DSP hiện đại yêu cầu các thành phần tính toán có tài nguyên thấp và hiệu năng cao vì vậy các kiến trúc hiệu quả cho các hàm toán học khác cần tiếp tục nghiên cứu và nâng cao hiệu năng cho các ứng dụng chuyên biệt.

- Phương pháp kết hợp xấp xỉ các hàm bằng phân đoạn tuyến tính với một LUT có kích thước nhỏ để bù sai số cho thấy hiệu quả với các ứng dụng đòi hỏi chi phí phần cứng thấp. Có thể phát triển phương pháp bằng cách tìm một thuật toán chung có tính hệ thống cho thực thi các hàm toán học. Trong đó, các tham số như độ rộng bit dữ liệu đầu vào,

kích thước của LUT bù sai số, số phân đoạn xấp xỉ sẽ được tối ưu theo độ chính xác yêu cầu.

- Nghiên cứu ứng dụng các lõi tính toán đề xuất trong luận án vào các ứng dụng như: xử lý tiếng nói, xử lý ảnh, đồ họa…

- Các lõi tính toán được thiết kế trong luận án này dựa trên SC được thực hiện trên các hàm có đầu vào và đầu ra dạng đơn cực. Do vậy có thể tiếp tục nghiên cứu thực thi với các hàm có đầu vào và đầu ra dạng lưỡng cực. Hơn nữa, sử dụng SC là giải pháp hứa hẹn trong nhiều ứng dụng như mạng nơ-ron, xử lý ảnh…Do vậy, nghiên cứu thiết kế các lõi tính toán dựa trên SC cho các ứng dụng này là một hướng có thể tiếp tục nghiên cứu phát triển. Đặc biệt các kiến trúc song song dựa trên SC là một hướng nghiên cứu mới có thể phát triển cho nhiều ứng dụng.