Thiết kế FFT 2048-điểm trên FPGA dựa trên thuật toán CORDIC xoay góc thích nghi với độ chính xác dấu chấm động đơn

- **Trương Thị Như Quỳnh**
- **Võ Thị Phương Thảo**
- **Hoàng Trọng Thức**
- **Lê Đức Hùng**

Trường Đại Học khoa học Tự nhiên, ĐHQG-HCM

(Bài nhận ngày 26 tháng 12 năm 2016, nhận đăng ngày 30 tháng 10 năm 2017)

TÓM TẮT

Ở bài báo này, thiết kế FFT 2048-điểm dựa trên thuật toán CORDIC xoay góc thích nghi với độ chính xác dấu chấm động đơn được thực thi trên FPGA. Thiết kế được xây dựng và kiểm tra trên chip FPGA Stratix V của Altera. Thực nghiệm trên FPGA cho tần số tối đa 102,55 MHz, hiệu suất 8424,382 FFTs/s, có nghĩa trong một giây thiết kế này sẽ thực thi được khoảng 8424 lần 2048-điểm FFT, tài nguyên tiêu tốn là 76.282 ALUTs và 15.687 thanh ghi. Kết quả cho ra độ chính xác 5,889E-06 sai số toàn phương trung bình (Mean-Square-Error - MSE).

Từ khóa: *FFT, CORDIC xoay góc thích nghi, dấu chấm động đơn*

MỞ ĐẦU

Năm 1965, Cooley và Tukey lần đầu tiên giới thiệu về mô hình Fast Fourier Transform (FFT) [1], một phương pháp tính nhanh của Discrete Fourier Transform (DFT). Ngày nay, biến đổi FFT và biến đổi ngược IFFT được xem là cốt lõi và được sử dụng rộng rãi trong các hệ thống truyền thông và xử lý tín hiệu số. Các ứng dụng có thể kể đến như *WiMax* [2], *3GGPLTE* [3], trong hệ thống truyền thông đa phương tiện *MIMO* [4], *CDMA* [5], hay xử lý tín hiệu SAR trong các ứng dụng ra-đa. Tuy nhiên, thuật toán FFT truyền thống không đủ để đáp ứng các yêu cầu cao như hiện nay.

Thuật toán FFT dấu chấm tĩnh truyền thống cho tỉ lệ lỗi thấp, tốc độ cao và chi phí thấp. Bên cạnh đó, khi thực hiện chúng sẽ bỏ qua các bit có trọng số thấp nhất (Least Significant Bit - LSB) trong tính toán. Điều này phù hợp với các ứng dụng không đòi hỏi độ chính xác cao như các hệ thống về âm thanh, truyền thông DVB-T/DVB, WLAN [6]. Tuy nhiên, các hệ thống ngày nay không chỉ cần độ chính xác cao mà còn đòi hỏi miền dữ liệu lớn như các hệ thống xử lý y khoa như *Fourier-Domain Optical Coherence Toomography (FD-OCT*) [7] hay các hệ thống truyền thông không dây. Để đáp ứng yêu cầu về độ chính xác cao và miền dữ liệu lớn, các nghiên cứu về FFT dấu chấm động là cần thiết.

Thuật toán FFT dấu chấm động đơn được chia ra làm hai loại là FFT dựa trên bộ nhớ (*Mem - FFT*) [8, 9] và FFT tính liên tục (*Continuous Flow - CF-FFT*) [10, 11]. Đối với thiết kế Mem-FFT sử dụng ít tài nguyên logic và dễ tương thích với các hệ thống có CPU được tích hợp sẵn. Thiết kế bao gồm các bộ tính Butterfly, bộ tạo địa chỉ, bộ điều khiển và các khối vùng nhớ. Trong khi thiết kế dựa trên CF-FFT cho các ưu điểm về tốc độ, có thể áp dụng kỹ thuật pipeline cho ra hiệu suất truyền lớn, nhưng tài nguyên sử dụng lớn. Thiết kế CF-FFT thực hiện cho FFT N-điểm có cấu trúc $log_2(N)$ tầng pipeline, trong đó mỗi tầng gồm các bộ đệm dịch thanh ghi và các bộ tính Butterfly. Dù kiến trúc dựa trên CF-FFT hay

Mem-FFT thì sự hạn chế về tốc độ của thiết kế vẫn ở tại bộ tính Butterfly hay chính là các bộ nhân số phức (Twiddle Factor - TF). Hơn thế nữa, để cải thiện độ chính xác của thuật toán thì TF đóng vai trò chủ chốt.

Thiết kế TF được chia ra làm hai kiến trúc là: thiết kế sử dụng bảng tra (Lookup Table - LUT) và cách tính trực tiếp. Thiết kế FFT sử dụng bảng tra LUT để lưu trữ các hằng số được tính sẵn trong ROM (Read Only Memmory) và các hằng số này được gọi ra để thực hiện tính toán bộ Butterfly khi thích hợp. Một số các thí dụ sử dụng thiết kế này được nêu ở [12, 13], và [14]. Thiết kế FFT dựa trên LUT cho độ chính xác cao dù hệ thống sử dụng định dạng dữ liệu dấu chấm tĩnh hay dấu chấm động. Bên cạnh đó, nó có độ trễ thấp và tài nguyên tính toán logic tương đối nhỏ. Tuy nhiên, hạn chế của kiến trúc này là đối với các tính toán FFT số điểm lớn sẽ yêu cầu dung lượng của ROM quá lớn, dẫn đến khó khăn cho việc ASIC. Có một số nghiên cứu đưa ra giải pháp về vấn đề này như [10] trình bày giải pháp để tối thiểu yêu cầu bộ nhớ cho CF-FFT [15], đưa ra thuật toán phân hủy cây nhị phân để tối thiểu số lượng TF được lưu trữ trong ROM. Vì vậy, kiến trúc FFT sử dụng LUT chỉ phù hợp với các thiết kế FFT ít điểm. Đối với các thiết kế FFT nhiều điểm, giải pháp tính trực tiếp TF thường được sử dụng hơn. Phương pháp tính TF trực tiếp được đề xuất ở bài báo này dựa trên sử dụng thuật toán COordinate Rotation DIgital Computer (CORDIC). Thuật toán này được đề xuất từ năm 1984 như [16], và cho đến nay vẫn được tiếp tục nghiên cứu và cải tiến như [8, 17], và [18]. Phương pháp CORDIC truyền thống đáp ứng tốt đối với kiến trúc FFT dấu chấm tĩnh do tối ưu được các phép nhân về các phép dịch thanh ghi và cộng trừ. Tuy nhiên đối với kiến trúc FFT ở dạng dấu chấm động, CORDIC truyền thống không tốt bởi việc thiết kế cộng trừ không đơn giản và phải thực hiện tính lặp nhiều lần để cho ra kết quả chính xác. Vì vậy sử dụng CORDIC truyền thống cho thiết kế FFT dạng dấu chấm động bị hạn chế với độ trễ cao và hiệu suất thấp. Do đó, để khắc phục hạn chế này, một thuật toán CORDIC cải tiến có độ hội tụ nhanh là cần thiết và thuật toán CORDIC cả tiến được dùng ở bài báo này chính là thuật toán CORDIC xoay góc thích nghi.

Ở bài báo này, thiết kế FFT 2048-điểm dựa trên CORDIC xoay góc thích nghi với độ chính xác dấu chấm động đơn được đề xuất và hiện thực trên FPGA. Thiết kế sử dụng kiến trúc FFT cơ số 2 đa đường (Radix-2 MDC). Thiết kế này được thử nghiệm và kiểm tra trên chip FPGA Altera Stratix V. Kết quả thực nghiệm cho thấy tần số tối đa là 102,55 MHz. Tài nguyên tiêu tốn là 76.282 ALUTs và 15.687 thanh ghi. Hiệu suất đạt được 8424,382 FFTs/s, nghĩa là thiết kế được thực thi khoảng 8424 lần 2048 điểm FFT trong một giây. Độ chính xác của thiết kế được kiểm định bằng việc triển khai sai số bình phương trung bình (Mean-Square-Error – MSE), phần nghìn tỉ lệ sai số cực đại (part-per thousand – ppt maximum error-ratio). Kết quả thu được là 5,889E-06 MSE và 2,456 ppt tỉ lệ sai số cực đại.

NỀN TẢNG KIẾN THỨC

Biến đổi nhanh Fourier

Biến đổi nhanh Fourier (Fast Fourier Transform - FFT) là một phương pháp biến đổi nhanh của Biến đổi rời rạc Fourier (Discrete Fourier Transform - DFT). Điều đó có nghĩa là về phương pháp và kết quả tính là tương tự nhau, nhưng điểm khác nhau là FFT yêu cầu độ phức tạp là $Nlog_2(N)$ trong khi DFT cần đến N^2. Công thức FFT được cho bởi phương trình (1).

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn} \text{ , } k = 0,1...N-1 \quad (1)$$

Trong đó, x(n) là dữ liệu ngõ vào ở miền liên tục thời gian, X(k) là giá trị ngõ ra ở miền tần số, W_N^{kn} là trọng số nhân và được tính theo công thức (2).

$$W_n^{kn} = exp\left(-\frac{j2n\pi k}{N}\right) = cos\left(\frac{2\pi nk}{N}\right) - jsin(\frac{2\pi nk}{N}) \quad (2)$$

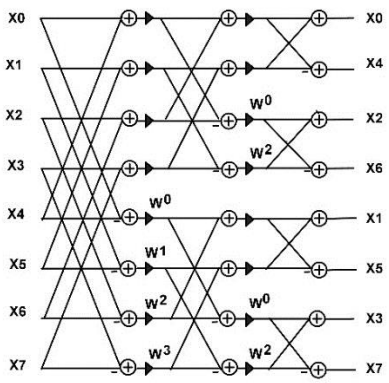
Biến đổi nhanh Fourier cơ số 2

FFT cơ số 2 (Radix-2) là thuật toán cơ bản và thông dụng. Nhờ đó mà thiết kế FFT cơ số 2 có phần tiết kiệm tài nguyên nhờ vào sự đơn giản đó. Ý tưởng của thiết kế Radix-2 là chuỗi X(k) có N-điểm được chia đôi thành chuỗi với *N*/2 điểm chẵn X(2 m), và *N*/2 điểm lẻ X(2 m+1), được tính độc lập. Khi đó việc thực thi phương pháp Radix-2 sẽ chuyển từ công thức (1) sang sử dụng công thức (3) và Hình 1 là thí dụ minh họa cho thiết kế FFT cơ số 2 với 8 điểm.

$$(2k) = \sum_{n=0}^{\frac{N}{2}-1} \left[x(n) + x\left(n + \frac{N}{2}\right)\right] W_{N/2}^{kn}$$

$$X(2k+1) = \sum_{n=0}^{\frac{N}{2}-1} \left\{\left[x(n) - x\left(n + \frac{N}{2}\right)\right] W_N^n\right\} W_{N/2}^{kn}$$

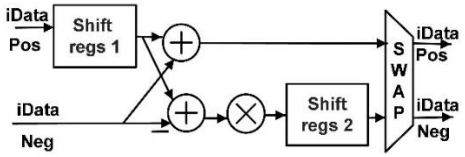
với k = 0,1…N/2 – 1 (3)



Hình 1. Mô hình 8-điểm FFT Radix2

Kiến trúc FFT đa đường

Hình 2 thể hiện ý tưởng thiết kế đa đường (Multi-path Delay Commutator - MDC) mỗi tầng có hai bộ đệm dữ liệu, bộ đệm thứ hai (Shift_regs2) có dung lượng bằng một nửa dung lượng bộ đệm thứ nhất (Shift_regs1), và bộ đệm thứ nhất có dung lượng bằng dung lượng bộ đệm thứ hai của tầng trước đó. Trong thiết kế Radix2 - MDC, dữ liệu ngõ vào sau khi được đệm qua Shift_regs1 và tính toán Butterfly sẽ cho ra hai đường dữ liệu. Một đường dữ liệu đi thẳng đến ngõ ra, còn đường còn lại thực hiện việc nhân số phức W rồi được đệm qua Shift_regs2 trước khi cho ra kết quả ngõ ra.



Hình 2. Ý tưởng kiến trúc MDC

Bộ nhân số phức sử dụng xoay góc thích nghi

Thuật toán bộ nhân số phức xoay góc thích nghi (Adaptive Angle Recoding CORDIC - AARC) là phương pháp thay vì sử dụng hết tất cả các góc hằng số thì nó chỉ chọn một số góc trong tất cả các góc đó mà thôi, nhưng vẫn đảm bảo đạt được kết quả tương tự. Áp dụng phương pháp này làm giảm độ trễ và chi phí tài nguyên thấp. Ví dụ, nếu chọn một góc là 25^0 với 16 góc không đổi theo như Bảng 1, thì chuỗi các góc của bộ nhân CORDIC truyền thống và chuỗi các góc của AARC được biểu diễn lần lượt ở phương trình (4) và (5), sai số của CORDIC truyền thống là 12,9E-04, trong khi sai số của AARC là 6,99E-04.

$$\theta_0 - \theta_1 + \theta_2 - \theta_3 - \theta_4 + \theta_5 + \theta_6 + \theta_7 + \theta_8 - \theta_9 - \theta_{10} + \theta_{11} + \theta_{12} - \theta_{13} + \theta_{14} + \theta_{15} = 24{,}998708138 \quad (4)$$

$$\theta_1 - \theta_5 + \theta_8 + \theta_{15} = 25{,}000699597 \quad (5)$$

$$ci = \begin{cases} \frac{\theta_i + \theta_{i+1}}{2}, & 0 \le i \le N-2 \\ \frac{\theta_i}{2}, & i < 0 \text{ và } i > N-2 \end{cases} \quad (6)$$

Hình (3) thể hiện chương trình giả mô tả cách thực hiện thuật toán AARC, ứng với mỗi lần lặp i chọn một góc θ_i sao cho khi thực hiện góc z_i sẽ hội tụ về 0 nhanh nhất. Để có thể lựa chọn được góc θ_i, thuật toán này dựa trên khái niệm tham số C, được tính theo công thức (6) và được thể hiện ở Bảng 1. Mỗi lần xoay sẽ có một hệ số k_i, đối với số góc xoay cố định của CORDIC truyền thống thì hệ số K tích lũy là không đổi,

trong khi AARC thì chỉ chọn một vài góc xoay nên hệ số K tích lũy qua các lần lặp là không giống nhau. Hệ số k_i ứng với lần lặp thứ i được biểu diễn ở Bảng 1. Hay nói cách khác, thuật toán AARC cho hệ số K động thay vì hằng số như trong CORDIC truyền thống.

Bảng 1. Các giá trị θ, C, và K trong thuật toán AARC

i	θ	C	K
0	45,000000000	35,782525588	0,707106781
1	26,565051177	20,300647322	0,894427191
2	14,036243468	10,580629908	0,970142500
3	7,125016349	5,350675362	0,992277877
4	3,576334375	2,683122491	0,998052578
5	1,789910608	1,342542159	0,999512076
6	0,895173710	0,671393940	0,999877952
7	0,447614171	0,335712335	0,999969484
8	0,223810500	0,167858088	0,999992371
9	0,111905677	0,083929284	0,999998093
10	0,055952892	0,041964672	0,999999523
11	0,027976453	0,020982340	0,999999881
12	0,013988227	0,010491170	0,999999970
13	0,006994114	0,005245585	0,999999993
14	0,003497057	0,002622792	0,999999998
15	0,001748528	0,000874264	0,999999999

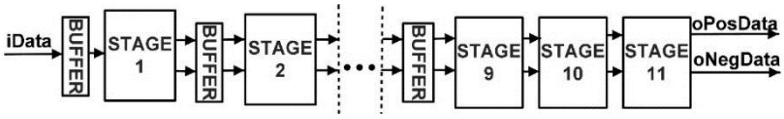
$i = j = 0; x_0 = 1; y_0 = 0; K = 1;$
while $|z_j| > c_{15}$ and $i < N$ **do**
  if $|z_j| \in (c_{i+1}; c_i]$ **then**
    $z_{j+1} = z_j - sign(z_j)\theta_i;$
    $x_{j+1} = x_j - sign(z_j)y_j 2^{-i};$
    $y_{j+1} = y_j + sign(z_j)x_j 2^{-i};$
    $K = K \times k_i; j = j + 1;$
  end if
  $i = i + 1$
end while
$X = x_j \times K; Y = y_j \times K;$

Hình 3. Đoạn chương trình mô phỏng phương pháp AARC

HIỆN THỰC THIẾT KẾ

Hiện thực thiết kế FFT cơ số 2 đa đường

Mô hình thiết kế FFT cơ số 2 đa đường (FFT Radix-2 MDC) 2048-điểm được thể hiện ở hình (4). Khi dữ liệu truyền đi giữa các tầng, việc thực thi các bộ tính giữa các tầng không chỉ trong một xung clock mà có thể là nhiều clock. Khi dữ liệu trước đang được thực thi thì các dữ liệu đến sau cần được giữ lại chờ thực thi xong các dữ liệu trước. Vì vậy, sử dụng bộ đệm Buffer để đảm bảo dữ liệu được truyền hoặc giữ đúng thời điểm.



Hình 4. Mô hình 11 tầng của Radix-2 MDC 2048-điểm

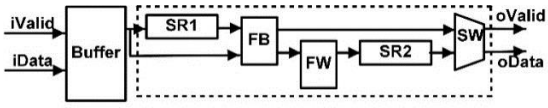
Mô hình khối của mỗi tầng của kiến trúc FFT Radix-2 MDC được biểu diễn ở Hình 5. Nguyên lý hoạt động của một tầng được thể hiện qua 4 phiên hoạt động như sau:

Phiên 1: Bắt đầu phiên truyền, *N*/2 dữ liệu đầu tiên được lưu vào bộ đệm Shift_regs1. Lúc này không có dữ liệu ngõ ra. Tại mỗi xung clock, phiên truyền luôn kiểm tra các tín hiệu điều khiển xem có tiếp tục truyền hoặc giữ dữ liệu.

Phiên 2: *N*/4 dữ liệu tiếp theo thông qua bộ đệm Buffer đến gặp trực tiếp dữ liệu bị trễ ở bộ đệm Shift_regs1, thực thi tại FloatButterfly cho kết quả phần (+) đi thẳng đến tầng tiếp theo, phần (-) được đưa đến bộ nhân số phức FloatW cho kết quả và được lưu vào bộ đệm Shift_regs2.

Phiên 3: *N*/4 dữ liệu còn lại đến gặp dữ liệu trễ ở bộ đệmShift_regs1 thực hiện tính toán tại FloatButterfly. Phần (+) tiếp tục đi đến tầng tiếp theo, trong khi phần (-) lưu vào Shift_regs2 đồng thời đẩy phần dữ liệu cũ được lưu trong Shift_regs2 đến tầng tiếp theo.

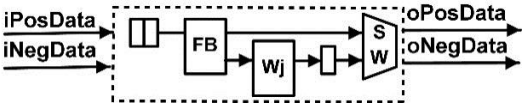
Phiên 4: Lúc này *N*/4 dữ liệu cuối cùng còn lại trong Shift_regs2 ở phiên 3 được đưa đến tầng tiếp theo để kết thúc phiên làm việc của toàn quy trình.



Hình 5. Mô hình chi tiết của một tầng

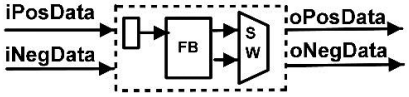
Theo như mô hình được thể hiện ở Hình 4, có thể thấy được các tối ưu ở tầng đầu tiên, tầng kề cuối và tầng cuối. Tối ưu ở tầng đầu tiên do dữ liệu ngõ vào là số nguyên dạng dấu chấm động đơn 32-bit và chỉ có phần thực, nên phần ảo được gán là 0, nhờ đó tối ưu được bộ nhân số phức triệt tiêu đi phần tính số ảo.

Tối ưu ở tầng kề cuối được mô tả ở Hình 6. Theo kiến trúc Radix- 2, tầng kề cuối chỉ nhân với *Wj*. Bộ nhân *Wj* này chính là đổi chỗ và đảo dấu giữa phần thực và phần ảo, nên thiết kế được tối ưu thành các bộ Multiplexer và đảo dấu, do đó tiết kiệm đáng kể tài nguyên dành cho bộ nhân số phức ở tầng này.



Hình 6. Mô hình tối ưu tầng kề cuối

Tối ưu ở tầng cuối được thể hiện ở Hình 7 do kiến trúc tầng cuối không nhân với W. Do đó tầng cuối chỉ thực hiện bộ cộng trừ ở FloatButterfly và cho ra kết quả cuối cùng không thực hiện phép nhân số phức.



Hình 7. Mô hình tối ưu tầng cuối.

Hiện thực thiết kế bộ nhân xoay góc thích nghi

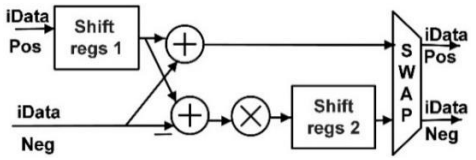
Mô hình thuật toán bộ nhân xoay góc thích nghi – AARC được mô tả ở Hình 8 bao gồm bộ chọn góc xoay (RotSel), bộ cộng (FALU_XY), bộ nhân hệ số tích lũy *ki* (Fmul_ki) và bộ nhân khử hệ số K và tối ưu ngõ ra (FmulK_Norm).

Bộ chọn góc xoay RotSel nhận dữ liệu từ ngõ vào iZ và tiến hành so sánh để tìm ra góc xoay thích hợp, cho dữ liệu ngõ ra gồm các thông tin của góc xoay như dấu, vị trí của góc được chọn và góc đó đã là góc cuối cùng hay chưa, ứng với Bảng 1.

Bộ cộng FALU_XY nhận dữ liệu iX, iY từ đường dữ liệu ngõ vào của bộ nhân và góc xoay được chọn từ bộ RotSel để thực hiện việc cộng tích lũy. Bộ cộng cho ra đường dữ liệu thực oY và phần ảo oX được định dạng 1 bit dấu (sign), 8 bit phần mũ (exponent), 24 bit phần giá trị (mantissa).

Bộ nhân hệ số *ki* Fmul_ki được thực thi đồng thời cùng với bộ cộng FALU_XY. Ứng với mỗi lần lặp, bộ nhân FMul_ki nhận dữ liệu góc xoay được chọn ở RotSel và cho ngõ ra là hệ số oK được tính tích lũy từ các hệ số *ki* ứng với Bảng 1. Định dạng hệ số K chỉ có 24-bit mantissa không cần bit dấu và phần mũ. Nguyên nhân là do hệ số K luôn là số dương nên bit dấu được lược bỏ trong thiết kế. Ngoài ra Bảng 1 cho thấy rằng khi i càng lớn thì K càng tiến về 1, do đó giá trị K không thể vượt qua 1 bằng việc nhân tích lũy. Vì vậy, với 24-bit mantissa là đủ cho biểu diễn hệ số K và giá trị phần mũ có thể cố định và lược bỏ trong thiết kế.

Bộ nhân khử hệ số K và tối ưu ngõ ra FmulK_Norm nhận dữ liệu ngõ vào gồm dữ liệu từ bộ FALU_XY và hệ số K của bộ nhân Fmul_ki. Ở đây, FmulK_Norm tiến hành khử K thông qua phép nhân nghịch đảo. Kết quả sau khi khử K được quy chuẩn về định dạng IEEE754 với 32-bit dấu chấm động gồm 1 bit dấu, 8 bit mũ và 23 bit phần giá trị.



Hình 8. Mô hình bộ nhân CORDIC sử dụng thuật toán xoay góc thích nghi

KẾT QUẢ

Thiết kế kiến trúc FFT cơ số 2 đa đường được xây dựng và kiểm tra trên chip FPGA

Stratix V của Altera. Kết quả thực nghiệm được trình bày trong Bảng 2.

Thiết kế cho tần số tối đa Fmax là 102,55MHz với tốc độ tương ứng là 8424,382 FFTs/s, có nghĩa là trong một giây thiết kế này thực thi khoảng 8424 lần 2048-điểm FFT. Tài nguyên sử dụng trên FPGA là 76.282 tài nguyên logic (ALUTs) và 15.687 thanh ghi (Register). Độ trễ của thiết kế cần 12.173 clock.

Đối với độ chính xác, sai số bình phương tối thiểu (Mean Square Error – MSE) và tỉ lệ lỗi tối đa là các thông số cần thiết trong việc đánh giá độ chính xác của các hệ thống xử lý tín hiệu số. Thiết kế được đề xuất cho kết quả thực nghiệm về độ chính xác là 5,889E-06 MSE và 2,456 mỗi phần nghìn - ppt tỉ lệ lỗi tối đa.

Bảng 2. Kết quả thực nghiệm của kiến trúc FFT 2048-điểm Radix-2 MDC

Các thông số	Kết quả thực nghiệm
ALUTs	76.282
Registers	15.687
Fmax (MHz)	102,55
Latency	12.173
Throughput (FFTs/s)	8424,382
MSE	5,889E-06
Error-ratio Max (ppt)	2,456

KẾT LUẬN

Bài báo này trình bày về thiết kế FFT 2048-điểm dựa trên thuật toán CORDIC xoay góc thích nghi với độ chính xác dấu chấm động đơn. Thiết kế được thực thi trên chip FPGA Stratix V của Altera. Kết quả thực nghiệm cho thấy, kiến trúc này cho độ chính xác cao với 5,889E-06 MSE và 2,456 mỗi phần nghìn - ppt tỉ lệ lỗi tối đa. Tần số tối đa đạt được là 102,55 MHz, hiệu suất thực thi được 8424 lần 2048-điểm FFT trên mỗi giây. Tài nguyên sử dụng cho thiết kế này là 76.282 ALUTs và 15.687 thanh ghi. Kết luận, thiết kế FFT với độ chính xác dấu chấm động đơn và sử dụng bộ nhân số phức dựa trên xoay góc thích nghi có thể khắc phục được các vấn đề về độ chính xác cao với miền dữ liệu lớn và độ trễ được cải thiện tốt hơn. Trong tương lai gần, đề tài có thể thực hiện với số điểm tính lớn hơn thí dụ như thiết kế FFT từ 128 đến 8192 điểm. Ngoài ra, thiết kế sau khi được thực nghiệm trên FPGA có thể tiến hành ASIC để có thêm các kết quả thực nghiệm về hiệu suất, độ chính xác cũng như tài nguyên sử dụng trên chip.

Lời cảm ơn: *Bài báo là kết quả hợp tác nghiên cứu giữa Phòng Thí Nghiệm DESLAB, Khoa Điện tử - Viễn thông, Trường Đại học Khoa học Tự nhiên – ĐHQG-HCM và Phòng Thí Nghiệm VLSI, Trường Đại Học Điện tử - Truyền thông (University of Electro-Communications - UEC), Tokyo, Nhật Bản. Nghiên cứu được tài trợ bởi Đại học Quốc gia Thành phố Hồ Chí Minh (ĐHQG - HCM) trong khuôn khổ Đề tài mã số B2017-18-05.*

An FPGA-based single-precision floating-point 2048-points FFT implementation based on adaptive angle recoding CORDIC

- **Truong Thi Nhu Quynh**
- **Vo Thi Phuong Thao**
- **Hoang Trong Thuc**
- **Le Duc Hung**

University of Science, VNU-HCM

ABSTRACT

In this paper, an FPGA-based single-precision floating-point 2048-point FFT implementation is proposed, based on an adaptive angle recoding CORDIC algorithm. The design is built and verified on Altera Stratix V FPGA chip. The implementation had 102.55 MHz maximum frequency, throughput result of 8424.382 FFTs/s, and resources utilization of 76,282 ALUTs and 15,687 registers. The accuracy results were 5.889E-06 (Mean-Square-Error (MSE).

Keywords: *FFT, adaptive angle recoding CORDIC, AARC, floating-point*

TÀI LIỆU THAM KHẢO

[1]. J.W. Cooley, J.W. Tukey, An Algorithm for the machine calculation of complex Fourier series, *Mathematics of Computation*, 19, 297–301 (1965).

[2]. J.G. Andrews, A. Ghosh, R. Muhamed, *Fundamentals of WiMAX: Understanding Broadband Wireless Networking*, Prentice Hall Communications Engineering and Emerging Technologies Series (2007).

[3]. E. Dahlman, *3G Evolution: HSPA and LTE for Mobile Broadband*, Academic Press (2008).

[4]. A.F. Molisch, X. Zhang, FFT-based hybrid antenna selection schemes for spatially correlated MIMO channels, *IEEE Communications Letters*, 8, 1, 36–38 (2004).

[5]. Y. Tang, B. Vucetic, The FFT-based multiuser detection for DSCDMA ultra - wideband communication systems, in *Ultra Wideband Systems Joint with Conf. on Ultrawideband Systems and Technologies*, Kyoto, Japan, 111–115 (2004).

[6]. X. Li, Z. Lai, J. Cui, A low power and small area FFT processor for OFDM demodulator, *IEEE Trans. on Consume Electronics*, 53, 2, 274–277 (2007).

[7]. J. Li, M.V. Sarunic, L. Shannon, Scalable, High Performance Fourier Domain Optical Coherence Tomography: Why FPGAs and not GPGPUs, in *IEEE 19th Annual Int. Symp. on Field-Programmable Custom Computing Machines (FCCM)*, Salt Lake City, Utah, USA, 49–56 (2011).

[8]. J. Zhou, Y. Dong, Y. Dou, Y. Lei, Dynamic Configurable FloatingPoint FFT Pipelines and Hybrid-Mode CORDIC on FPGA, in *2008 Int. Conf. on Embedded Software and Systems (ICESS'08)*, Sichuan, China, 616–620 (2008).

[9]. E. Oruklu, X. Xiao, J. Saniie, Reduced memory and low power architecture for CORDIC-based FFT processors, *Journal of*

Signal Processing Systems, 2, 66, 129–134 (2011).

[10]. R. Radhouane, P. Liu, C. Modlin, Minimizing the Memory Requirement for Continuous Flow FFT Implementation: Continuous Flow Mixed Mode FFT (CFMM-FFT), in *IEEE Int. Symp. on Circuits and Systems (ISCAS 2000)*, Genève, Switzerland, I–116 I–119 (2000).

[11]. V. Arunachalam, A.N.J. Raj, Efficient VLSI Implementation of FFT for orthogonal frequency division multiplexing applications, *IET Circuits, Devices & Systems*, 8, 6, 526–531 (2014).

[12]. V. Montano, M. Jiménez, Design and Implementation of a Scalable Floating-point FFT IP Core for Xilinx FPGAs, in *53rd IEEE Int. Midwest Symp. on Circuits and Systems*, Seattle, USA, 533–536 (2010).

[13]. S. Mou, X. Yang, Research on the RAW Dependency in Floatingpoint FFT Processors, in *8th ACIS Int. Conf. on Software Engineering, Artificial Intelligence, Networking, and Parallel Distributed Computing*, Qingdao, China, 88–92 (2007).

[14]. J.H. Min, S.W. Kim, E.E. Swartzlander Jr., A FloatingPoint Fused FFT Butterfly Arithmetic Unit with Merged MultipleConstant Multipliers, in *2011 Conf. Record of the 45th Asilomar Conf. on Signals, Systems and Computers (ASILOMAR)*, Pacific Grove, USA, 520–524 (2011).

[15]. H.Y. Lee, I.C. Park, Balanced Binary-Tree Decomposition for AreaEfficient Pipelined FFT Processing, *IEEE Trans. on Circuits and Systems I: Regular Papers*, 54, 4, 889–900 (2007).

[16]. E.H. Wold, A.M. Despain, Pipeline and parallel-pipelined FFT processors for VLSI Implementations, *IEEE Trans. on Computers*, C-33, 5, 414–426, (1984).

[17]. X. Xiao, E. Oruklu, J. Saniie, Reduced Memory Architecture for CORDIC-based FFT, in *Proc. of 2010 IEEE Int. Sym. On Circuits and Systems*, Paris, France, 2690–2693 (2010).

[18]. T.Y. Sung, H.C. Hsin, Y.P. Cheng, Low-power and High-speed CORDIC-based Split-radix FFT Processor for OFDM Systems, Digital Signal Processing, 2, 20, 511–527 (2010).