XÂY DỰNG BỘ PHÂN TÍCH PHỔ SỬ DỤNG FFT 256 TRIỂN KHAI TRÊN FPGA

BUILT SPECTRUM ANALYZER USING FFT 256 ON FPGA

Nguyễn Đức Nam[1,*], Trần Văn Tiến[1], Phạm Thị Thanh Xuân[1], Đinh Thị Kim Phượng[2]

TÓM TẮT

Bài báo trình bày phương pháp xây dựng bộ phân tích phổ sử dụng FFT 256. Thực hiện triển khai thuật toán FFT 256 điểm theo kiến trúc SDF sử dụng cơ số 2^2 bo mạch FPGA để phục vụ cho các mục tiêu về sau. Nghiên cứu sử dụng phần mềm MatLab, Simulink để xây dựng mô hình và kiểm tra kết quả.

Từ khóa: *Phân tích phổ, FFT, kiến trúc SDF.*

ABSTRACT

The paper presents method of building the spectrum analyzer using FFT 256. Implementing the 256-point FFT algorithm of SDF architecture using base 2^2 the FPGA board to serve the following goals. This paper using Matlab, Simulink software to build models and test results.

Keywords: *Spectrum analyzer, FFT, FFT algorithm.*

[1]Lớp TTMMT1, Khoa Điện tử, Trường Đại học Công nghiệp Hà Nội
[2]Khoa Điện tử, Trường Đại học Công nghiệp Hà Nội
[*]Email: nguyenducnam2509@gmail.com

1. GIỚI THIỆU

Trong ngành điện tử viễn thông, việc phân tích phổ như phổ năng lượng, phổ biên độ, phổ pha của tín hiệu nói chung đóng một vai trò hết sức quan trọng. Nó cho ta biết vai trò đóng góp của các thành phần tần số trong tín hiệu như thế nào, năng lượng của chúng ra làm sao,… từ đó chúng ta có phương hướng xử lý tín hiệu đó một cách thích hợp. Vấn đề đặt ra là làm thế nào để thực hiện biến đổi tín hiệu số từ miền thời gian sang miền tần số để quan sát phổ của nó. Câu trả lời đơn giản nhất chính là sử dụng biến đổi Fourier: với tín hiệu liên tục theo thời gian ta có biến đổi Fourier liên tục (Continuous Time Fourier Transform), với tín hiệu rời rạc theo thời gian ta có biến đổi Fourier rời rạc theo thời gian (Discrete Time Fourier Transform - DTFT). DTFT là 1 hàm liên tục của tần số (miền tần số liên tục), tuy nhiên các mẫu rời rạc của nó có thể được tính thông qua DFT (Discrete Fourier Transform), điều này mang lại lợi ích rất lớn do ta có thể áp dụng các thuật toán nhanh để tính DFT.

Biến đổi Fourier rời rạc (DFT) được sử dụng trong rất nhiều các lĩnh vực, nó được sử dụng trong việc xử lý tiếng nói, xử lý ảnh,… Sẽ không phải là phóng đại nếu như nói là bất cứ việc gì liên quan đến xử lý tín hiệu số đều phải dùng đến biến đổi Fourier. Tuy nhiên việc sử dụng biến đổi Fourier rời rạc có một vấn đề, đó là việc tính toán tương đối phức tạp khi chiều dài dữ liệu cần tính toán tăng lên. Chính vì vậy mà thuật toán biến đổi Fourier nhanh - FFT (Fast Fourier Transform) hay STFT (short time Fourier transform) đã ra đời.

Ý tưởng của thuật toán FFT đó là kỹ thuật chia - trị. Thay vì việc tính DFT cho cả 1 tín hiệu có độ dài lớn chúng ta sẽ thực hiện tính DFT cho từng đoạn tín hiệu nhỏ hơn trong tín hiệu đó rồi từ kết quả thu được chúng ta tính ngược lại DFT của tín hiệu cần tính ban đầu.

Một trong những cách tính FFT rất phổ biến là khi độ dài dãy là lũy thừa của 2 do Cooley và Tukey đưa ra vào năm 1965 (thường gọi là FFT cơ số 2). Yêu cầu của phương pháp này đó là dãy tín hiệu cần tính phải có độ dài N phải là lũy thừa của 2. Cách tính của phương pháp này là thực hiện chia dãy tín hiệu cần tính thành 2 nửa có độ dài N/2 rồi tính DFT của 2 dãy nhỏ này. Mỗi dãy nhỏ lại có thể được tính bằng cách tiếp tục chia nhỏ chúng ra thành 2 dãy con nhỏ hơn…Công việc trên được lặp lại đến khi nào việc tính DFT của dãy con trở nên đơn giản rồi, không cần chia nữa (thường là khi độ dài dãy con chỉ còn là 2), khi đó từ kết quả DFT của dãy con chúng ta tính ngược lại DFT của dãy to.

Từ FFT cơ số 2, giờ đây chúng ta còn có các cơ số khác như cơ số 4, cơ số 8, cơ số 2^2, cơ số 2^3 cùng với nhiều kiểu cấu trúc tính FFT khác nhau như song song, SDF(single delay feedback), MDC (multipath delay commutator), tại chỗ (in - place), thác lũ,…ngày càng khẳng định được vai trò quan trọng của FFT.

2. XÂY DỰNG BỘ PHÂN TÍCH PHỔ SỬ DỤNG FFT 256 TRÊN FPGA

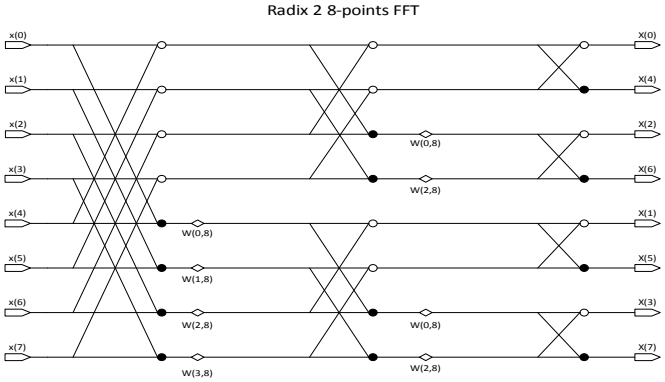
2.1. FFT cơ số 2

Kiến trúc thuật toán FFT cơ số 2 cho 8 điểm được thể hiện trên hình 1.

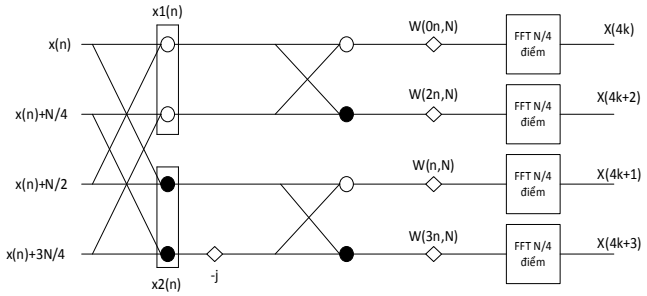
2.2. FFT cơ số 2^2

Trên hình 2 là kiến trúc của thuật toán FFT với cơ số 2^2. Thực chất của thuật toán này là ghép 2 tầng cơ số 2 thành 1 tầng cơ số 2^2. Tuy nhiên thuật toán cơ số 2^2 có 1 ưu điểm so với thuật toán cơ số 2. Cứ sau mỗi một tầng cơ số 2 chúng ta phải nhân thêm hệ số twiddle factor, còn ở trong cơ số 2^2 chúng ta chỉ nhân twiddle factor ở sau tầng thứ 2 còn ở

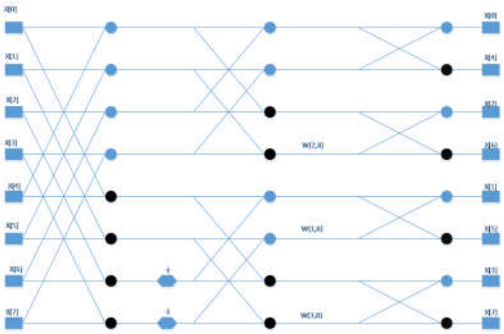
tầng trước chỉ nhân với hệ số -j - hệ số tầm thường, do đó khi thực hiện bằng máy tính sẽ bớt được số lượng phép tính toán phức và cải thiện được độ phức tạp.



Hình 1. Kiến trúc của thuật toán FFT cơ số 2 cho 8 điểm



Hình 2. Kiến trúc của thuật toán FFT cơ số 2^2



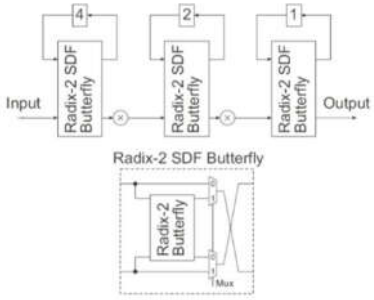
Hình 3. Sơ đồ tính FFT cơ số 2^2 cho dãy đầu vào độ dài bằng 8

2.3. Kiến trúc SDF

Trong ví dụ ở trên hình 2, kiến trúc mà chúng ta sử dụng là kiến trúc song song, tức là chúng ta trải hết tất cả các đầu vào (FFT bao nhiêu điểm thì dùng ngần đấy đầu vào) và thực hiện tính toán. Ưu điểm của kiến trúc này đó là tốc độ xử lý cao, không có delay, chỉ cần có đủ N đầu vào là lập tức tính toán ra được kết quả đầu ra. Tuy nhiên trong thực tế, các bộ FFT thường được sử dụng trong các thiết bị chạy thời gian thực real - time, đầu vào của chúng ta không phải là 1 mảng có sẵn N phần tử mà chỉ là từng phần tử 1, do đó kiến trúc song song như thế này không phù hợp.

Để khắc phục nhược điểm này, người ta sẽ sử dụng 1 kiến trúc khác. Một trong những kiến trúc đơn giản, hay được sử dụng là kiến trúc SDF (single delay feedback).

Dưới đây là 1 ví dụ về FFT cơ số 2 dùng kiến trúc SDF với độ dài N = 4.



Hình 4. FFT 8 điểm dùng cơ số 2 SDF

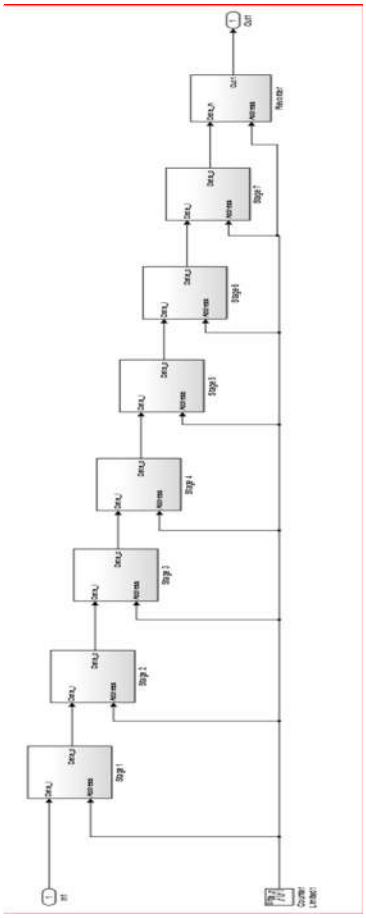
2.4. Triển khai FFT 256 điểm cơ số 2^2 theo kiến trúc SDF

Hình 4 là sơ đồ thuật toán tính FFT 256 điểm cơ số 2^2 theo kiến trúc SDF thực hiện trên phầm mềm Simulink của MatLab. Bao gồm 7 stage, 1 khối Revorder và 1 bộ đếm 7 bit.

+ **7 stage**: trong 7 stage thì stage 1 là cơ số 2 còn lại là cơ số 2^2, cứ 2 tầng liên tiếp tạo thành một tầng cơ số 2^2. Các Stage được kết nối theo kiến trúc SDF và có chức năng thực hiện việc tính toán FFT.

+ **Khối đếm 8 bit** có chức năng đếm từ 0 đến 256, khối này có nhiệm vụ tạo ra giá trị đầu vào Address (từ 0 đến 256 để cho các Stage thực hiện điều khiển việc tính FFT và để khối Revorder thực hiện việc sắp xếp.

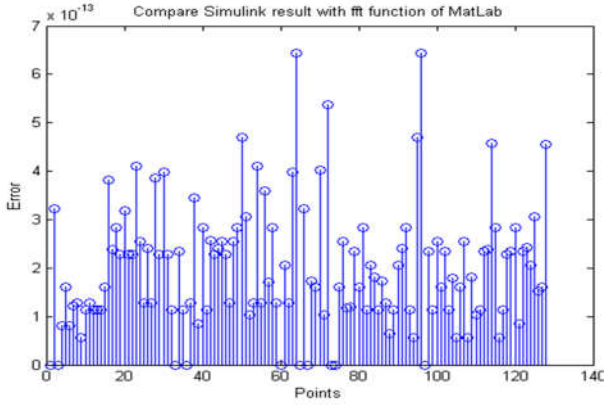
+ **Khối Revorder** có chức năng sắp xếp lại các giá trị đầu ra.



Hình 5. FFT 256 điểm cơ số 2^2 kiến trúc SDF

Sau khi xây dựng kiến trúc, tiến hành mô phỏng. Đầu tiên là mô phỏng rồi so sánh với kết quả thu được khi sử dụng hàm fft của MatLab.

Như ta thấy trên hình 5, giá trị sai số chỉ khoảng 10^{-13}, nghĩa là khối FFT của chúng ta đã xây dựng khá là chính xác.



Hình 6. Sai số giữa Simulink và hàm FFT của Matlab

2.5. Chuyển đổi sang kiểu dữ liệu dấu phẩy tính (Fixed point)

Kiểu dữ liệu dấu phảy tĩnh là một cách biểu diễn số thực nhưng sử dụng số nguyên để biểu diễn trong đó số bit để biểu diễn phần nguyên và phần thập phân của số thực được lựa chọn cố định. Điều này khác với dấu phẩy động khi mà trong số thực dấu phẩy động, dấu phẩy thập phân có thể thay đổi vị trí bằng cách thay đổi giá trị trường mũ.

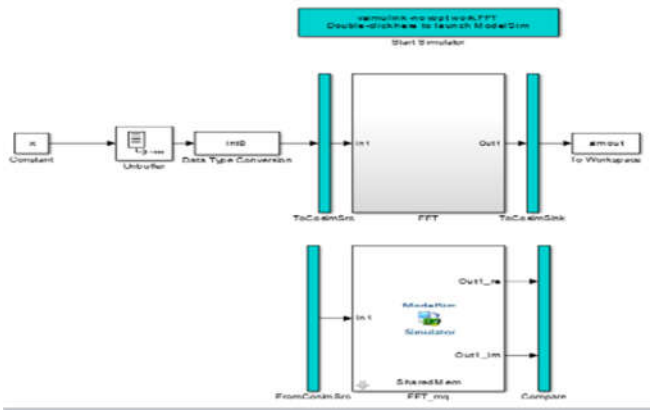
	Phần nguyên														Phần thập phân														
S	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
31	30													16	15														0

Hình 7. Biểu diễn số thực dấu phảy tĩnh 32 bit với 16 bit thập phân ở dạng nhị phân

Trên hình 7 là cách biểu diễn 1 số thực có dấu dưới dạng dấu phảy tĩnh 32 bit với 16 bit phần thập phân. Cách tính giá trị của số được biểu diễn được tính bình thường giống như khi chuyển giá trị của 1 số nhị phân sang hệ 10.

3. KẾT QUẢ, KẾT LUẬN VÀ ĐÁNH GIÁ

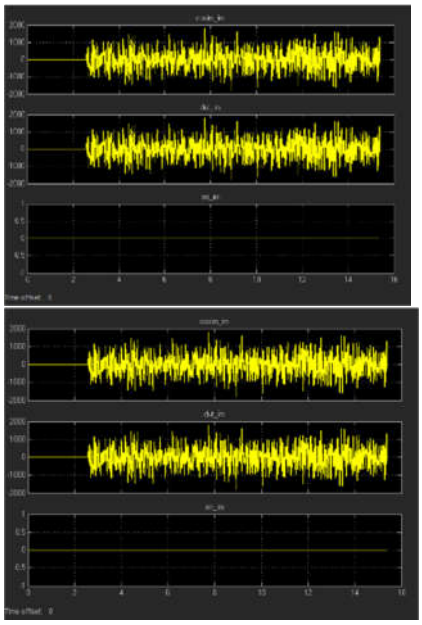
3.1. Kết quả kiểm thử



Hình 8. Co - Simulation model với ModelSim

Sau khi thực hiện biến đổi mô hình FFT sang kiểu dữ liệu fixed point, chúng ta sử dụng công cụ HDL Coder của MatLab để thực hiện gen code Verilog. Hình 8 là kết quả sau khi gen để mô phỏng với ModelSim.

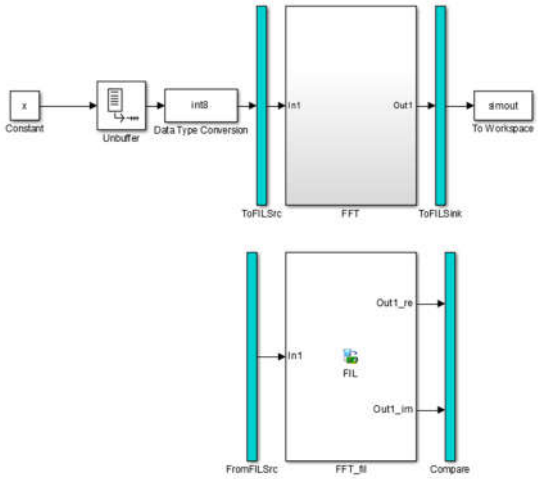
Hình 9 là kết quả Co - Simulation với ModelSim. Việc mô phỏng được thực hiện 10 lần, với đầu vào là các số nguyên 8 bit từ -257 đến 256 và được thực hiện theo mô phỏng Monte - Carlo.



Hình 9.Kết quả thực hiện Co - Simulation với ModelSim

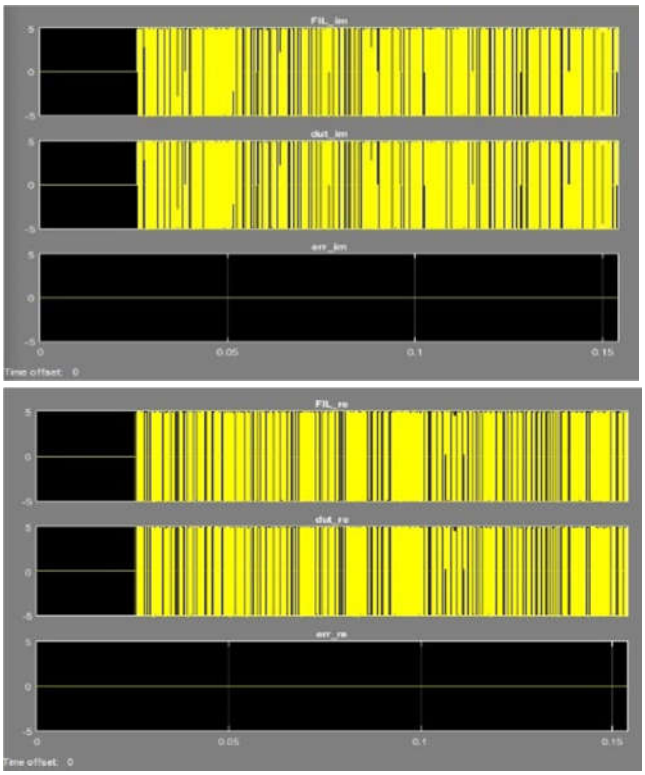
Từ hình 9ta thấy khối phần cứng tổng hợp bằng ngôn ngữ Verilog cho kết quả giống với mô hình Simulink, vậy là khối đã hoạt động đúng.

Hình 10 là kết quả sau khi gen FPGA - in - loop với HDL Workflow Advisor của MatLab.



Hình 10. Khối Co - Simulation FPGA - in - loop

Việc mô phỏng Co - Simulation FPGA - in - loop cũng được thực hiện bằng mô phỏng Monte - Carlo, thực hiện mô phỏng 300 lần với đầu vào khối FFT vẫn là các số nguyên có dấu nằm trong khoảng [-128 127]. Kết quả mô phỏng được thể hiện trên hình 11.



Hình 11. Kết quả mô phỏng Co - Simulation FPGA - in - loop

Từ hình 10 ta thấy khối FFT được triển khai trên kit Atlys đã chạy đúng kết quả, như vậy đề tài đã được hoàn thành.

3.2. Đánh giá, ưu nhược điểm

Ưu điểm

- Phát triển các kĩ năng viết báo cáo kĩ thuật, cải thiện được kĩ năng đọc tiếng anh chuyên ngành.

- Trong quá trình làm nhiệm vụ, em đã được học các kĩ năng lập trình MatLab, Simulink, được học thêm về ngôn ngữ mô tả phần cứng Verilog, tìm hiểu về FPGA, Xilinx, ModelSim, FPGA - in - loop,…

Nhược điểm

- Phần cứng sử dụng khá đắt

- Ngôn ngữ lập trình không phổ biến

4. KẾT LUẬN

Nhóm tác giả đã xây dựng thành công bộ phân tích phổ sử dụng FFT 256 triển khai trên FPGA. Bài báo đã đạt được một số kết quả như: nâng cao tốc độ, độ chính xác của xử lý số tín hiệu; mở ra một lĩnh vực rất rộng của phân tích phổ: viễn thông, thiên văn, địa lý, chẩn đoán y khoa,…; đặt nền móng cho việc tính toán các biến đổi khác như biến đổi Walsh, biến đổi Hamadard, biến đổi Haar,…

TÀI LIỆU THAM KHẢO

[1]. Nguyễn Quốc Trung. *Xử lý tín hiệu và lọc số Tập 2*. Nhà xuất bản Khoa học và kỹ thuật.

[2]. Guoan Bi, Gang Li, 2011. *Pipelined Structure Based on Radix - 22 FFT Algorithm*. in Industrial Electronics and Applications (ICIEA), 2011 6th IEEE Conference.

[3]. Trio Adiono, Rella Mareta, 2012. *Low Latency Parellel – Pipelined Configurable FFT/IFFT 128/256/512/1024/2048 for LTE*. in Intelligent and Advanced Systems (ICIAS), 2012 4th International Conference.