THIẾT KẾ BỘ XỬ LÝ TÍN HIỆU DỰA TRÊN THUẬT TOÁN LMS

SIGNAL PROCESSOR DESIGN BASED ON LMS ALGORITHM

Đỗ Thị Huyền[1,*], Vũ Trọng Nghĩa[1], Đặng Hồng Đức[1], Nguyễn Văn Khuê[1], Bồ Quốc Bảo[2]

TÓM TẮT

Bài báo trình bày về thuật toán thích nghi LMS và cách thiết kế bộ xử lý tín hiệu dựa trên thuật toán LMS. Bài toán xử lý nhiễu và tạp âm là vấn đề quan trọng trong xử lý tín hiệu khi truyền đi. Để nâng cao chất lượng tín hiệu thu được, ở thiết bị thu cần phải tích hợp các khối xử lý để giảm thiểu ảnh hưởng của nhiễu và tạp âm, đồng thời bù trừ những thay đổi của kênh truyền.

Từ khóa: *Thuật toán thích nghi LMS, nhiễu, tạp âm.*

ABSTRACT

The paper presents LMS adaptive algorithm and how to design signal processor based on LMS algorithm. The problem of noise and noise processing is an important issue in signal processing when transmitting. To improve the quality of the received signal, the receiver needs to integrate processing units to minimize the effects of noise and noise, while compensating for changes in the channel.

Keywords: *LMS adaptive algorithm, noise, noise processing.*

[1]Lớp TTMMT1, Khoa Điện tử, Trường Đại học Công nghiệp Hà Nội
[1]Khoa Điện tử, Trường Đại học Công nghiệp Hà Nội
[*]Email: thuhuyen.010198@gmail.com

1. GIỚI THIỆU

Ngày nay cùng với sự phát triển vượt bậc của khoa học kỹ thuật, các hệ thống tương tự được thay thế bằng các hệ thống số. Các công nghệ mới được ứng dụng rộng rãi cho xử lý tín hiệu. Bài toán loại bỏ can nhiễu và tạp âm luôn luôn là vấn đề lớn trong các hệ thống xử lý tín hiệu. Để loại bỏ can nhiễu và tạp âm thường sử dụng các bộ lọc. Các bộ lọc kinh điển được thiết kế với mục đích chọn lọc tần số (bộ lọc thông thấp, bộ lọc thông cao, bộ lọc thông dải,...) hay cực tiểu hóa bình phương trung bình của tín hiệu sai lệch. Tuy nhiên những phương pháp này yêu cầu cần phải biết trước các đặc trưng thống kê cơ bản của nhiễu như kỳ vọng, phương sai, hàm tương quan,... giả định nhiễu và tập âm là những quá trình ngẫu nhiên không dừng do đó các tham số của nó thay đổi theo thời gian và do vậy việc thiết kế các bộ lọc theo phương pháp kinh điển rất khó đạt được hiệu quả cao. Để phù hợp với điều kiện thực tế người ta đã đề xuất phương pháp xử lý tín hiệu thích nghi. Mục đích của xử lý tín hiệu thích nghi là đạt được tín hiệu đầu ra tối ưu. Việc nghiên cứu và xử lý tín hiệu trong môi trường không ngừng dựa trên các thuật toán xử lý thích nghi có một ý nghĩa thực tiễn rất lớn khi thiết kế các hệ thống thông tin có độ chính xác cao.

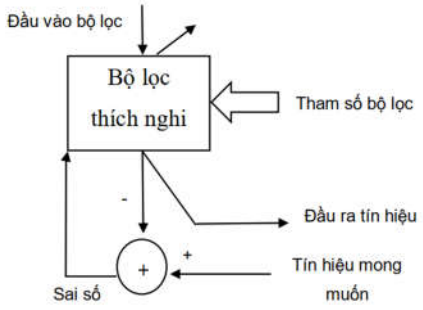
Trước kia do công nghệ chế tạo IC còn hạn chế nên việc thực hiện các thuật toán xử lý tín hiệu thích nghi là rất khó khăn. Ngày nay công nghệ chế tạo IC phát triển vượt bậc nên việc sử dụng các dụng cụ điện tử, thiết bị bán dẫn và các bộ vi xử lý có độ tích hợp cao như DSP, FPGA,... để thực hiện thuật toán xử lý tín hiệu thích nghi dễ dàng hơn rất nhiều.

2. TỔNG QUAN VỀ LỌC THÍCH NGHI

2.1. Xử lý tín hiệu thích nghi

Xử lý tín hiệu thực chất là một quá trình lấy ra tín hiệu mong muốn từ một tập tín hiệu có lẫn nhiễu tại đầu vào máy thu. Tín hiệu khi được truyền đi trong môi trường bị biến dạng bởi các tác động của can nhiễu và tạp âm. Do vậy tại thiết kế bộ thu ta phải thiết kế như thế nào để giảm được tác động của nhiễu càng nhiều càng tốt. Với mục đích nâng cao độ tin cậy cho thiết bị thu thì các hệ thống thông tin cần phải tích hợp các khối xử lý để giảm ảnh hưởng của nhiễu và tạp âm. Những khối này luôn tồn tại trong các hệ thống thông tin tương tự cũng như các hệ thống thông tin số, chúng có thể quy về các bộ lọc và các bộ san bằng.

Một trong những ứng dụng quan trọng của các bộ lọc là loại bỏ nhiễu và tạp âm. Các bộ lọc kinh điển được thiết kế với mục đích chọn lọc tần số, nó sẽ rất có hiệu quả nếu phổ của tín hiệu có ích và nhiễu ổn định, phân bố ở những vùng riêng biệt trên miền tần số.



Hình 1. Sơ đồ khối của hệ thống xử lý tín hiệu thích nghi

Để phù hợp với điều kiện thực tế người ta đã đề xuất phương pháp xử lý tín hiệu thích nghi. Mục đích của xử lý tín hiệu thích nghi là tách ra thành phần có ích tốt nhất theo nghĩa này hay nghĩa khác. Mọi thuật toán xử lý tín hiệu thích nghi đều xuất phát từ một tập điều kiện ban

đầu, để đảm bảo được tín hiệu thu tốt nhất thì các bộ lọc thích nghi vẫn phải thực hiện quá trình điều chỉnh trọng số bộ lọc dù không biết trước được các tính chất thống kê của tín hiệu vào. Nhưng thay vì phải đưa ra tất cả mọi thông tin về một quá trình nào đó thì ta chỉ phải đưa ra một chuỗi mẫu tín hiệu trong các thời điểm kế tiếp. Có rất nhiều biện pháp để có thể tìm ra được tín hiệu mong muốn nhưng phương pháp hiệu chỉnh theo sai số bình phương trung bình là phổ biến.

2.2. Lọc tối ưu cầu biên Wiener

Có hai yêu cầu đặt ra với bộ lọc:

• Bộ lọc phải tuyến tính, điều này nhằm để đơn giản trong quá trình tính toán.

• Bộ lọc hoạt động rời rạc theo thời gian, yêu cầu này nhằm để cho bộ lọc có thể xây dựng bằng phần cứng hay phần mềm số.

Tiêu chuẩn tối ưu có thể được lựa chọn từ một trong các phương án sau:

• Giá trị bình phương trung bình của sai số được đánh giá.

• Kỳ vọng của giá trị tuyệt đối của sai số được đánh giá.

• Kỳ vọng bậc ba hoặc cao hơn giá trị tuyệt đối của sai số được đánh giá.

Ở đây xét một bộ lọc tuyến tính tối ưu sử dụng tiêu chuẩn bình phương trung bình, lỗi e(n) của bộ lọc tuyến tính được xác định như sau:

e(n) = d(n) – y(n)

Hàm tổn thất (hàm định giá) J: là tổ hợp của nhiều e(n), được định giá cho cả 1 chuỗi tín hiệu, với E là toán tử kì vọng.

$J = |\overline{e(n)}|^2 = E[|e(n)|^2]$

Lọc Wiener có hàm định giá cực tiểu theo tiêu chuẩn bình phương trung bình, xét điều kiện để J cực tiểu.

Ta có quan hệ vào - ra của lọc rời rạc tuyến tính thỏa mãn phương trình tích chập như sau:

$Y(n) = \sum_{k=0}^{N-1} W_k^* u(n-k)$

Thành phần lỗi e(n) được xác định bằng hiệu của tín hiệu mong muốn d(n) và y(n):

$$e(n) = d(n)-y(n) = d(n)- \sum_{k=0}^{N-1} W_k^* u(n-k)$$

Hàm định giá J có thể được viết như sau:

$$J= \sigma_d^2 - \sum_{k=0}^{M-1} w_k^* p(-k) - \sum_{k=0}^{M-1} w_k p^*(-k) + \sum_{k=0}^{M-1} \sum_{i=0}^{M-1} w_k^* w_i r(i-k)$$

Hệ phương trình Wiener - Hoft có dạng:

$\sum_{i=0}^{M-1} w_{0i} r(i-k) = p(-k)$, $k=\overline{1, M-1}$

Các hệ số lọc: $W_0=R^{-1}p$

2.3. Thuật toán lọc thích nghi gradient giảm dốc nhanh nhất

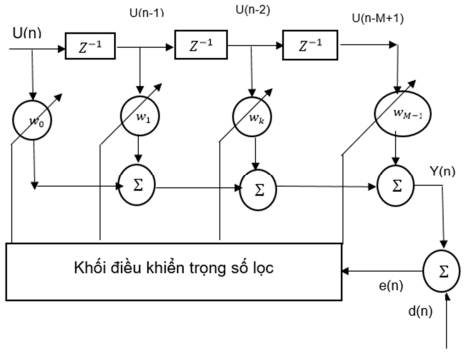
Các bước thực hiện của thuật toán này như sau:

• Vector trọng số lọc w được gán giá trị ban đầu w(0), đây là một giá trị dự đoán ban đầu. Trừ phi ta đã biết trước được gia trị ban đầu, bằng không thì w(0) thường được đặt bằng vector 0.

• Tính vector gradient $\nabla J_n(n)$ của hàm J(n).

• Giá trị tiếp theo của vector trọng số lọc được xác định theo bởi chiều của vector gradient và vector trọng số lọc trước đó.

• Quay lại bước hai và lập lại quá trình trên, bằng trực quan ta có thể thấy rằng sự hiệu chỉnh trọng số lọc liên tiếp theo chiều âm của vector gradient sẽ kiến cho J(n) -> J_{min}, tại đó hệ số lọc là tối ưu và tương ứng với lọc [1].



Hình 2. Lọc thích nghi gradient giảm dốc nhanh nhất

3. TỔNG QUAN VỀ PHẦN MỀM SYSTEMVUE

3.1. Khái niệm

SystemVue một môi trường thiết kế điện tử tự động (Electronic Design Automation - EDA) phục vụ việc thiết kế tầng hệ thống điện tử (Electronic System Layer - ESL). Phần mềm cho phép các nhà thiết kế hệ thống và thuật toán có thể nâng cấp tầng vật lý (PHY Layer) của các hệ thống không dây và các hệ thống thông tin và cung cấp nhưng công cụ đắc lực cho RF, DSP và hệ thống nhúng FPGA/ASIC. Như một nền tảng phục vụ cách nhìn nhận hiện tại về thiết kế ESL và xử lý số tín hiệu, SystemVue thay thế các môi trường số thông dụng, tương tự, và toán học hiện nay [2].

Hình 3. Logo phần mềm SystemVue

3.2. Các tính năng chính của phần mềm SystemVue

Mô trường làm việc chính:

- Dễ sử dụng, đa nhiệm và là công cụ Windows cao cấp.

- Mục thiết kế đa hình hỗ trợ quy trình thiết kế khối (khối GUI, hỗ trợ ngôn ngữ C++ hoặc toán học, VHDL).

- Chú thích, đồ thị và lưu trữ các giá trị kiểm thử đầu vào/ra hệ thống.

- Tùy chỉnh thời gian thực cùng với công cụ thông báo trực tiếp hỗ trợ tương tác trực tiếp và liên tục trong quá trình thiết kế.

- Dễ dàng đóng gói các IP hiện tại dưới nhiều định dạng và thực hiện ghép nối các quy trình không liên tục.

- Môi trường làm việc nhóm qua mạng đáng giá và trực quan hơn so với môi trường công nghiệp.

Giao diện thiết kế tùy chỉnh ngôn ngữ C++:

- Xây dựng hệ thống dấu phẩy động - tĩnh bằng ngôn ngữ C++.

- Sửa lỗi khối với giao diện Microsoft Visual Studio quen thuộc.

Ngôn ngữ toán phổ thông và công cụ sửa lỗi:

- Tự động hỗ trợ với hàng trăm phép toán và lỗi về giao tiếp hệ thống.

- Hỗ trợ cả dưới dạng văn bản cũng như giao diện GUI giúp cho việc tạo, mô phỏng và kiểm thử dễ dàng hơn.

- Giao diện các dòng lệnh, công cụ sửa lỗi trực quan và liên kết TCP/IP quen thuộc, thay thế cho một loạt các công cụ bản quyền khác.

Cơ chế mô phỏng dòng dữ liệu hiệu năng cao:

- Hỗ trợ sóng mang phức RF, dòng dữ liệu đồng bộ thời gian và dòng dữ liệu tùy biến cho các hệ thống tầng vật lý hiệu năng cao hiện tại và các hiệu ứng RF, bao gồm cả đo đạc" thông lượng" và hệ thống radio có nhận thức (cognitive radio).

- Chức năng Advanced Scheduler cho phép hệ thống liên kết phức tạp đa tốc độ.

- Hoạt động đa nhiệm vụ giúp tăng tốc độ mô phỏng trên các CPU có nhiều nhân.

- Hỗ trợ việc thiết kế với các đoạn code HDL và MATLAB có sẵn bên ngoài.

Hiệu ứng mô hình lớp vật lý và bộ khối đa năng:

- Các khối RF, DSP, logic và kênh đã bao gồm trong môi trường thiết kế cơ bản.

- Kiểm soát các hiệu ứng tương tự, tạp âm pha, hệ số S, các hiệu ứng lệch IF DC, phụ thuộc tần số và nhiều hơn thế.

Liên kết với việc đo đạc và kiểm thử:

- Giao tiếp I/O TCP/IP với các thiết bị nhúng trực tiếp bên trong dòng dữ liệu mô phỏng hoặc trong các dòng kênh.

- Tái sử dụng cùng một cài đặt, mã, vector kiểm tra và IP mạng không dây khi chuyển từ thuật toán sang việc kiểm tra thực tế.

- SystemVue được cài đặt sẵn trên nhiều thiết bị giúp tạo ra nhiều tùy chọn mới.

Tổng hợp bộ lọc số:

- Các loại bộ lọc tương tự FIR, IIR.

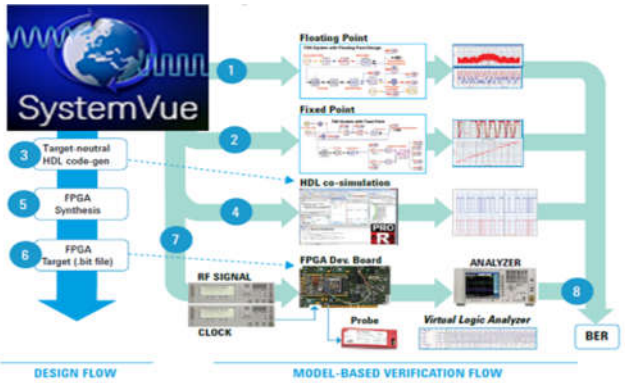
- Xem được trong miền thời gian cũng như tần số, một cách trực quan.

- Thiết lập các bộ lọc một cách đơn giản chỉ với việc click chuột [3].

3.3. Quy trình thiết kế trên FPGA với phần mềm SystemVue

Phần này mô tả về quy trình thiết kế trên FPGA sử dụng phần mềm SystemVue. Tương tự, việc nhúng một hệ thống FPGA tập trung chính vào việc xây dựng theo hướng từ dưới lên trên (bottom-up) ở VHDL/Verilog. Cụ thể hơn, sẽ rất khó để có thể mô phỏng và kiểm thử quy trình của một khối nhỏ trong cả một hệ thống lớn với nhiều ngôn ngữ thiết kế khác nhau, như Matlab Simulink, C/C++.

Hình 4 cho thấy quy trình thiết kế với phần mềm Agilent SystemVue.



Hình 4. Quy trình thực hiện nhúng trên FPGA

Quy trình bao gồm 8 bước:

Bước 1. Thiết kế hệ thống và kiểm thử trên môi trường dấu phảy động.

Bước 2. Thiết kế hệ thống và kiểm thử trên môi trường dấu phảy tĩnh.

Bước 3. Tạo code HDL.

Bước 4. Kiểm thử mô phỏng với ngôn ngữ HDL.

Bước 5. Tạo file lập trình cho FPGA.

Bước 6. Nạp file .bit vào FPGA.

Bước 7. Tạo tín hiệu phát tới FPGA để kiểm thử.

Bước 8. Kiểm thử FPGA.

4. THIẾT KẾ BỘ XỬ LÝ TÍN HIỆU DỰA TRÊN THUẬT TOÁN LMS

4.1. Thuật toán LMS

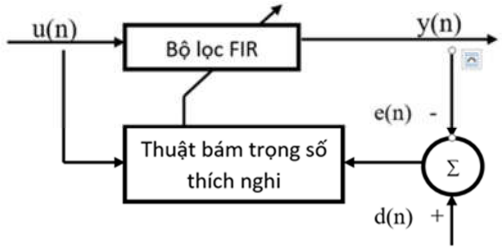
Thuật toán LMS được xây dựng bởi Wiener và Hopf, là một trong những thành viên quan trọng trong hệ thuật toán gradient ngẫu nhiên. Đặc điểm nổi bật nhất của LMS là đơn giản, không yêu cầu tìm ra ma trận tương quan và cũng không cần tính ma trận nghịch đảo, do vậy thuật toán này đơn giản và được sử dụng làm tiêu chuẩn cho các thuật toán xử lý thích nghi khác.

LMS là thuật toán lọc thích nghi tuyến tính,bao gồm hai quá trình cơ bản sau:

• Quá trình lọc: quá trình này bao gồm việc tính toán đầu ra của bộ lọc theo các tín hiệu vào băng lọc và đánh giá sự sai lệch giữa đầu ra và tín hiệu chuẩn (tín hiệu mong muốn).

• Quá trình thích nghi: Đây là quá trình điều khiển tự động trọng số lọc tương ứng với sai số được đánh giá.

Như vậy, thuật toán LMS là sự kết hợp đồng thời của hai quá trình này và được minh họa ở hình 5.



Hình 5. Sơ đồ biểu diễn thuật toán LMS

Thuật toán LMS sử dụng tiêu chuẩn bình phương trung bình cực tiểu để đánh giá sai số học.

Kết quả của thuật toán LMS gồm:

• Tín hiệu đầu ra bộ lọc: y(n)= $\widehat{W}^H$(n).u(n).

• Sai số đánh giá: e(n) = d(n)-y(n)

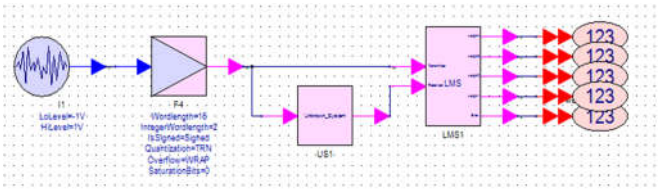
• Phương trình cập nhật trọng số [1]:

$\widehat{W}$(n+1)=$\widehat{W}$(n) +$\mu u(n)e^*$(n)

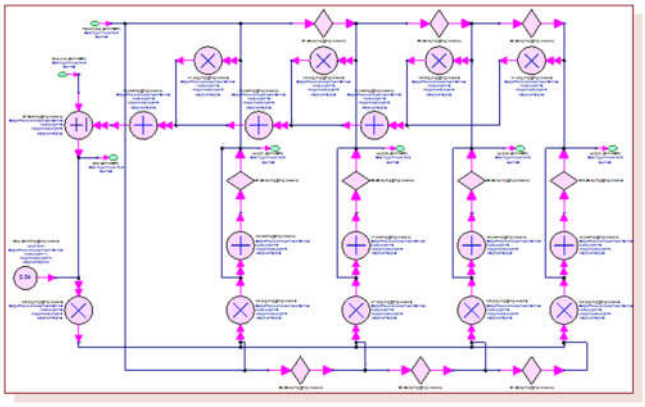
4.2. Thiết kế bộ xử lý tín hiệu

Mục tiêu: Thiết kế phần cứng từ các thiết kế ở mức hệ thống như thiết kế dựa trên mô hình.

Yêu cầu: Thiết kế mạch điện tử từ mức hệ thống trên phần mềm Systemvue. Thiết kế bộ xử lý tín hiệu dựa trên thuật toán LMS dựa trên mô hình với bậc của bộ lọc là 4.



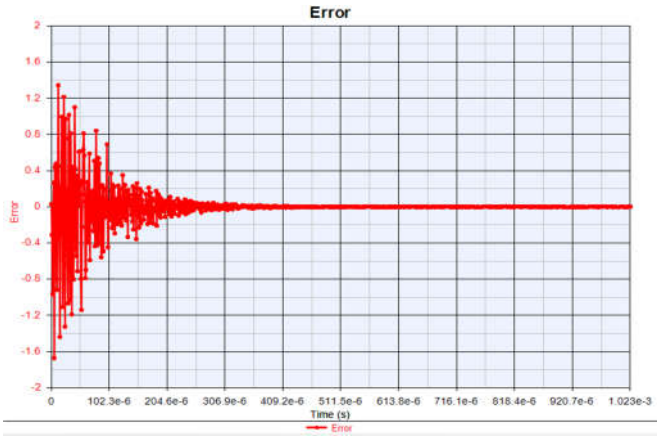
Hình 6. Sơ đồ khối bộ xử lý tín hiệu



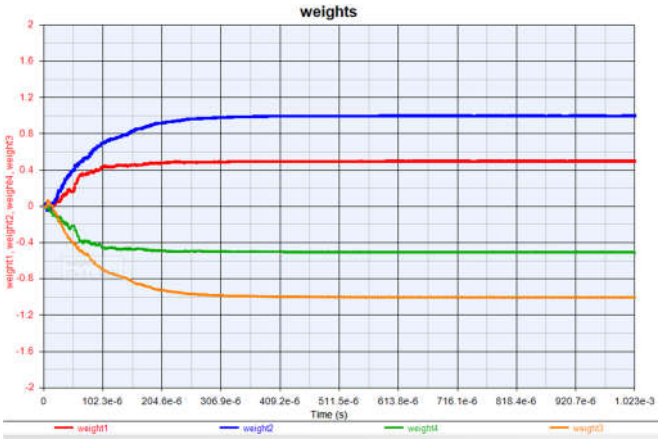
Hình 7. Sơ đồ nguyên lý của LMS

5. KẾT QUẢ

- Biểu đồ trạng thái lỗi như hình 8.
- Biểu đồ trong số lỗi như hình 9.
- Datasheet sau khi thực hiện như hình 10.



Hình 8. Biểu đồ trạng thái lỗi



Hình 9. Biểu đồ trọng số lỗ

Variable	Index	weight1_Time (s)	weight1
Error	1	0	0
Error_Time	2	1e-6	61.04e-6
FixedPointAnalysis	3	2e-6	9.094e-3
LogOutput=Data Flow Analysis : L	4	3e-6	0.021
weight1	5	4e-6	0.023
weight1_Time	6	5e-6	0.032
weight2	7	6e-6	-0.021
weight2_Time	8	7e-6	-0.017
weight3	9	8e-6	-0.039
weight3_Time	10	9e-6	-0.034
weight4	11	10e-6	-0.033
weight4_Time	12	11e-6	-0.023
	13	12e-6	-0.023
	14	13e-6	8.179e-3
	15	14e-6	0.024
	16	15e-6	0.029
	17	16e-6	0.011
	18	17e-6	8.667e-3
	19	18e-6	6.775e-3
	20	19e-6	0.027
	21	20e-6	0.029
	22	21e-6	0.032
	23	22e-6	0.043
	24	23e-6	0.082
	25	24e-6	0.087
	26	25e-6	0.049
	27	26e-6	0.052
	28	27e-6	0.052
	29	28e-6	0.054

Hình 10. Datasheet sau khi thực hiện

6. KẾT LUẬN

Nghiên cứu đã trình bày quy trình và các bước thiết kế mạch dùng phần mềm SystemVue, thiết kế mạch dùng SystemVue, thiết kế được bộ xử lý tín hiệu dựa trên thuật toán LMS.

TÀI LIỆU THAM KHẢO

[1]. K. Technologies, 2014. *FPGA Prototyping Using keysight systemVue*. USA.

[2]. Richard E. Haskell, 2009. *Introduction to Digital Design Using Digilent FPGA Boards.*