THIẾT KẾ BIẾN ĐỔI HADAMARD TỐC ĐỘ CAO SỬ DỤNG TRONG CÁC HỆ XỬ LÝ TÍN HIỆU SỐ

TO DESIGN A HIGH SPEED HADAMARD TRANSFORM BLOCK USING IN THE DIGITAL SIGNAL PROCESSING SYSTEMS

Đỗ Xuân Tiến[1], Hoàng Thị Phương[2]
[1] Học Viện Kỹ thuật Quân Sự
[2] Trường Đại học Sư phạm Kỹ Thuật Nam Định

TÓM TẮT

Trong các hệ xử lý tín hiệu số như hệ thống giấu thông tin vào dữ liệu âm thanh, hình ảnh đang được coi là phương pháp bảo mật thông tin có tính hiệu quả và tính khả thi cao cho vấn đề bảo vệ bản quyền. Trong sơ đồ truyền thống của hệ thống giấu thông tin thì khối biến đổi theo thuật toán Hadamard được sử dụng nhiều vì tính đơn giản và tính trực giao của của thuật toán này. Bài báo trình bày phương pháp thiết kế khối biến đổi Hadamard tốc độ cao nhờ sự kết hợp đồng thời giữa tổ chức phần cứng và phần mềm ở mức lõi của kiến trúc ALU của hệ thống giúp thực hiện được nhiều thao tác song song nên tốc độ thực hiện thuật toán rất cao.

Từ khóa: *Khối biến đổi Hadamard tốc độ cao, công nghệ DHT, xử lý song song.*

ABSTRACT

In the digital signal processing systems such as withholding information system on sound data, images are treated as information security measures are effective and feasible for the copyright protection. In traditional scheme of hiding information systems the Hadamard transform block is used more for its simplicity and orthogonality of this algorithm. This paper presents a design methodology of the high speed Hadamard transform block through a combination held simultaneously between hardware and software at the core level of ALU system architecture helps accomplish multiple tasks in parallel so speed implementation of algorithm is very high.

Keywords: *High speed Hadamard transform block, DHT technology, parallel processing.*

ĐẶT VẤN ĐỀ

Các phương pháp giấu thông tin trong ảnh có thể được chia thành hai nhóm chính. Thứ nhất là nhóm các phương pháp trên miền không gian. Khi đó các giá trị cường độ của các pixel được thay đổi để chứa thông tin cần giấu. Phương pháp đầu tiên và đơn giản nhất của nhóm này là sử dụng các bit ít quan trọng LSB (Least Significant Bit) để giấu thông tin. Phương pháp này có thể áp dụng cho các ảnh đen trắng, ảnh đa cấp xám hoặc ảnh màu,vì nó chỉ sử dụng các bit LSB nên có ưu điểm là chất lượng ảnh hầu như không thay đổi và với những cách thông thường khó phát hiện ra là nó được giấu tin hay không.

Tuy nhiên, nhược điểm chính của nó là dung lượng giấu tin thấp, kém bền vững trước các thao tác xử lý ảnh.

Thứ hai là nhóm các phương pháp giấu ảnh trên miền biến đổi. Trong đó các hệ số biến đổi được thay đổi để giấu thông tin. Thực tế cho thấy các phương pháp trong nhóm này có nhiều ưu điểm và hiệu quả hơn so với nhóm thứ nhất. Các phương pháp

trong nhóm này có khả năng cho dung lượng thông tin giấu cao và khả năng bền vững trước các tác động của bên ngoài cũng như các thao tác xử lý ảnh.

Tuy nhiên, việc thực hiện sẽ phức tạp hơn so với nhóm các phương pháp trên miền không gian.

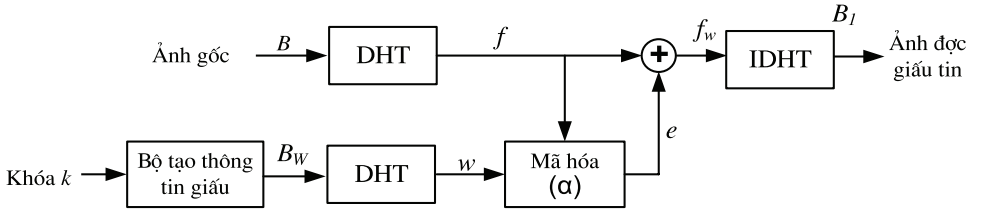
Trong các phương pháp miền biến đổi thì phương pháp sử dụng biến đổi Hadamard (DHT) có nhiều ưu điểm so với các phép biến đổi khác về sự đơn giản trong thực hiện, tốc độ xử lý nhanh và khả năng bền vững cao trước các tác động của các thao tác xử lý ảnh cũng như các tác động từ bên ngoài trong quá trình truyền dữ liệu ảnh số. Đặc biệt, với điều kiện thường gặp trong thực tế là tạp âm xử lý cao thì các nghiên cứu cho thấy DHT tỏ ra bền vững và hiệu quả hơn hẳn các phương pháp biến đổi khác [3].

Sơ đồ của quá trình nhúng và tách thông tin giấu trong ảnh sử dụng biến đổi DHT được thể hiện trong hình 1.

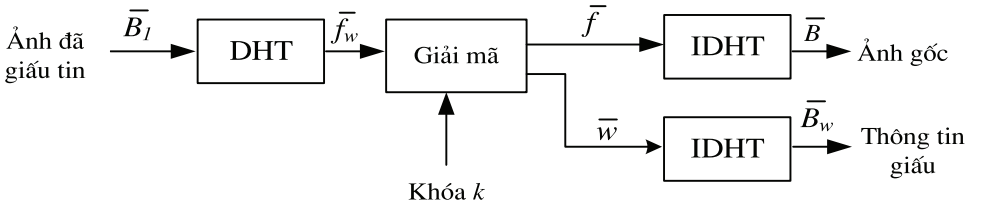
Kí hiệu giá trị các điểm ảnh của ảnh gốc là B và thông tin cần dấu là B_W. Cả hai đều được biến đổi DHT để được các giá trị hệ số biến đổi tương ứng là f và w. Bộ mã hóa sử dụng các hệ số này để mã hóa được giá trị hàm mã hóa: $e = \alpha \,.\, w \,.\, f$

Trong đó α là hệ số của bộ mã hóa, chẳng hạn chọn α = 0,1. Sau bộ cộng ta được hàm f_W: $f_W = f + \alpha \,.\, w \,.\, f$

Và sau biến đổi Hadamard ngược (IDHT) của f_w thu được B_1 là giá trị của ảnh đã được giấu tin, và từ đó ảnh này sẽ được gửi đi trên kênh truyền thông.



a. Quá trình nhúng thông tin giấu



b. Quá trình tách thông tin giấu

***Hình 1**. Quá trình nhúng và tách thông tin giấu trong ảnh sử dụng biến đổi DHT.*

Tại phía thu sẽ nhận được ảnh đã giấu tin với các giá trị điểm ảnh là $\overline{B}_1$ có thể sai khác so với B_1 do tác động của kênh truyền. Thực hiện biến đổi DHT trên ảnh này thu được các hệ số $\overline{f}_w$ đưa tới bộ giải mã cùng với khóa k giống hệt phía phát để tách riêng các hệ số biến đổi Hadamard của ảnh gốc ($\overline{f}$) và của thông tin giấu ($\overline{w}$). Từ đó, phép biến đổi DHT ngược (IDHT) được thực hiện trên các hệ số này để khôi phục lại ảnh gốc ($\overline{B}$) và thông tin giấu ($\overline{B}_w$).

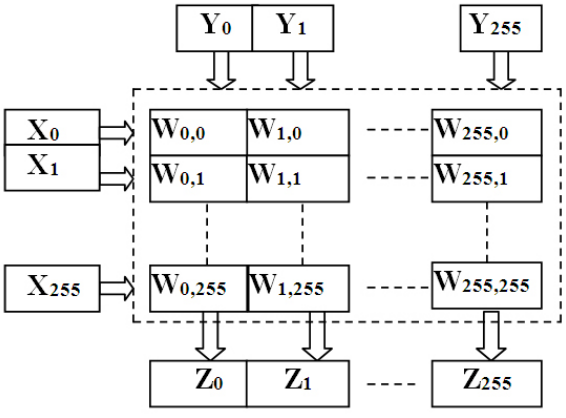
Có thể thấy rằng các khối biến đổi DHT và IDHT có vai trò rất quan trọng trong hệ thống giấu tin trong ảnh số. Do tính chất đặc biệt của ma trận Hadamard (chỉ gồm các phần tử 1, -1 và có tính đối xứng cao) nên các tính toán trong IDHT hoàn toàn giống với DHT. Vì vậy, trong phần sau của bài báo

chỉ xây dựng thuật toán xử lý DHT thuận, song kết quả này hoàn toàn có thể áp dụng cho IDHT.

I. Thiết kế khối biến đổi Hadamard tốc độ cao

Trong các khối chức năng trên hình 1 chỉ có khối DHT và IDHT gây ảnh hưởng nhiều nhất đến tốc độ xử lý của hệ thống. Mặt khác về nguyên tắc, việc thực hiện chức năng khối DHT cũng tương tự như IDHT nên nếu tổng hợp tối ưu được khối DHT thì sẽ giải quyết được vấn đề tốc độ xử lý chung. Nếu đầu vào là luồng dữ liệu vectơ 256 phần tử, mỗi phần tử là số nguyên có dấu kích thước 8 bit. Do luồng dữ liệu đi vào khối ALU là dữ liệu vector nên phép xử lý bằng cấu trúc ma trận lưới thao tác rất phù hợp với việc xử lý thuật toán biến đổi Hadamard DHT. Thực hiện DHT 256 điểm trên khối ALU với dữ liệu đầu vào X, Y là một vectơ có kích thước N=256.

Ma trận Hadamard tương ứng phải có kích thước 256x256 phần tử, mỗi phần tử là một trọng số $W_{i,j}$. Kết quả phép biến đổi DHT là một vectơ cũng có chiều dài N = 256, mà mỗi phần tử là một số nguyên có dấu được biểu diễn bằng một word 16 bit. Nguyên lý của phép biến đổi này được thể hiện trên hình 2.



256 phần tử.

Theo tọa độ i,j với chiều cột i được đánh số từ 0 đến 255 và hàng j được đánh số từ 0 đến 255 thì kết quả biến đổi DHT Z(i) được tính

$$Z(i) = Y_i + \sum_{j=0}^{255} W_{i,j} \times X_i$$

N ể lập trình (chỉ viết phần vòng lặp):

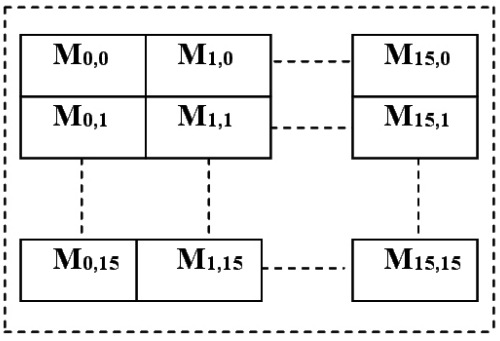
```
for i:=0 to 255 do
      begin
        for j:=0 to 255 do
        begin
        Z[i]:=Z[i]+W[i,j]*X[i];
end;//for j:=0 to 7 do
      end;//for i:=0 to 7 do
```

Nếu sử dụng ngôn ngữ bậc thấp Assembly để lập trình (chỉ viết phần vòng lặp):

```
MOV CX, 256;
Label1:
    MOV SI, 256
    Label2:
       MOV AL, X[CX];
       MUL  W[(256-CX)+SI];
       ADD  Z[CX], AX;
        DEC SI
        JNZ  Label2
   LOOP  Label1;
```

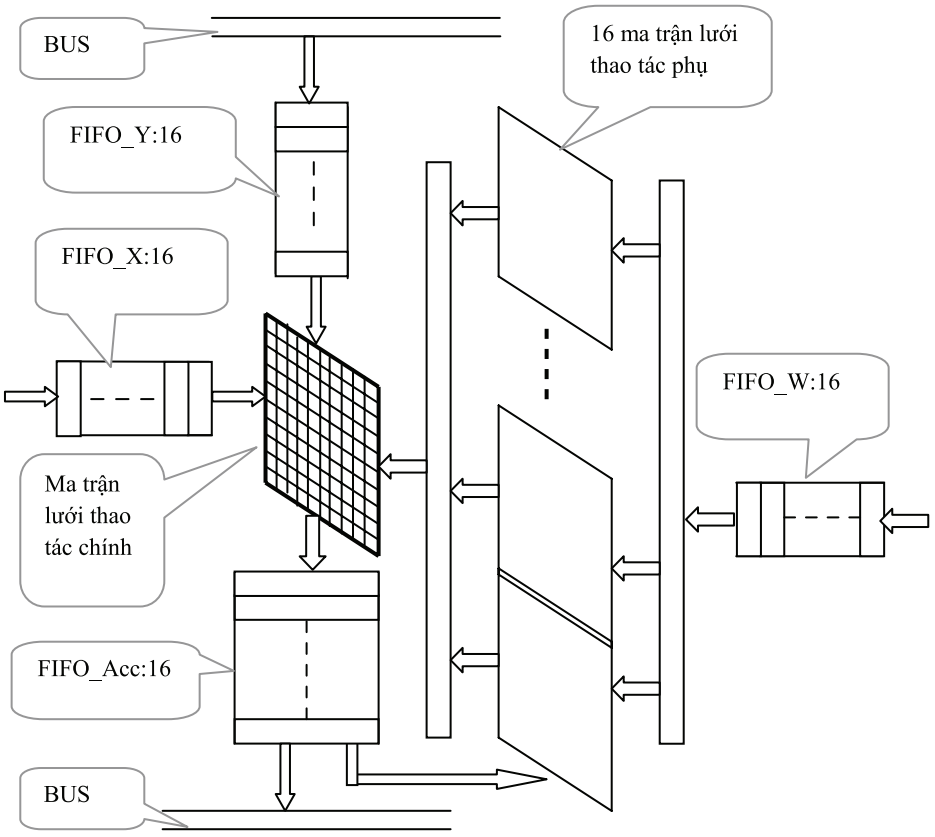
Do ma trận thao thác được thiết kế có kích thước 16x16 nên ma trận tổng chia thành 256 ma trận con được đánh số theo qui luật cột trước hàng sau $\mathbf{H_{0,0}} - \mathbf{H_{0,15}}$, $\mathbf{H_{1,0}} - \mathbf{H_{1,15}}$,... $\mathbf{H_{15,0}} - \mathbf{H_{15,15}}$.

Với mỗi tập hợp theo cột của ma trận con (16 ma trận con) sẽ kết xuất được 16 phần tử vector kết quả trung gian.



Hình 3. *Ma trận tổng chia thành 256 ma trận con kích thước 16x16 phần tử vector*

Và cấu trúc khối tính toán ALU với ma trận lưới thao tác được thiết kế như hình 4:



Hình 4. *Khối biến đổi Hadamard tốc độ cao*

Tuy nhiên, trong thực tế không thể tổ chức lưới thao tác kiểu ma trận kích thước quá lớn nên trong khối ALU của CPU sẽ được thiết kế một ma trận lưới thao tác có chức năng là ma trận trọng số kích thước tính theo bit là hàng x cột=128×256 bit có khả năng tái kiến trúc bằng phần mềm cho kích thước dữ liệu được xử lý [1,2]. Kích thước ma trận như vậy là phù hợp với công nghệ thiết kế vi mạch đương đại. Đối với cấu trúc này, ở lối vào là FIFO_X sẽ dẫn 16 nhóm, mỗi nhóm 16 phần tử lần lượt đi vào ma trận lưới thao tác có chức năng là ma trận trọng số. Mỗi lần xử lý được 16 phần tử 8 bit và tạo ra 16 phần tử 16 bit ở lối ra và lần lượt đi vào bộ đệm FIFO_Acc ở lối ra. Lưu ý rằng bộ đệm FIFO_X và FIFO_Acc đều có kích thước 16 ngăn nhưng mỗi ngăn của bộ đệm FIFO_X là 128 bit còn bộ đệm FIFO_Acc 256 bit tương ứng.

Qui luật thành lập ma trận Hadamard có thể thấy qua hình 5.

Ta có lần lượt $\mathbf{H}_2$, $\mathbf{H}_{2N}$, $\mathbf{H}_{16}$ như sau:

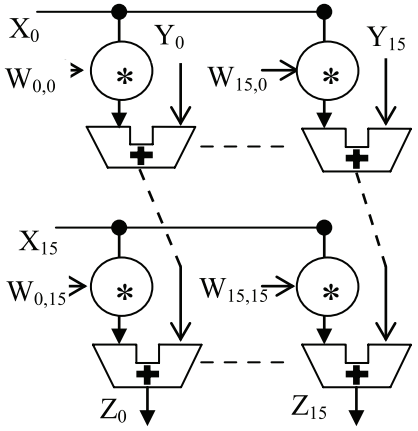
$$H_2 = \frac{1}{\sqrt{2}} \begin{vmatrix} 1 & 1 \\ 1 & -1 \end{vmatrix}$$

$$H_{2N} = \frac{1}{\sqrt{2}} \begin{vmatrix} H_N & H_N \\ H_N & -H_N \end{vmatrix}$$

$$H_{16} = \frac{1}{\sqrt{16}} \begin{vmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\
1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\
1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\
1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\
1 & -1 & 1 & 1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 & 1 \\
1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\
1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\
1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 \\
1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 \\
1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 \\
1 & -1 & 1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & -1 \\
1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 \\
1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1
\end{vmatrix}$$

Hình 5. *Ma trận Hadamard 16x16.*

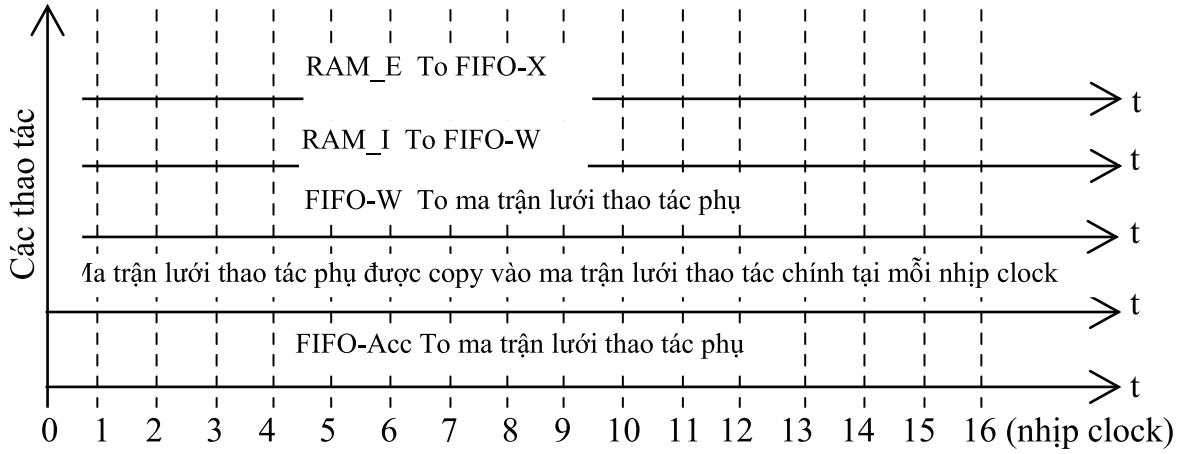
Khối ALU với các các phép tính nhân và cộng số có dấu được thể hiện trên hình 6.



Hình 6. *ALU với ma trận biến đổi Hadamard kích thước 16x16.*

I. THIẾT KẾ CẤU TRÚC LỆNH SONG SONG

Trên cơ sở của khối biến đổi Hadamard tốc độ cao (hình 4) và kiến trúc ALU với ma trận biến đổi Hadamard kích thước 16x16 có thể thiết lập các lệnh thực hiện các thao tác xếp chồng về mặt thời gian (lệnh xử lý song song). Đồ thị thời gian cho các thao tác xếp chồng được biểu diễn trên hình 7.



Hình 7. *Các thao tác song song của ALU Hadamard kích thước 16x16*

Để xây dựng phần mềm cho kiến trúc xử lý này, trước hết cần xây dựng cơ sở dữ liệu trọng số của các ma trận con Hadamard theo cấu trúc sau (0001h tương đương với +1, còn 0ffffh tương đương với -1):

```
Weight0   DW 0001h, 0001h, 0001h, 0001h, 0001h, 0001h, 0001h, 0001h
          DW 0001h, 0ffffh, 0001h, 0ffffh, 0001h, 0ffffh, 0001h, 0ffffh
          DW 0001h, 00001h, 0ffffh, 0ffffh, 0001h, 0001h, 0ffffh, 0ffffh
          DW 0001h, 0ffffh, 0ffffh, 0001h, 0001h, 0ffffh, 0ffffh, 0001h
          DW 0001h, 0001h, 0001h, 0001h, 0ffffh, 0ffffh, 0ffffh, 0ffffh
          DW 0001h, 0ffffh, 0001h, 0001h, 0ffffh, 0001h, 0ffffh, 0001h
          DW 0001h, 0001h, 0ffffh, 0ffffh, 0ffffh, 0ffffh, 0001h, 0001h
          DW 0001h, 0ffffh, 0ffffh, 0001h, 0ffffh, 0ffffh, 0ffffh, 0ffffh

          DW 0001h, 0001h, 0001h, 0001h, 0001h, 0001h, 0001h, 0001h
          DW 0001h, 0ffffh, 0001h, 0ffffh, 0001h, 0ffffh, 0001h, 0ffffh
          DW 0001h, 00001h, 0ffffh, 0ffffh, 0001h, 0001h, 0ffffh, 0ffffh
          DW 0001h, 0ffffh, 0ffffh, 0001h, 0001h, 0ffffh, 0ffffh, 0001h
```

DW 0001h, 0001h, 0001h, 0001h, 0ffffh, 0ffffh, 0ffffh, 0ffffh
DW 0001h, 0ffffh, 0001h, 0001h, 0ffffh, 0001h, 0ffffh, 0001h
DW 0001h, 0001h, 0ffffh, 0ffffh, 0ffffh, 0ffffh, 0001h, 0001h
DW 0001h, 0ffffh, 0ffffh, 0001h, 0ffffh, 0ffffh, 0ffffh, 0ffffh

DW 0001h, 0001h, 0001h, 0001h, 0001h, 0001h, 0001h
DW 0001h, 0ffffh, 0001h, 0ffffh, 0001h, 0ffffh, 0001h, 0ffffh
DW 0001h, 00001h, 0ffffh, 0ffffh, 0001h, 0001h, 0ffffh, 0ffffh
DW 0001h, 0ffffh, 0ffffh, 0001h, 0001h, 0ffffh, 0ffffh, 0001h
DW 0001h, 0001h, 0001h, 0001h, 0ffffh, 0ffffh, 0ffffh, 0ffffh
DW 0001h, 0ffffh, 0001h, 0001h, 0ffffh, 0001h, 0ffffh, 0001h
DW 0001h, 0001h, 0ffffh, 0ffffh, 0ffffh, 0ffffh, 0001h, 0001h
DW 0001h, 0ffffh, 0ffffh, 0001h, 0ffffh, 0ffffh, 0ffffh, 0ffffh

DW 0ffffh, 0ffffh, 0ffffh, 0ffffh, 0ffffh, 0ffffh, 0ffffh, 0ffffh
DW 0ffffh, 0001h, 0ffffh, 0001h, 0ffffh, 0001h, 0ffffh, 0001h
DW 0ffffh, 0ffffh, 0001h, 0001h, 0ffffh, 0ffffh, 0001h, 0001h
DW 0ffffh, 0001h, 0001h, 0ffffh, 0ffffh, 0001h, 0001h, 0ffffh
DW 0ffffh, 0ffffh, 0ffffh, 0ffffh, 0001h, 0001h, 0001h, 0001h
DW 0ffffh, 0001h, 0ffffh, 0ffffh, 0001h, 0ffffh, 0001h, 0ffffh
DW 0ffffh, 0ffffh, 0001h0001h, 0001h, 0001h, 0ffffh, 0ffffh
DW 0ffffh, 0001h, 0001h, 0ffffh, 0001h, 0ffffh, 0ffffh, 0001h

Weight1 ...

Nếu xử lý theo cột để bảo đảm xử lý được 16 phần tử vector kết quả cuối cùng cần tiến hành các bước sau:

Bước 1. Đưa vào FIFO_Y các gia trị 0. Xử lý theo thuật toán Hadamar cho 16 ma trận con (lần đầu là 16 ma trận con $\mathbf{M_{0,0}}$ – $\mathbf{M_{0,15}}$) sẽ 16 phần tử vector kết quả trung gian. Kết quả này chứa trong 16 ngăn nhớ của FIFO_Acc, đồng thời thực hiện chuyển kết quả trong FIFO_Acc vào FIFO_W, từ đó chuyển tiếp vào ma trận lưới thao tác phụ.

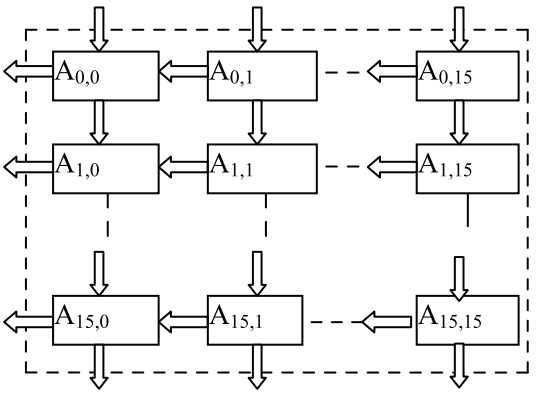
Bước 2. Đưa vào FIFO_X các gia trị +1, vào FIFO_Y các gia trị 0. Thực hiện cộng theo cột sẽ được 16 phần tử vector kết quả cuối cùng và copy chúng vào bộ nhớ RAM.

Lặp lại 16 lần bước 1 và bước 2 sẽ được 256 phần tử vector kết quả cuối cùng chứa trong bộ nhớ RAM.

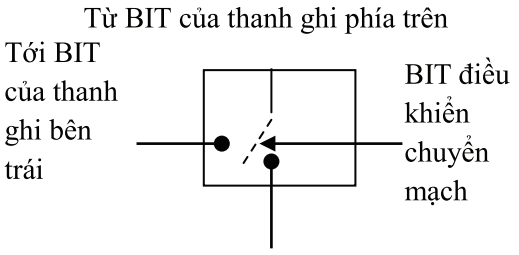
Các tiến trình sẽ được thực hiện song song trên cấu trúc của khối biến đổi tốc độ cao:

-16 x16 nhịp đầu sẽ tự động thực hiện việc xử lý từng 16 phần tử vector X đầu vào với ma trận Hadamard con $\mathbf{H_{i,j}}$ tương ứng. $\mathbf{H_{i,j}}$ tương ứng được nạp trước vào ma trận lưới thao tác phụ rồi copy vào ma trận lưới thao tác chính ở sau ngay khi sườn xuống của mỗi nhịp clock thứ.

Sử dụng bộ đệm FIFO_Acc đặc biệt ở lối ra để chuyển từ hàng thành cột cho mỗi lần xử lý 16 ma trận con theo cột (lần đầu là 16 ma trận con $\mathbf{H_{0,0}}$ –$\mathbf{H_{0,15}}$).



(a)



(b)

Hình 7. *a) Cơ chế chuyển từ hàng thành cột của FIFO_Acc.*
b) BIT điều khiển chuyển mạch từ hàng thành cột của FIFO_Acc.

Thiết kế các lệnh song song: Các lệnh vector của khối chia thành vế trái và vế phải nhưng có thêm trường phụ mà mọi lệnh vector đều phải có. Trường này chứa thông tin về số lần lặp. Một lệnh vector có thể xử lý 16 word kích thước 16 bit. Các lệnh này thực hiện theo nguyên tắc SIMD [3,4]. Cụ thể:

```
// Nạp địa chỉ mảng nhớ Ram chứa ma trận trọng số Hadamard đầu, Khởi tạo gr4=1
   ar0 = Weight0 with gr4++;
   CX=16;
HadamarTrans:
   // Nạp 16 word vào FIFO_W, chuyển 16 word tới ma trận lưới thao tác phụ
   // và copy nội dung của ma trận lưới thao tác phụ vào ma trận lưới thao tác chính
   rep 16 wfifo = [ar0++], ftw, wtw;
   // Nạp địa chỉ của mảng nhớ Ram chứa dữ liệu nguồn, thanh ghi gr4 = 2
   ar0 = [--ar5] with gr4 <<= 1;
   // Nạp địa chỉ của mảng nhớ Ram chứa dữ liệu đích, gr5 = 4; <- Increment value
   ar4 = [--ar5] with gr5 = gr4 << 1;
   // ar5 trỏ vào địa chỉ bộ đệm đích + 2
   ar5 = ar4 + gr4 with gr4 = gr5;
   // Tải dữ liệu nguồn, thực hiện các tính toán
   // và chuyển 16 word từ FIFO_W tới ma trận lưới thao tác phụ
    rep 16 ram = [ar0++], ftw with vsum , data, 0;
   // Lưu các kết quả vào bộ nhớ. Hai lệnh này được thực thi song song
   rep 16 [ar4++gr4] = FIFO_Acc;
   ar0 = ar0 +256
   LOOP HadamarTrans
```

I. ĐÁNH GIÁ KẾT QUẢ

Khối biến đổi Hadamard tốc độ cao với nhịp clock 40 MHZ hoàn thành biến đổi DHT chỉ cần 426 chu kỳ clock (mất 10,65 micro sec) để hoàn thành tính toán DHT 256 điểm. Nếu sử dụng máy tính Pentium với nhịp clock 2660 MHz (gấp 66,5 lần tần số của khối biến đổi Hadamard) cần thời gian tính toán DHT 256 điểm hết 16,4 micro sec.

Cơ cấu xử lý	ALU của khối biến đổi Had-amard với nhịp clock 40 MHZ	Pentium IV với nhịp clock 2660 MHz
Thời gian chạy tính theo micro sec	10,65	16,4
Số chu kỳ clock	426	$4,36.10^4$

Kết quả này đã minh chứng cho hiệu năng tính toán mạnh mẽ trong các thao tác tính toán vectơ của khối biến đổi Hadamard tốc độ cao.

Như vậy, với mục đích xây dựng khối biến đổi Hadamard tốc độ cao để xử lý thuật toán DHT trong các kỹ thuật xử lý số tín hiệu. Kết quả chạy chương trình cho thấy với cùng một bài toán tính DHT 256 điểm, thời gian tính toán của khối biến đổi Hadamard tốc độ cao chỉ bằng hơn nửa thời gian cần thiết cho bộ xử lý Pentium 2,66 GHz. Điều này đã minh chứng khối biến đổi Hadamard tốc độ cao hơn hẳn so với các thiết bị xử lý khác trong các thao tác tính toán với dữ liệu vectơ.

I. KẾT LUẬN:

Sử dụng phương pháp kết hợp tổ chức phần cứng và xây dựng phần mềm (phương pháp đồng tổng hợp) trong thiết kế khối biến đổi Hadamard ở mức ALU với ma trận lưới thao tác cho phép đạt tốc độ xử lý cao.

Tần số làm việc của khối biến đổi Hadamard không đòi hỏi cao nên hệ thống làm việc ổn định và tin cậy.

TÀI LIỆU THAM KHẢO

1.Chun-Shien Lu, *Multimedia Security Steganography and Digital Water-marking Techniques for Protection of Intellectual Property*, Idea Group Publishing, London 2005.

2. Sergey Mushkaev and Sergey Landyshev*, Implementing image compression algorithms on NeuroMatrix architecture*: *New approaches*, Module Research Center, Russia 2004.

3. Saeid Saryazdi, Hossein Nezamabadi-pour, *A Blind Digital Watermark in Hadamard Domain*, Transaction on Engineering, Computing and Technology, Iran 2004.

4. Martin Schmucker, Michael Arnold, Stephen D.Wolthusen, *Techniques and Applications of Digital Watermarking and Content Protection*, Artech House, London 2003.