MÃ HÓA VÀ THỰC HIỆN PHẦN CỨNG CHO BỘ MÃ HÓA TIER-2 TRONG HỆ THỐNG NÉN ẢNH CHUẨN JPEG2000

PHẠM THỊ THU HƯỜNG, NGUYỄN MINH KHÁNH NGỌC
Trung tâm Nghiên cứu và đào tạo thiết kế vi mạch (ICDREC)
Đại học Quốc gia TP Hồ Chí Minh

Trong chuẩn nén ảnh JPEG2000, Tier-2 là khối mã hóa cuối cùng có nhiệm vụ sắp xếp và đóng gói dữ liệu sau quá trình nén. Bài báo này phân tích chi tiết nguyên tắc sắp xếp dữ liệu và đóng gói dữ liệu trong trường hợp ảnh có nhiều mức biến đổi wavelet khác nhau tương ứng với năm tiến trình sắp xếp khác nhau được hỗ trợ trong tiêu chuẩn ISO/IEC 15444-1:2000. Bên cạnh đó, các tác giả đã xây dựng phần cứng, thiết kế một lõi Tier-2 sử dụng trong nén ảnh tuân theo chuẩn JPEG2000. Sau quá trình kiểm tra chức năng trên phần mềm ModelSim, lõi cứng Tier-2 được thi hành trên chip FPGA của Altera với tốc độ hơn 50 MHz và tiêu tốn ít tài nguyên. Lõi cứng Tier-2 đáp ứng đầy đủ yêu cầu của khối sắp xếp trong hệ thống nén ảnh theo chuẩn JPEG2000 hoạt động với tốc độ 50 MHz.

***Từ khóa*: Tier-2, JPEG2000, coding pass, sắp xếp chuỗi mã bit, tiến trình chất lượng, tiến trình quality, tiến trình resolution, tiến trình position, tiến trình component, gói, lớp.**

CODING TECHNIQUE AND HARDWARE IMPLEMENTATION OF TIER-2 CODING IN JPEG2000 ENCODER

Summary

In JPEG2000 Image Compression Standard, Tier-2 is the last coding block for arranging and packaging the compressed data. This paper analyzes in detail the coding techniques: arranging and packaging the compressed data in case of an image having various wavelet transform levels within five different arranging progressions supported in ISO/IEC 15444-1:2000 standard. In addition, a hardware implementation of a Tier-2 core used in JPEG2000 Encoder is presented and the verification is based on the ModelSim tools. The Tier-2 core operates at more than 50 MHz and consumes little resources on FPGA chip of Altera. Therefore, this core meets the requirements of 50 MHz JPEG2000 Encoder System.

***Keywords:* Tier-2, JPEG2000, coding pass, codestream reorganization, quality progression, resolution progression, position progression, component progression, packet, layer.**

I. GIỚI THIỆU

Năm 2000, tiêu chuẩn JPEG2000 - Part 1 được xuất bản. Tiêu chuẩn này được thiết kế mới dựa trên nền phương pháp wavelet. Các phần tiếp theo của tiêu chuẩn JPEG2000 được xuất bản vào các năm tiếp sau [1]. Cũng giống như các tiêu chuẩn nén ảnh và video khác (JPEG, MPEG-1, MPEG-2, MPEG-4), mục đính chính của JPEG2000 là làm giảm kích thước của ảnh gốc bằng các thuật toán và quá trình mã hóa phức tạp mà vẫn duy trì được một mức độ chất lượng ảnh chấp nhận được. Với JPEG2000 kỹ thuật xử lý hình ảnh sẽ đạt được những kết quả rất ngoạn mục vì có thể nén nhỏ từ 100-200 lần mà hình ảnh không sai sót bao nhiêu so với hình ảnh gốc. Thuật toán trong kỹ thuật JPEG2000 là chọn một số nhỏ các sóng ngắn (wavelet), các sóng này được lập lại ở những nơi khác nhau, tỷ lệ khác nhau để mô tả chính xác tín hiệu của hình ảnh [2].

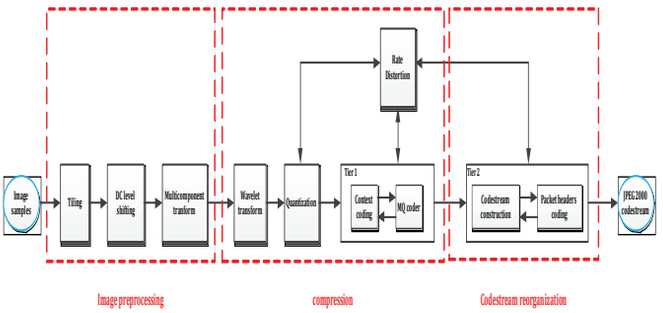
Về cơ bản, có thể phân chia hệ thống

nén ảnh JPEG2000 thành ba phần chính là tiền xử lý ảnh (image preprocessing), nén (compression) và sắp xếp dữ liệu (codestream reorganization).

- Bước tiền xử lý ảnh bao gồm ba chức năng chính là Tiling, DC Level shifting và Multicomponent Transformation.

- Quá trình nén (compression) ảnh trải qua ba bước chính là biến đổi wavelet rời rạc (discrete wavelet transform), lượng tử hóa (quantization) và mã hóa Tier-1.

- Sắp xếp dữ liệu (codestream reorganization): Tier-2 đóng gói các thông tin tổng hợp cho mỗi code block cùng với dữ liệu nén tạo thành các gói và sắp xếp các gói vào file JPEG2000 cuối cùng [4].



Hình 1: hệ thống nén ảnh JPEG2000

II. LÝ THUYẾT BỘ MÃ HÓA TIER-2

Tier-2 bao gồm hai nhiệm vụ chính: sắp xếp các chuỗi mã bit (codestream) đã được mã hóa vào các lớp (layer) và gói chúng thành các gói dữ liệu gọi là packet. Dựa trên các nguyên tắc sắp xếp chuỗi mã bit này mà ảnh JPEG2000 có khả năng "compress one, decompress many ways" (tạm dịch: nén bằng một cách nhưng giải nén với nhiều cách).

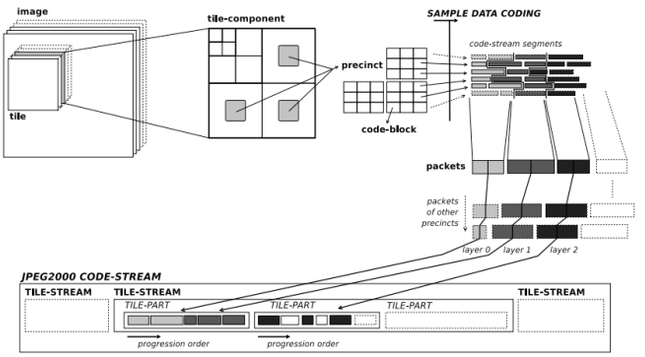
II.1. Sắp xếp dữ liệu

Sau quá trình nén, dữ liệu đi vào Tier-2 là các đoạn chuỗi mã bit (codestream segment) (hình 2). Tier-2 chia các đoạn chuỗi mã bit này vào các gói (packet) khác nhau và sắp xếp các gói đó vào file dữ liệu của JPEG2000 [5].

Dựa trên các mục đích khác nhau sau khi nén mà cách sắp xếp dữ liệu nén của JPEG2000 cũng khác nhau. Chuẩn JPEG2000 hỗ trợ năm tiến trình sắp xếp dữ liệu với các mục đích khác nhau [2].

Xét các gói thuộc tile **t**, component **c**, resolution **r**, precinct **p** và layer **l**. Gọi gói đó là: $\mathcal{G}_{t,c,r,p,l}$.

Tile **t** có tọa độ **[t1, t2]** trong ảnh. Precinct **p** có tọa độ **[p1, p2]** trong tile **t.**



Hình 2: đường dữ liệu của chuỗi mã bit JPEG2000

Điều kiện tồn tại gói $\mathcal{G}_{t,c,r,p,l}$ là:

$$0 \le t1 < N_1^T;\ 0 \le t2 < N_2^T;$$
$$0 \le c < C;\ 0 \le r < D_{t,c} + 1;$$
$$0 \le p1 < N_1^{p,t,c,r};\ 0 \le p2 < N_2^{p,t,c,r}$$
$$0 \le l < \Lambda_t$$

Với:

- N_1^T và N_2^T là hệ số số lượng theo hàng và cột của tile trong ảnh.

- **C** là số lượng component của ảnh.

- $D_{t,c} + 1$ là mức resolution của component **c**, của tile **t**.

- $D_{t,c}$ là mức phân tích wavelet.

- $N_1^{p,t,c,r}$ và $N_2^{p,t,c,r}$ là số lượng precinct theo hàng và cột tại resolution **r** của component **c** của tile **t**.

Để đơn giản, xét trường hợp một gói là một layer của một precinct của một resolution của một component của một tile. Mỗi component của một tile (tile-component) của một ảnh có thể có nhiều mức phân tích wavelet khác nhau. Đặt: $D_{t,max} = max_c\{D_{t,c}\}$

Xem xét năm tiến trình sắp xếp trong chuẩn JPEG2000:

1. Tiến trình Quality (LRCP: layer, resolution, component, position)

Tiến trình quét:

For l = 0, 1, ..., $\Lambda_t - 1$
For r = 0, 1, ..., $D_{t,max}$
For c = 0, 1, ..., C - 1
If(r < $D_{t,c}$)
For p1 = 0, 1, ..., $N_1^{p,t,c,r}$ - 1
For p2 = 0, 1, ..., $N_2^{p,t,c,r} - 1$
Include $\mathcal{G}_{t,c,r,p,l}$

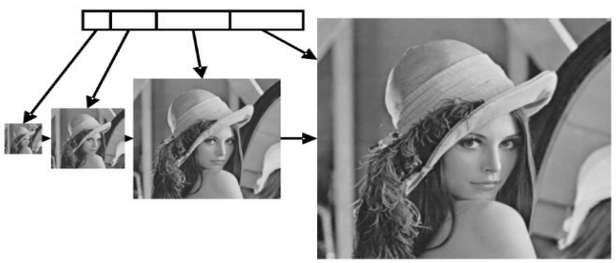
Hình 3: hình nén với các tỷ lệ nén khác nhau theo tiến trình quality

Tiến trình này còn được gọi là tiến trình quality. Dữ liệu được quét theo từng layer. Tất cả dữ liệu thuộc layer 0 của toàn bộ các codeblock của tất cả các resolution của tất cả các component sẽ được ghi vào trong chuỗi mã bit trước khi dữ liệu thuộc layer 1 của toàn bộ các codeblock được ghi vào codestram. Mỗi code block bên trong mỗi precinct có thể đóng góp một số lượng coding pass khác nhau cho mỗi layer được xét.

2. Tiến trình Resolution 1 (RLCP: resolution, layer, component, position)

Tiến trình quét:

```
For r = 0, 1, ..., D_{t,max}
    For l = 0, 1, ..., Λ_t − 1
        For c = 0, 1, ..., C - 1
            If(r < D_{t,c})
                For p1 = 0, 1, ..., N_1^{p,t,c,r} - 1
                    For p2 = 0, 1, ..., N_2^{p,t,c,r} − 1
                        Include g_{t,c,r,p,l}
```



Hình 4: hình nén với các kích thước khác nhau khi đọc chuỗi mã bit theo tiến trình Resolution

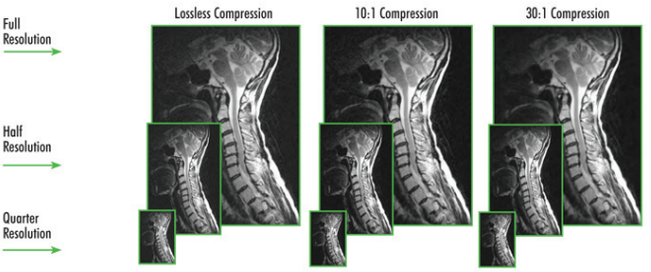
Tiến trình này còn được gọi là "tiến trình resolution". Khi quét từng resolution, toàn bộ các layer của các codeblock thuộc resolution 0 bao gồm toàn bộ các component của tile được ghi trước khi resolution 1 được ghi vào.

3. Tiến trình Resolution 2 (RPCL: resolution, position, component, layer)

Tiến trình quét:

Tiến trình này cũng được gọi là "tiến trình resolution".

```
For r = 0, 1, ..., D_{t,max}
    If(r < D_{t,c})
        For p1 = 0, 1, ..., N_1^{p,t,c,r} - 1
            For p2 = 0, 1, ..., N_2^{p,t,c,r} − 1
                For c = 0, 1, ..., C - 1
                    For l = 0, 1, ..., Λ_t − 1
                        Include g_{t,c,r,p,l}
```



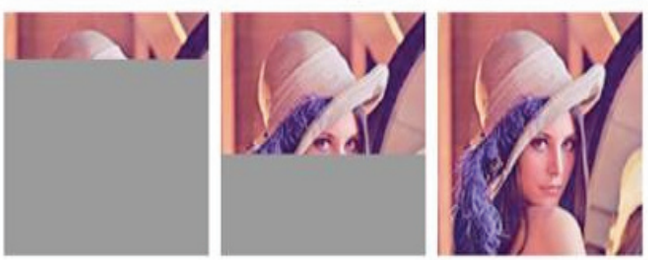
Hình 5: hình ảnh với tỷ lệ nén và độ phân giải khác nhau được chiết xuất từ một chuỗi bit nén tổng thể

Trong tiến trình resolution ở phần 2, khi ghi một resolution, tiến trình ghi được tiến hành theo từng layer. Ở tiến trình này, tiến trình ghi được tiến hành theo từng position. Tất cả các layer của tất cả các component của từng position được ghi trước khi xét sang một position khác.

4. Tiến trình Position (PCRL: position, component, resolution, layer)

Tiến trình quét:

```
For p1 = 0, 1, ..., N_1^{p,t,c,r} - 1
    For p2 = 0, 1, ..., N_2^{p,t,c,r} − 1
        For c = 0, 1, ..., C - 1
            For r = 0, 1, ..., D_{t,max}
                If(r < D_{t,c})
                    For l = 0, 1, ..., Λ_t − 1
                        Include g_{t,c,r,p,l}
```



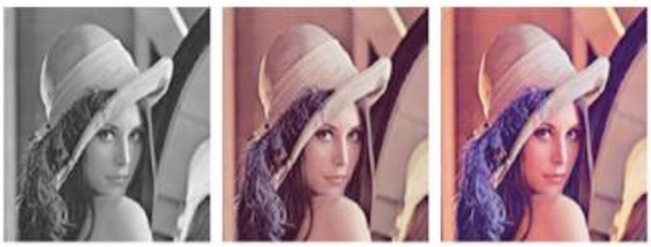
Hình 6: hình nén với các vùng khác nhau theo tiến trình Position

Tiến trình này còn được gọi là "tiến trình position" hoặc "tiến trình không gian". Với từng vùng precinct được chọn, chuỗi mã bit sẽ ghi theo tiến trình từ trên của tile xuống dưới tile và từ trái sang phải. Tiến trình này thường được sử dụng khi quá trình tiling không được áp dụng.

5. Tiến trình Component (CPRL: component, position, resolution, layer)

Tiến trình quét:

```
For c = 0, 1, ..., C - 1
For p1 = 0, 1, ..., N_1^{p,t,c,r} - 1
    For p2 = 0, 1, ..., N_2^{p,t,c,r} - 1
        For r = 0, 1, ..., D_{t,max}
            For l = 0, 1, ..., Λ_t - 1
                Include g_{t,c,r,p,l}
```



Hình 7: hình nén với các màu khác nhau theo tiến trình Component

Tiến trình này còn được gọi là tiến trình component. Tất cả các gói của component 0 của tile sẽ được ghi trước khi dữ liệu nén thuộc component 1 được ghi vào trong chuỗi bit (bitstream). Trong quá trình nén ảnh BMP, component 0 là thành phần Y (luminance component), bao gồm tất cả các gói chứa thông tin ảnh (thành phần grayscale) được ghi trước khi các component chứa thông tin về các thành phần màu được ghi vào trong chuỗi mã bit.

II.2. Đóng gói và sắp xếp dữ liệu nén

1. Đóng gói dữ liệu thành gói [6]

Sau khi được nén, chuỗi bít nén của mỗi codeblock được phân tách vào các layer l với $0 \le l \le \Lambda_t - 1$.

Gọi z_i^l: số lượng coding pass của codeblock B_i sau quá trình nén đóng góp vào layer Q_0 đến layer Q_l.

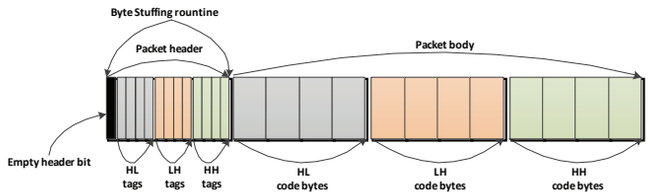
→ Số lượng coding pass của codeblock B_i đóng góp vào layer l : $\Delta z_i^l = z_i^l - z_i^{l-1}$

$L_i^{z_i^l}$ là tổng số code bytes của z_i^l coding pass.

→ Số lượng code byte đóng góp vào layer Q_l của Δz_i^l coding pass là: $\Delta L_i^{z_i^l} = L_i^{z_i^l} - L_i^{z_i^{l-1}}$

Khi $z_i^l = z_i^{l-1}$ thì ta nói codeblock B_i không đóng góp vào layer Q_l.

Đơn vị cơ bản tổng hợp nên chuỗi mã bit của JPEG2000 là các "**Gói**". Mỗi gói, $g_{t,c,r,p,l}$, chứa codeblock đóng góp cho một layer Q_l từ một thành phần màu, c, tại một resolution đơn, R_r, thuộc một precinct đơn, p.



Hình 8: cấu trúc gói [4]

Trong trường hợp đơn giản, xét ảnh bao gồm một thành phần màu và một precinct, một resolution tạo thành một gói. Và với trường hợp này, việc sắp xếp dữ liệu theo tiến trình quality và tiến trình resolution được ưu tiên. Với trường hợp nhiều thành phần màu và nhiều precinct, các tiến trình position và component được ưu tiên.

Một gói gồm có hai phần chính: thông tin đầu gói (packet header) và thông tin dữ liệu gói (packet body). Thông tin đầu gói chứa thông tin của gói và thông tin dữ liệu gói chứa dữ liệu nén. Bít đầu tiên của gói xác định gói có dữ liệu hay không. Bít này được gọi là Empty header bít, $e_{t,c,r,p,l}$.

- Nếu $e_{t,c,r,p,l} = 1$: gói có dữ liệu.
- Nếu $e_{t,c,r,p,l} = 0$: gói không có dữ liệu.

Trường hợp gói không có dữ liệu nghĩa là không có codeblock nào đóng góp dữ liệu vào gói này. Khi đó toàn bộ gói chỉ có một byte đơn bao gồm một bít MSB chỉ giá trị của $e_{t,c,r,p,l}$ và bảy bít còn được nhồi cho đủ một byte.

Khi gói có dữ liệu, kích thước và các thông tin khác của mỗi codeblock ghi trong tag bit sẽ được mã hóa và ghi trong thông tin đầu gói. Tổng hợp của empty header bit và các tag bit đã mã hóa của các khối dữ liệu tạo thành thông tin đầu gói. Có 4 thuật toán mã hóa được sử dụng để đóng gói các tag bit thành bốn gói thông tin (số lượng bitplane có nghĩa của coding pass; layer đầu tiên mà codeblock đóng góp dữ liệu; số lượng coding pass đóng góp vào trong layer đó; chiều dài (bytes) mà các coding pass đó đóng góp vào layer đang xét) trong thông tin đầu gói là:

a. Thuật toán mã hóa Tagtree

Tagtree là phương pháp mã hóa một mảng hai chiều số nguyên không âm theo cấu trúc phân cấp, nó lần lượt tạo ra sự suy hao độ phân giải của mảng hai chiều để hình thành cây. Ưu điểm của thuật toán mã hóa tagtree là sự linh động trong mã hóa dữ liệu và khả năng mã hóa tối ưu số bít cần thiết để biểu diễn một mảng dữ liệu hai

chiều.

Tagtree mã hóa hai thông tin của codeblock là: số lượng bitplane có nghĩa của coding pass và thông tin về layer đầu tiên mà codeblock đóng góp dữ liệu.

b. Thuật toán mã hóa số lượng coding pass

Thuật toán mã hóa số lượng coding pass (là thông tin Δz_i^l) như sau:

- Nếu $\Delta z_i^l = 1$ gửi một bít đơn "**0**".

- Nếu $\Delta z_i^l = 2$ gửi một bít đơn "**1**", theo sau bởi một bít đơn "**0**".

- Nếu $3 \le \Delta z_i^l \le 5$ gửi hai bít "**1**", theo sau bởi hai bít đơn biểu diễn thông tin $\Delta z_i^l - 3$.

- Nếu $6 \le \Delta z_i^l \le 36$ gửi 4 bít "**1**", theo sau bởi 5 bít đơn biểu diễn thông tin $\Delta z_i^l - 6$.

- Nếu $37 \le \Delta z_i^l \le 164$ gửi 9 bít "**1**", theo sau bởi 7 bít đơn biểu diễn thông tin $\Delta z_i^l - 37$.

Bởi vì số lượng coding pass lớn nhất được tạo bởi một codeblock là 109 coding pass [1] nên bảng mã hóa trên thỏa mãn mọi trường hợp của coding pass.

c. *Thuật toán mã hóa chiều dài (bytes)*

Chiều dài (bytes) $\Delta L_i^{z_i^l}$ của Δz_i^l đóng gói vào layer l là: $\Delta L_i^{z_i^l} = Lblock + \lfloor log_2(\Delta z_i^l) \rfloor$

Với $Lblock$ là trạng thái của codeblock ($Lblock$ được xét mặc định bằng 3).

d. Thuật toán ghép - nhồi bít

Nguyên tắc hoạt động của khối ghép - nhồi bít:

- Bits được gói thành byte từ MSB tới LSB.

- Khi một byte được tạo thành từ các bits, nó được gắn vào thông tin đầu gói.

- Nếu giá trị của byte mới được tạo đó là 0xFF, trong byte tiếp theo, một bít "**0**" (zero) được nhồi vào MSB của byte tiếp theo.

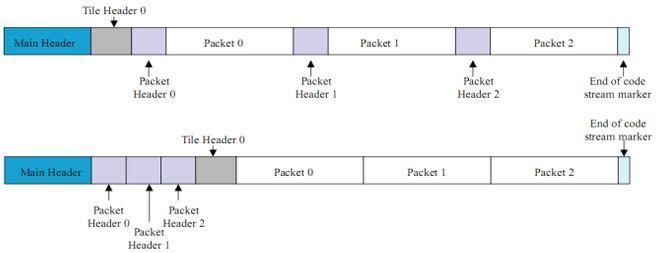
- Trong trường hợp tất cả các bít của thông tin đầu gói đã được nhúng, byte cuối cùng được nhồi bít để gói tới hết biên của byte đó và được ghi vào trong thông tin đầu gói.

- Byte cuối cùng của thông tin đầu gói sẽ không là 0xFF (vì một bít 0 luôn được nhồi vào byte sau một byte 0xFF, ngay cả khi 0xFF là byte cuối cùng). Trong trường hợp byte cuối cùng này là 0xFF, chuỗi bit của thông tin đầu gói sẽ được nhồi thêm một byte 0x00.

Mục đích của khối ghép - nhồi bít là ghép các bít dữ liệu của thông tin đầu gói sau khi mã hóa thành các bytes để ghi vào trong chuỗi bit cuối cùng.

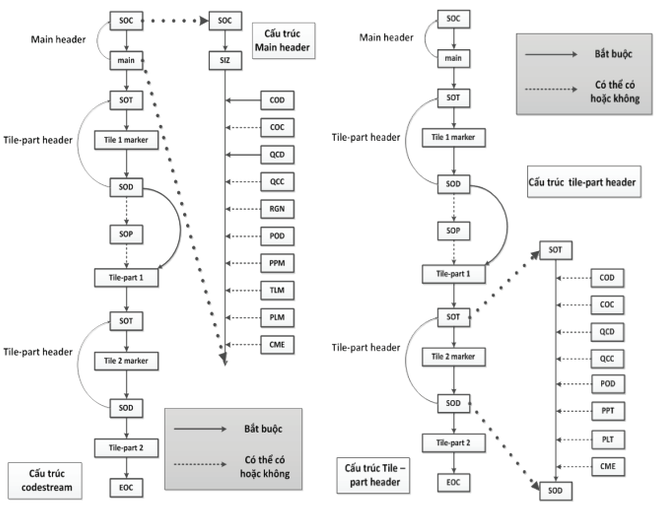
2. Cấu trúc chuỗi bit JPEG2000

Có hai cách sắp xếp các gói (thông tin đầu gói và thông tin dữ liệu gói) trong chuỗi bit JPEG2000 [3].



Hình 9: sắp xếp các gói trong chuỗi bit JPEG2000

Cấu trúc chi tiết chuỗi bit của JPEG2000 [2]:



Hình 10: cấu trúc chi tiết chuỗi bit của JPEG2000 (trong đó SOC, SOT, SOD, SOP, EOC, COD, COC, QCD, QCC... là các marker segment chứa toàn bộ thông tin của ảnh JPEG2000)

III. THIẾT KẾ KHỐI MÃ HÓA TIER-2

III.1. Cách xây dựng khối Tier-2 trên phần cứng

Dựa vào phân tích trên, một số vấn đề cần giải quyết khi xây dựng khối Tier 2 trên phần cứng là:

- Trước khi sắp xếp dữ liệu theo một trong năm cách, phải lưu toàn bộ dữ liệu nén của tất cả các thành phần màu của cùng một tile của ảnh. Sau đó, với cách sắp xếp khác nhau, các block dữ liệu nén sẽ được quét theo các cách khác nhau. Nhược điểm là sẽ tiêu tốn thời gian trong quá trình lưu trữ.

- Với trường hợp không tiling, phải lưu lại toàn bộ dữ liệu nén của ảnh. Với kích thước lớn nhất của ảnh là 4096 x 4096 thì kích thước của dữ liệu ảnh sau khi nén là khá lớn. Để giải quyết vấn đề về lưu trữ, một RAM ngoài được dùng để lưu trữ dữ liệu sau khi nén cho Tier 2.

- Phần chính của Tier-2 phải bao gồm bốn khối con chính:

+ Khối mã hóa thông tin của main header (main header chứa thông tin cơ bản của ảnh: chiều dài, chiều rộng, số thành phần màu, tỉ lệ lấy mẫu, vùng ROI…).

+ Khối mã hóa thông tin của tile - part header (tương tự như main header, tile - part header chứa thông tin mã hóa của riêng từng tile).

+ Khối đóng gói các gói.

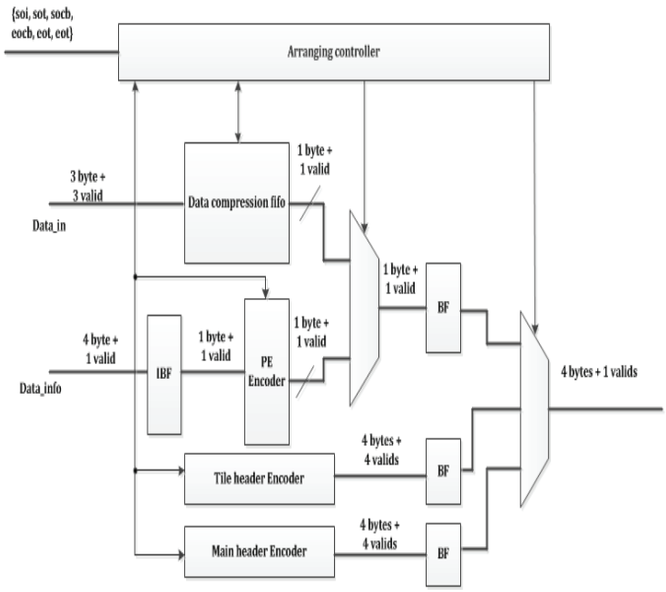
+ Khối Controller điều khiển quá trình quét và ghi.

Vậy khối Tier - 2 gồm có bốn phần chính:

- Main header Encoder.
- Tile header Encoder.
- Packet Encoder.
- Controller.

III.2. Thiết kế khối Tier-2

Dựa trên những phân tích này, tác giả xây dựng thiết kế phần cứng cho khối Tier-2 như sau:



Hình 11: sơ đồ phần cứng khối Tier-2

Trong đó:

- Khối Arranging controller làm nhiệm vụ giao tiếp với RAM ngoài, quét dữ liệu và điều khiển hoạt động của cả module Tier-2.

- Khối Data compression FIFO chứa dữ liệu nén sau khi quét hết một gói.

- Khối PE Encoder: mã hóa thông tin header của gói.

- Tile header Encoder: mã hóa thông tin header của tile đang quét.

- Main header Encoder: mã hóa thông tin header của toàn bộ ảnh.

- BF (Buffer) là các buffer có tác dụng ghép nối đường dữ liệu tương ứng giữa các khối con.

- Khối MUX: làm nhiệm vụ chuyển kênh, được điều khiển bởi khối Arranging Controller.

- Khối IBF (Input Buffer): lưu trữ các gói dữ liệu data info đầu vào của Tier-2.

Ngay khi cài đặt các thông số mã hóa cho ảnh (tiling, mức wavelet, mức lượng tử, tỷ lệ nén, vùng ROI…) và nhận thông tin ảnh (chiều dài, chiều rộng, thành phần màu…) khối Arranging controller cho phép dữ liệu từ Main Header ghi vào trong file JPEG2000 trước. Sau đó, thông tin của từng tile sẽ được ghi vào trong file JPEG2000 (trường hợp không tiling được coi như toàn bộ ảnh thuộc cùng một tile). Đi kèm với tile header (hình 10) là dữ liệu nén của từng tile. Các tile được ghi theo thứ tự từ trên xuống dưới, từ trái sang phải cho đến khi quét xong toàn bộ ảnh. Bên trong các dữ liệu tile là các gói (bao gồm thông tin đầu gói và thông tin dữ liệu gói) được ghi theo một trong hai cách như trong hình 9. Các thông tin đầu gói được ghi trước khi ghi thông tin dữ liệu gói vào trong file JPEG2000.

Quá trình đóng gói dữ liệu nén thành các gói và sắp xếp vị trí của các gói được khối Arranging Controller tiến hành dựa trên một trong năm tiến trình do khách hàng (user) lựa chọn (hoặc tiến trình phù hợp với mục đích nén). Quá trình đóng gói và sắp xếp được tiến hành như trong phần II.2.

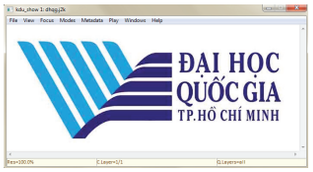
IV. KẾT QUẢ KIỂM TRA VÀ TỔNG HỢP PHẦN CỨNG

IV.1. Kết quả kiểm tra thiết kế

Sau khi hoàn thành thiết kế và viết code RTL bằng ngôn ngữ Verilog, lõi Tier-2 được kiểm tra chức năng bằng phần mềm mô phỏng ModelSim [9].

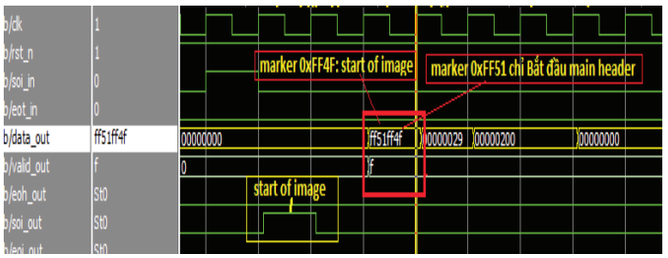
Trong môi trường kiểm tra này, tác giả sử dụng dữ liệu trong một ảnh chuẩn BMP là logo của Đại học Quốc gia TP Hồ Chí Minh, kích thước ban đầu là 677x291 (hình 12). Sau khi gán các header (main header, tile header, packet header) và sắp xếp lại dữ liệu thực hiện trong khối Tier-2, file ảnh chuẩn BMP sẽ chuyển thành file ảnh chuẩn JPEG2000 (hình 13) với tỉ lệ nén sáu lần. Việc hiển thị hình ảnh sử dụng phần mềm kakadu [8] - phần mềm hiển thị ảnh JPEG2000 trong hệ điều hành Windows. Ta có kết quả kiểm nghiệm như sau:

Hình 12: ảnh BMP ban đầu (677x291)



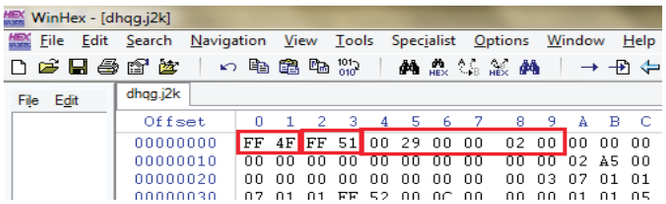
Hình 13: ảnh sau khi sắp xếp dữ liệu theo chuẩn JPEG2000

Kết quả quá trình sắp xếp dữ liệu trên ModelSim:



Hình 14: kết quả mô phỏng quá trình xuất dữ liệu

Dữ liệu ngõ ra là các header và các chuỗi bit đã được sắp xếp. Để kiểm chứng việc sắp xếp các header và chuỗi dữ liệu này, tác giả sử dụng phần mềm Winhex [7] để đọc dữ liệu hex text trong file JPEG2000.



Hình 15: kết quả đọc dữ liệu hex từ file ảnh JPEG2000 bằng phần mềm Winhex

IV.2. Kết quả tổng hợp phần cứng

Sau khi thiết kế và kiểm tra kỹ lưỡng chức năng hoạt động, lõi Tier-2 đã được tổng hợp trên FPGA của Altera bằng phần mềm Quartus II, version 11.1. Sau đây là kết quả tổng hợp:

Thiết bị FPGA	Cyclone II EP2C20F484C6	Cyclone III EP3C16F484C6	Stratix II EP2S15F484C3	Stratix III EP3SE50F484C3
Tần số max (MHz)	87,9	107,91	129,57	157,9
Tài nguyên sử dụng				
+ Registers	199 (1%)	196 (1%)	197 (2%)	197 (<1%)
+ ALUTs (Les)	657 (4%)	544 (4%)	412 (3%)	417 (1%)
+ Memory bits	87040 (36%)	87040 (17%)	87040 (21%)	87040 (2%)

Khối Tier-2 hoạt động với tần số trên 50 MHz, tài nguyên tiêu tốn là rất ít, sử dụng 4% thành phần logic và 1% thanh ghi dành cho dòng chip FPGA giá rẻ (Cyclone) và số lượng tài nguyên thấp nhất. Do đó, khối Tier-2 phù hợp với yêu cầu hoạt động của bộ nén ảnh chuẩn JPEG2000 tốc độ 50 MHz.

V. KẾT LUẬN

Hệ thống nén ảnh chuẩn JPEG2000 với nhiều ưu điểm vượt trội về chất lượng ảnh với hệ số nén ảnh rất cao. Tuy nhiên kèm theo đó, các thuật toán mã hóa sử dụng trong nén ảnh chuẩn JPEG2000 thật sự phức tạp và tiêu tốn nhiều thời gian xử lý. Trong đó, Tier-2 với các giải thuật sắp xếp phức tạp là một trong các khối tiêu tốn nhiều thời gian tính toán nhất, cùng với khối Tier-1 và khối biến đổi wavelet rời rạc. Tuy nhiên, Tier-2 không phải là khối nén chính trong chuẩn JPEG2000 nên các giải thuật này chỉ giúp Tier-2 nén được một lượng nhỏ dữ liệu. Ở đây, tác giả đã tìm hiểu cặn kẽ các kỹ thuật mã hóa được quy định bởi chuẩn nén ảnh JPEG2000 - part 1, nhằm giúp bạn đọc hiểu rõ quy trình sắp xếp và đóng gói dữ liệu nén ảnh trong chuẩn nén ảnh JPEG2000. Việc tối ưu hóa các thuật toán sử dụng trong thiết kế phần cứng chưa được phân tích chi tiết ở đây vì các thuật toán này khá phức tạp. Tác giả sẽ phân tích chi tiết các thuật toán này trong một tài liệu khác.

Việc kiểm tra thiết kế cùng với việc thi hành trên chip FPGA được thực hiện theo đúng quy trình thiết kế vi mạch. Tuy nhiên, thiết kế phần cứng Tier-2 này chỉ áp dụng cho quá trình nén ảnh JPEG2000 với một đến ba thành phần màu (không hỗ trợ ảnh nhiều hơn ba thành phần màu). Ngoài ra, thiết kế sắp xếp dữ liệu thành file JPEG2000 theo chuẩn ".j2k", chưa sắp xếp theo chuẩn ".jp2", một chuẩn sắp xếp dữ liệu khác được ghi trong JPEG2000 - part 2 ■

TÀI LIỆU THAM KHẢO

[1] ISO/IEC 15444-1:2000, "Information technology - JPEG2000 image coding system - Part 1: Core coding system", 2000.

[2] ITU - T recommendation T.800, *Information technology - JPEG2000 image coding system: Core coding system*, 8.2002.

[3] David S. Taubman, *JPEG2000 Image Compression Fundamentals, Standards and Practice*, 2002.

[4] Joint Photographic Experts Group, *Information Technology - JPEG2000 image coding system: Core coding system*, bản sửa thứ 2, tháng 9.2004.

[5] Diego Stanta-Cruz, Raphael Grosbois, and Touradj Ebrahimi, *JPEG2000 performance evaluation and assessment. Signal Processing: Image communication*, 17(1):113-130, 2002.

[6] OpenJPEG - http://www.openjpeg.org/

[7] Winhex software - http://winhex.com/winhex/index-f.html

[8] Kakadu software - http://www.kakadusoftware.com/

[9] Mentor Graphics, ModeilSim SE Reference Manual, Version 6.5