VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY
UNIVERSITY OF SCIENCE, HO CHI MINH CITY
FACULTY OF ELECTRONICS & TELECOMMUNICATIONS
COMPUTER & EMBEDDED SYSTEM LABORATORY

JPEG Encoder Specification

Summary:

Version: 1.0	**Release Date**: Month: July Year: 2011	**Total Size**: ☑ Reports ☑ References ☑ Simulation	☑ Images ☑ Source Codes
Solution Type:	☑ IP Core ☐ Megafunction ☐ SoPC Builder Component ☐ Qsys Component ☐ System Design	**External Devices**:	☐ Lcd TRDB-LCM
Supported Device(s)/ Speed (MHz)/ LEs:	☐ Stratix IV .....MHz .....LEs ☑ Stratix III .....MHz .....LEs ☐ Stratix II .....MHz .....LEs ☐ Cyclone III .....MHz .....LEs ☐ Cyclone II .....MHz .....LEs		
Description:	JPEG Encoder Core (verilog HDL). The Core is compatible with Burst Transfer.		

1 THUẬT TOÁN NÉN ẢNH CHUẨN JPEG

1.1 Tổng Quan

Trước khi đi vào thuật toán nén ảnh, việc hiểu một số khái niệm quan trọng cũng như những thành phần nhỏ hơn được sử dụng trong thuật toán là thật sự cần thiết. Vì thế phần này sẽ dành trình bày về những định nghĩa quan trọng cũng như những thành phần cốt lõi được sử dụng trong thuật toán nén ảnh theo chuẩn JPEG.

1.1.1 Hệ màu (Color Spaces)

Đa số các ứng dụng video đều có phần hiển thị và trình chiếu video màu, vậy nên cần thiết phải có cơ chế để thu nhận và biểu diễn các thông tin về màu sắc. Trong khi ảnh xám chỉ cần một con số để chỉ độ sáng (brightness) của bức ảnh, thì ảnh màu cần ít nhất 3 thành phần trong mỗi pixel để diễn tả chính xác màu sắc của nó. Các cách thức dùng để biểu diễn độ sáng (luminance hay brightness) và màu sắc của bức ảnh được gọi là hệ màu. Phần sau đề cập tới 2 hệ màu cần thiết cho core nén ảnh JPEG: RGB và YCbCr

Hệ màu RGB dùng 3 con số để biểu diễn 3 thành phần màu: đỏ, xanh lam và xanh lục – cũng là 3 màu cơ bản dùng để tạo ra các màu khác nhau. Hệ màu RGB phù hợp cho việc thu nhận và hiển thị hình ảnh

Hệ thống thị giác của con người (HVS) tinh nhạy với độ sáng (brightness) hơn là màu sắc (color). Trong hệ màu RGB, 3 thành phần màu có vai trò như nhau nên thường được lưu giữ với cùng một độ phân giải. Tuy nhiên, với đặc điểm kể trên của HVS, việc biểu diễn hình ảnh có thể được cải tiến bằng cách tách riêng thông tin về độ sáng khỏi thông tin về màu sắc, và sau đó biểu diễn thành phần độ sáng này ở độ phân giải cao hơn.

Hệ màu YCbCr và các biến thể của nó (YUV) nếu tính đầy đủ thì bao gồm 4 thành phần: Y (luminance component) biểu diễn độ sáng, và 3 thành phần Cb, Cr, Cg (Chrominance) dùng để chỉ độ sai biệt giữa thành phần màu xanh lam, đỏ, xanh lục với độ sáng trung bình của mỗi bức ảnh. Tuy nhiên, vì tổng của 3 thành phần Cb, Cr và Cb là 1 hằng số nên việc thu nhận và truyền tín hiệu chỉ sử dụng 2 thành phần Cb và Cr, còn Cg có thể được suy ra từ 2 thành phần kia

Công thức chuyển đổi từ hệ màu RGB sang YCbCR:

$$
\begin{aligned}
Y &= 0.299R + 0.587G + 0.114B \\
Cb &= -0.16874R - 0.33126G + 0.50000B + 0.5 \\
Cr &= 0.50000R - 0.41869G - 0.08131B + 0.5
\end{aligned}
\qquad (1)
$$

1.1.2 Lấy mẫu (YCbCr Sampling Formats)

Hình 1.1 trình bày 3 cách giảm mẫu: 4:4:4 Sampling, 4:2:2 Sampling và 4:2:0 Sampling.

Cách giảm mẫu 4:4:4 có nghĩa là 3 thành phần (Y, Cb, Cr) sẽ có cùng độ phân giải và do đó, tại mỗi pixel đều tồn tại đầy đủ cả 3 thành phần trên. Ba con số 4 (4:4:4) ám chỉ cứ mỗi 4 mẫu Luminance sẽ có 4 mẫu Cb và 4 mẫu Cr

Trong cách giảm mẫu 4:2:2 (còn gọi là YUV2), các mẫu Chrominance (Cb và Cr) sẽ có độ phân giải về chiều dọc bằng với các mẫu Luma (Y), nhưng chỉ có độ phân giải bằng phân nửa các mẫu Luma theo chiều ngang.

Giảm mẫu 4:2:0 nghĩa là thành phần Cb và Cr chỉ có độ phân giải bằng phân nửa so với thành phần Y tính theo cả 2 chiều. Con số 4:2:0 ở đây chỉ có nghĩa về lịch sử chứ không thể hiện được nghĩa của cách lấy mẫu này. Độ phân giải của video 4:2:0 bằng đúng phân nửa độ phân giải của video 4:4:4

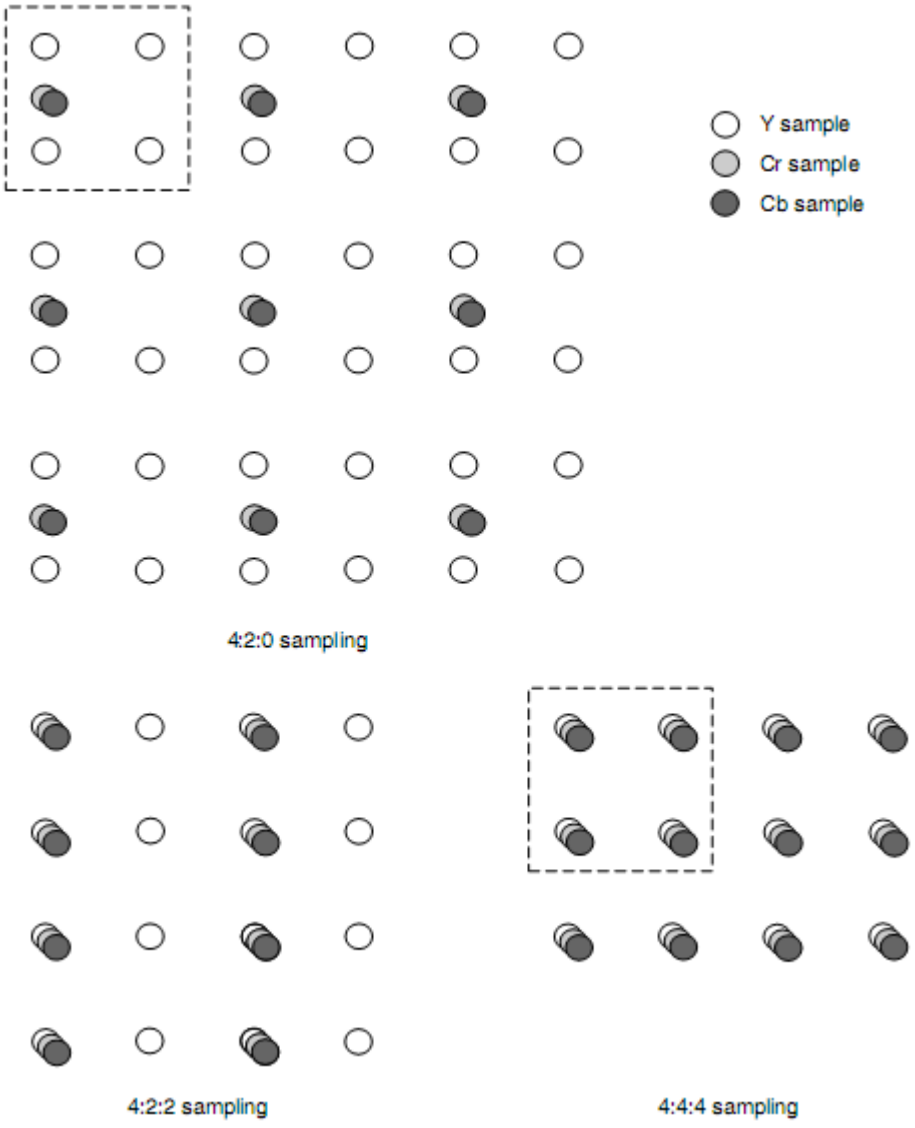

Figure 1. Cách giảm mẫu kiểu 4:4:4, 4:2:2 và 4:2:0

1.1.3 Biến đổi Cosine rời rạc (Discrete Cosine Transform)

Phép biến đổi Cosine rời rạc (DCT) được dùng để biến đổi tín hiệu từ miền thời gian sang miền tần số. DCT được sử dụng rộng rãi trong các hệ thống nén ảnh, video, âm thanh…

Hình ảnh sau khi được chuyển sang hệ màu YCbCr sẽ được biến đổi thành các ma trận 64 hệ số thể hiện tần số các pixels. Hệ số đầu tiên được gọi là thành phần DC, thể hiện đặc trưng cao nhất, các hệ số sau là thành phần AC và có giá trị giảm dần. Các thành phần AC càng cách xa DC thì càng ít quan trọng và có thể được lược bỏ, nhờ vậy mà ảnh được nén lại. Công thức biến đổi DCT và đảo DCT (IDCT) được biểu diễn như hình sau:

$$\text{FDCT: } X(m) = \frac{s(m)}{2}\sum_{n=0}^{N-1} x(n)\cos\left(\frac{(2n+1)m\pi}{2N}\right) \quad (2)$$

$$\text{IDCT: } x(n) = \sum_{n=0}^{N-1} \frac{s(m)}{2} X(m)\cos\left(\frac{(2n+1)m\pi}{2N}\right)$$

Trong đó,

x(n): ngõ vào miền thời gian

X(m): hệ số DCT ở miền tần số

Tuy nhiên, vì phép biến đổi DCT quá phức tạp và không thể thực hiện được trên phần cứng, hàng loạt các phương pháp biến đổi nhanh DCT đã ra đời nhằm giảm thiểu số lượng phép tính cần thiết, và qua đó giúp việc triển khai thuật toán này trên phần cứng trở nên khả thi hơn. Core nén ảnh JPEG trong báo cáo này sử dụng thuật toán BinDCT, giúp tính các hệ số DCT gần đúng chỉ bằng các phép cộng và dịch bit

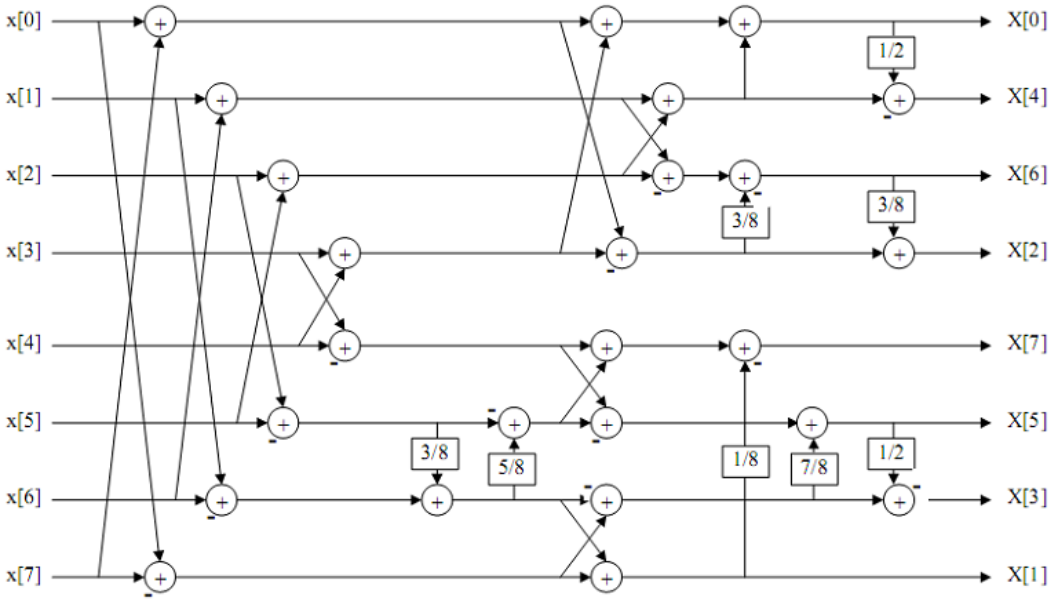

Figure 2. Mô hình Butterfly của thuật toán BinDCT

1.1.4 Lượng tử hóa

Vì mắt người dễ nhận ra những sai biệt ở các tín hiệu tần số thấp, trong khi lại gần như không thể phân biệt được sự thay đổi ở các tín hiệu tần số cao, nên lượng tử hóa thực hiện việc giảm thông tin của các thành phần tần số cao bằng cách chia mỗi phần tử trong miền tần số (các hệ số DCT) với hằng số tương ứng với phần tử đó, rồi lấy kết quả làm tròn tới số nguyên gần nhất. Vì các hệ số DCT có dạng những ma trận 8x8, nên các hằng số tương ứng với nó cũng dưới dạng ma trận 8x8, và còn được gọi là ma trận lượng tử. Quá trình này sẽ giảm giá trị của các thành phần tần số cao về 0, còn những thành phần còn lại cũng được giảm xuống nhỏ hơn để có thể được biểu diễn với một số lượng bit ít hơn, qua đó giúp giảm dung lượng lưu trữ thông tin.

Lượng tử hóa là bước quan trọng trong nén ảnh JPEG, thông qua quá trình này, ta có thể cấu hình tỉ lệ nén khác nhau tùy vào mục đích sử dụng. Chất lượng được quy ước trong dải từ 1 – 100, trong đó mức độ 1 cho tỉ lệ nén cao nhất nhưng chất lượng ảnh nén tệ nhất, còn 100 cho ảnh nén chất lượng cao nhưng tỉ lệ nén không hiệu quả. Những nghiên cứu liên quan tới thị giác con người đã giúp tìm ra ma trận lượng tử tiêu chuẩn (mức độ 50), cung cấp ảnh nén chất lượng chấp nhận được với tỉ lệ nén vừa phải

Ma trận lượng tử điển hình (Q=50) trong tiêu chuẩn JPEG như sau:

$$Q = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}. \quad (3)$$

1.1.5 Mã hóa Huffman

Huffman là thuật toán dùng để nén và khôi phục dữ liệu mà không làm mất thông tin. Nguyên tắc của thuật toán là dựa vào tần suất xuất hiện của các kí tự cần mã hóa để tạo ra một bộ mã nhị phân cho các kí tự đó sao cho số bits sau khi mã hóa là nhỏ nhất

1.2 Quy Trình Nén Ảnh chuẩn JPEG

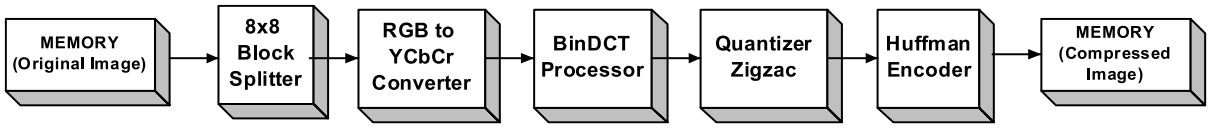

Figure 1. Quy trình thuật toán nén ảnh chuẩn JPEG

Ảnh gốc được lấy từ bộ nhớ (hoặc từ camera) và được chuyển vào khối xử lý tách block 8x8. Khối này có nhiệm vụ tách dữ liệu đưa vào thành các ma trận 8x8

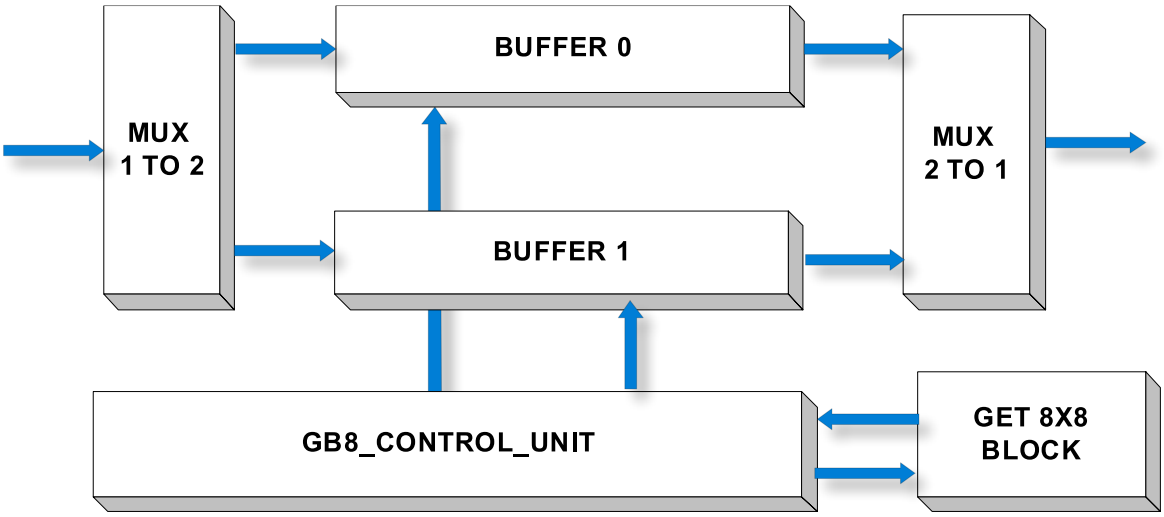

Figure 2. Tách dữ liệu thành từng block 8x8

Nguyên tắc hoạt động của khối tách block 8x8 là dùng bộ nhớ đệm (buffer) để lưu giữ 8 dòng của bức ảnh. Sau đó đọc chúng ra theo thứ tự mong muốn để thu được các ma trận có kích thước 8x8 pixels. Thứ tự đọc ra phụ thuộc vào khối GET8x8BLOCK, khối này có nhiệm vụ đưa địa chỉ của dữ liệu cần đọc trong buffer cho khối Control Unit để điều khiển quá trình lấy dữ liệu.

Nhận thấy hệ thống buộc phải chờ trong thời gian lấy dữ liệu ra (lúc đang lấy dữ liệu ra thì dữ liệu của 8 dòng kế tiếp không vào được), nên thay vì dùng 1 bộ nhớ đệm, hệ thống sử dụng 2 bộ nhớ đệm thay phiên nhau lưu trữ dữ liệu. Khi 8 dòng đầu tiên được đọc vào buffer 0, buffer 1 ở trạng thái không hoạt động, tới khi 8 dòng đầu tiên được đọc ra khỏi buffer 0 thì buffer 1 sẽ tiếp nhận 8 dòng kế tiếp của bức ảnh. Hai bộ MUX được sử dụng để điều khiển dữ liệu vào ra các buffers. Kĩ thuật sử dụng 2 khối xử lý song song như thế này còn gọi là kĩ thuật ping-pong. Khối Control Unit dùng để kiểm soát hoạt động của toàn khối tách block 8x8.

Dữ liệu sau khi được tách thành từng block có kích thước 8x8 sẽ đi qua quá trình đổi hệ màu từ RGB thành YCbCr theo công thức 1 ở phần 1.1.1

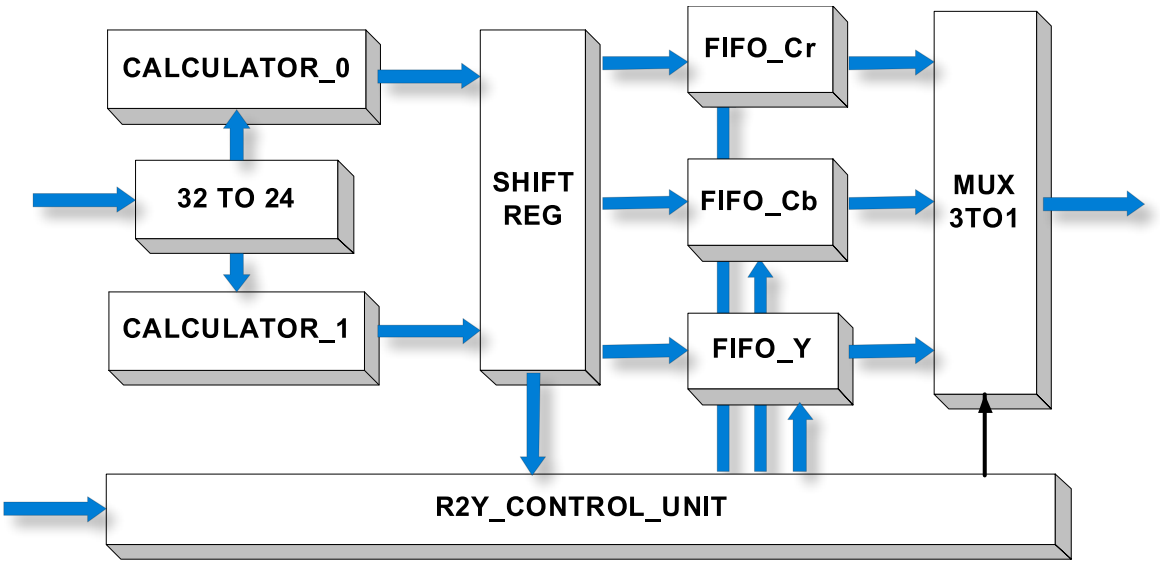

Figure 3. Khối đổi hệ màu từ RGB sang YCbCr và giảm mẫu

Đầu tiên, dữ liệu (có độ rộng 32 bits) sẽ được tách ra thành từng pixel, mỗi pixel RGB được biểu diễn bởi 24 bits. Nếu dữ liệu đưa vào là ảnh bitmap 24-bit thì cứ 4 lần dữ liệu đi vào sẽ mang thông tin của 3 pixels, còn trong trường hợp ảnh đầu vào là bitmap 16-bit thì cứ 1 lần dữ liệu đi vào sẽ mang thông tin của 2 pixels.

Các pixels (24 bits) sau khi được tách ra sẽ được biến đổi sang hệ màu YCbCr thông qua khối Calculator. Vì mỗi một lần dữ liệu 32 bits đi vào luôn mang thông tin của nhiều hơn 1 pixel, nên cùng một lúc có thể sẽ có hơn 1 pixel được tách ra, và trong trường hợp đó thì chỉ 1 khối Calculator xử lý có thể làm hệ thống phải chờ (vì 1 khối Calculator một lúc chỉ xử lý được 1 pixel). Do đó, hai khối Calculators được luân phiên sử dụng để đảm bảo xử lý kịp dữ liệu.

Khối Calculator sẽ cho ra kết quả là giá trị của các thành phần Y, Cb và Cr của bức ảnh trong hệ màu YCbCr, những giá trị của mỗi thành phần trên được lưu vào các bộ đệm (buffer) tương ứng. Khối điều khiển Control Unit sẽ điều khiển toàn bộ hoạt động, đồng thời tiến hành giảm mẫu dữ liệu bằng cách lược bỏ các thành phần Cb, Cr tùy theo lựa chọn giảm mẫu là 4:2:2 hay 4:4:4

Dữ liệu đã được biến đổi sang hệ màu YCbCr và giảm mẫu sẽ sẵn sàng để qua bước biến đổi DCT.

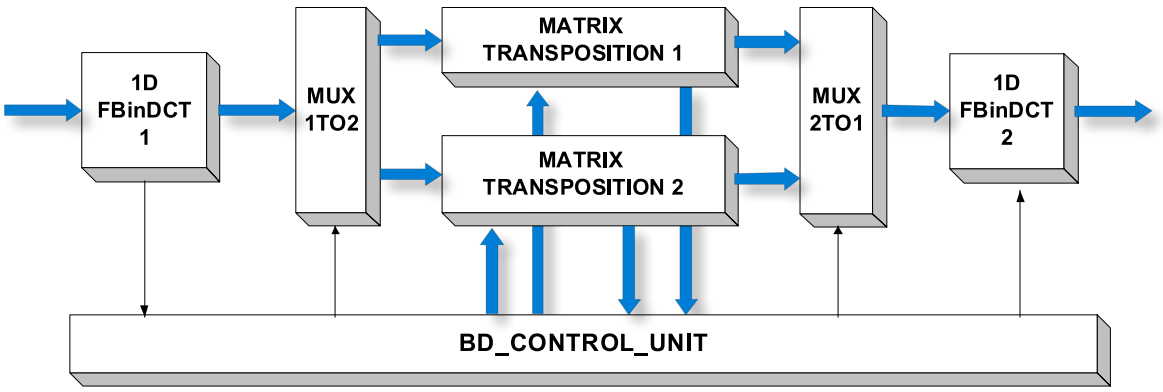

Figure 4. Khối biến đổi DCT dùng thuật toán BinDCT trên phần cứng

Mỗi ma trận 8x8 dữ liệu đầu vào sẽ được biến đổi DCT 2 lần, lần đầu biến đổi DCT trên từng dòng ma trận (ngõ vào 8 hệ số), sau tiếp tục biến đổi DCT trên từng cột của ma trận để thu được kết quả sau cùng

Vì ở khối trước (khối đổi hệ màu và giảm mẫu), dữ liệu đưa ra đã là dưới dạng từng dòng một, nên mỗi dòng dữ liệu này (8 phần tử) sẽ đi qua khối BinDCT thứ nhất để thực hiện biến đổi DCT trên dòng, sau đó, kết quả sẽ qua khối Matrix Transposition để thực hiện chuyển vị ma trận, bước này dùng để chuẩn bị cho biến đổi DCT lần 2 trên cột.

Tương tự như những phần trên, kĩ thuật ping-pong được sử dụng với 2 khối Matrix Transposition luân phiên thực hiện và các bộ Multiplexer để điều khiển luồng dữ liệu

Các hệ số DCT là các thành phần tín hiệu ở miền tần số. Các hệ số này sẽ qua bước lượng tử hóa để chuẩn bị cho mã hóa Huffman.

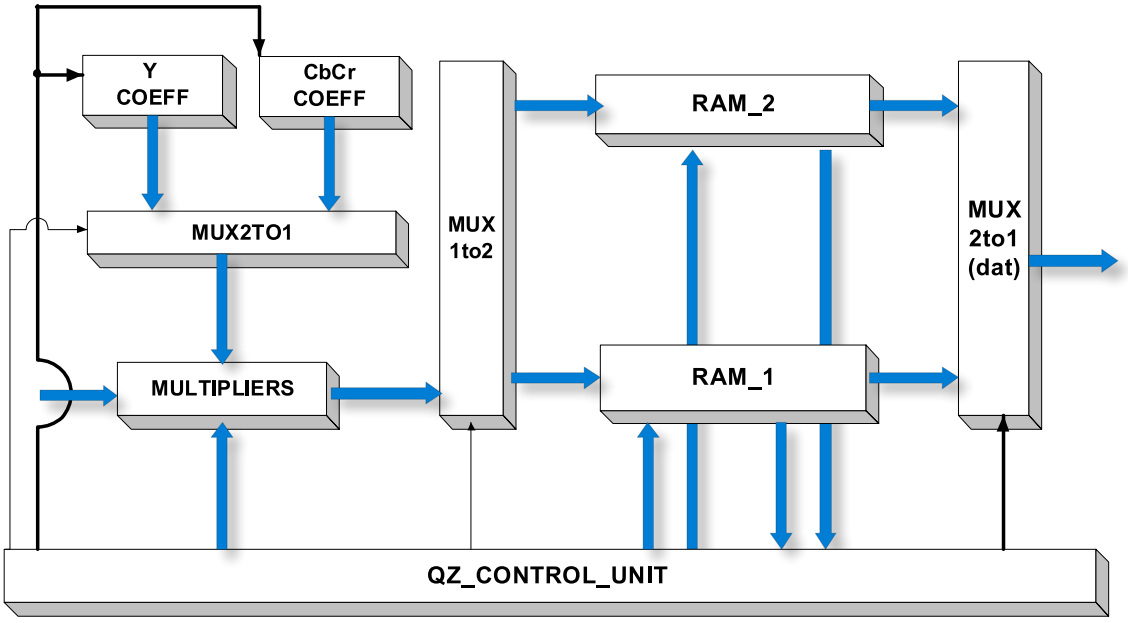

Figure 7. Sơ đồ thiết kế trên phần cứng của khối lượng tử hóa

Khối lượng tử hóa sẽ xử lý từng dòng dữ liệu một, hay nói cách khác, nó sẽ xử lý 8 hệ số của 1 dòng cùng một lúc. Khối Multiplers trong hình 7 bao gồm 8 khối nhân nhỏ, dùng để thực hiện phép chia 8 hệ số với 8 hằng số tương ứng của ma trận lượng tử. Vì phép chia cũng chính là phép nhân nghịch đảo nên các hằng số lượng tử sẽ được nghịch đảo và lưu trữ cố định trong ROM. Ta có 2 bảng lượng tử được sử dụng, 1 dành cho thành phần Y (YCoeff), và cái còn lại dành cho Cb và Cr (CbCrCoeff).

Kết quả sau khi lượng tử hóa sẽ được chuyển vào RAM để tiến hành sắp xếp Zigzac. Mục đích của việc sắp xếp này nhằm chuẩn bị cho mã hóa Huffman. Mô hình sắp xếp được minh họa như hình sau:

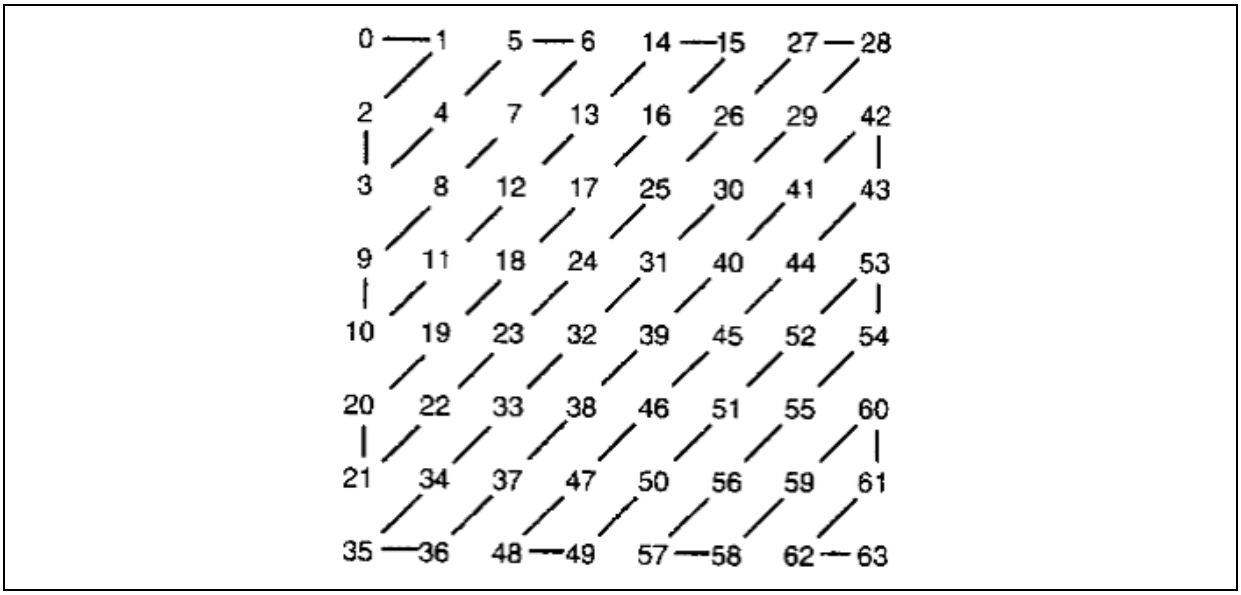

Figure 8. Thứ tự Zigzac

Bộ nhớ RAM dùng để lưu trữ 1 block 8x8, và 2 bộ nhớ RAM như vậy được sử dụng để luân phiên lưu trữ dữ liệu một cách liên tục (ping-pong). Dữ liệu của 1 block sau đó sẽ được đọc ra theo thứ tự Zigzac và được chuyển tới khối mã hóa Huffman

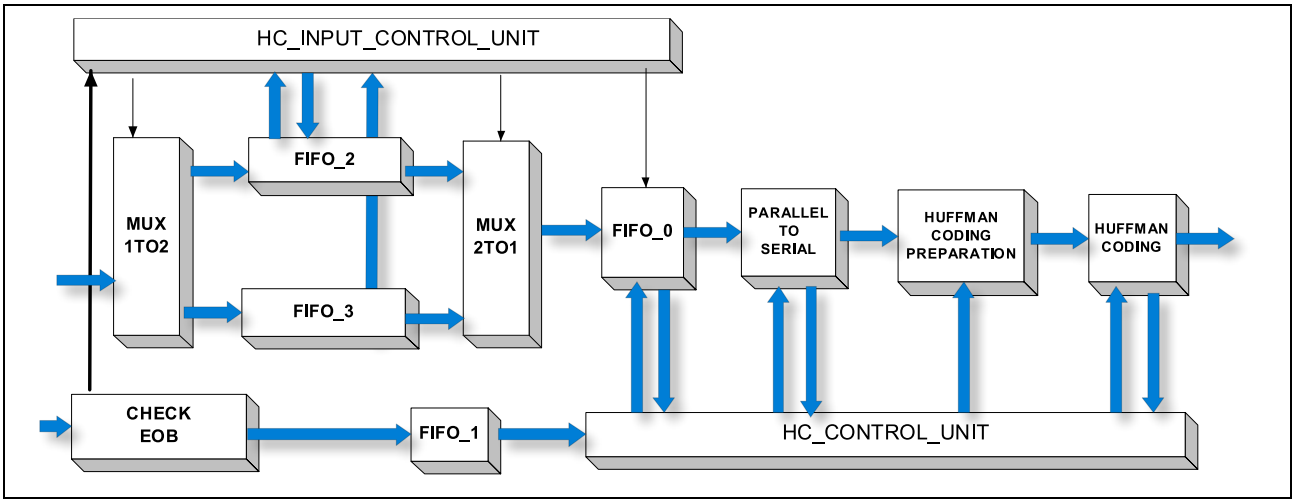

Figure 9. Khối mã hóa Huffman

Khối Huffman bao gồm 2 phần: phần chuẩn bị và phần mã hóa

Phần chuẩn bị có nhiệm vụ tìm ra vị trí dữ liệu của dòng chứa giá trị khác 0 cuối cùng trong block 8x8. Những vị trí này sẽ được lưu vào FIFO_1

FIFO_2 và FIFO_3 chỉ đơn giản là dùng làm bộ đệm để chuyển dữ liệu **khác 0** từ ngõ vào tới FIFO_0

Khối Parallel To Serial sẽ chuyển đổi từng dòng dữ liệu 8 phần tử thành từng phần tử riêng biệt

Hai khối Huffman Preparation và Huffman Coding thực hiện việc mã hóa các giá trị theo thuật toán Huffman

Khối HC_Control_Unit điều khiển toàn bộ hoạt động của khối mã hóa, còn Input_Control_Unit dùng để điều khiển phần chuẩn bị của khối mã hóa

2 CHI TIẾT THIẾT KẾ

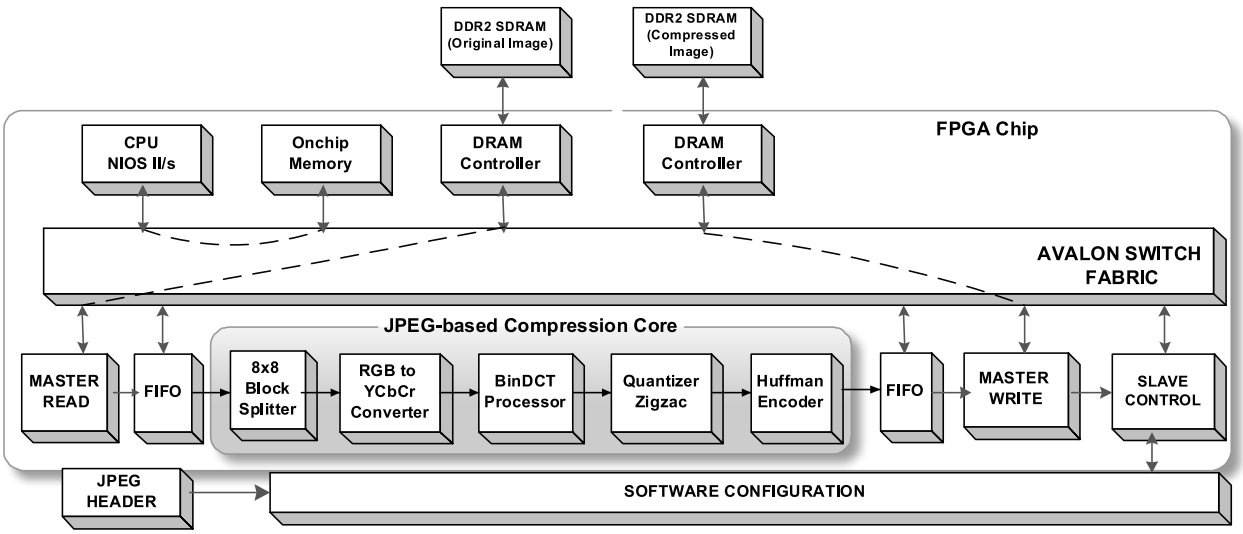

Figure 10. Sơ đồ hệ thống SoPC thực hiện nén ảnh JPEG

Hệ thống nén ảnh bao gồm 2 phần:

- Phần nén ảnh JPEG (JPEG-based Compression Core): nhận dữ liệu từ Master Read của bộ DMA (Direct Memory Access) để nén ảnh rồi ghi ảnh nén lên RAM thông qua Master Write
- Phần còn lại: bao gồm DMA dùng để chuyển dữ liệu giữa Core nén và RAM và các phần giao tiếp với Avalon Bus

1.3 PHẦN GIAO TIẾP VỚI AVALON SWITCH FABRIC

Bao gồm Slave Control, các Master đọc/ghi, các bộ nhớ RAM để lưu dữ liệu ảnh **Error! Reference source not found.**. Để tăng tốc độ truy xuất, các Master được thiết kế để phù hợp với kiểu truyền Burst.

1.3.1 Slave Controller

Với nhiệm vụ nhận các cấu hình từ CPU**Error! Reference source not found.** đưa xuống để cung cấp thông tin cho khối nén ảnh hoạt động, Slave Control chứa trong nó nhiều thanh ghi nhằm lưu giữ các giá trị cấu hình cũng như các kết quả sau khi thực thi hoàn tất.

số	Tên thanh ghi	Độ rộng	Ghi/ Đọc	Mô tả
0	control	32	Chỉ ghi	Tập hợp các tín hiệu điều khiển cho các khối: ✓ control[31:7]: Reserved ✓ control[6]: cấu hình giảm mẫu 4:4:4 ✓ control[5]: cấu hình giảm mẫu 4:2:2 ✓ control[4]: Reserved ✓ control[3]: điều khiển tín hiệu GO ✓ control[2]: cho biết đô rộng dữ liệu là 1 word (32 bits) ✓ control[1]: cho biết đô rộng dữ liệu là 1 halfword (16 bits) ✓ control[0]: cho biết đô rộng dữ liệu là 1 byte (8 bits)
1	src_address	32	Chỉ ghi	Source Address: địa chỉ đầu tiên của vùng nhớ chứa dữ liệu ảnh gốc
2	dest_address	32	Chỉ ghi	Destination Address: địa chỉ đầu tiên của vùng nhớ chứa dữ liệu ảnh nén
8	X_image_mul_Y_image	32	Chỉ ghi	Khổ ảnh (tính theo pixel) = chiều cao x chiều rộng.
21	image_dimensions	32	Chỉ ghi	16 bit thấp: chiều rộng của ảnh (pixel) 16 bit cao: chiều cao của ảnh (pixel)
26	MW_lastaddress	32	Chỉ đọc	Master Write Last Address: Địa chỉ của ô nhớ chứa dữ liệu ảnh nén cuối cùng được Master Write ghi lên RAM
27	global_timer	32	Chỉ đọc	Cho biết số chu kỳ xung clock cần thiết cho 1 lần nén ảnh (kể cả thời gian đọc và ghi dữ liệu của bộ DMA)
29	Status	32	Chỉ đọc	Trạng thái hoạt động hiện tại của khối nén ảnh JPEG ➢ status [31:2]: Reserved ➢ status [1]: khối nén ảnh đang hoạt động

				➢ status [0]: việc nén ảnh đã hoàn tất

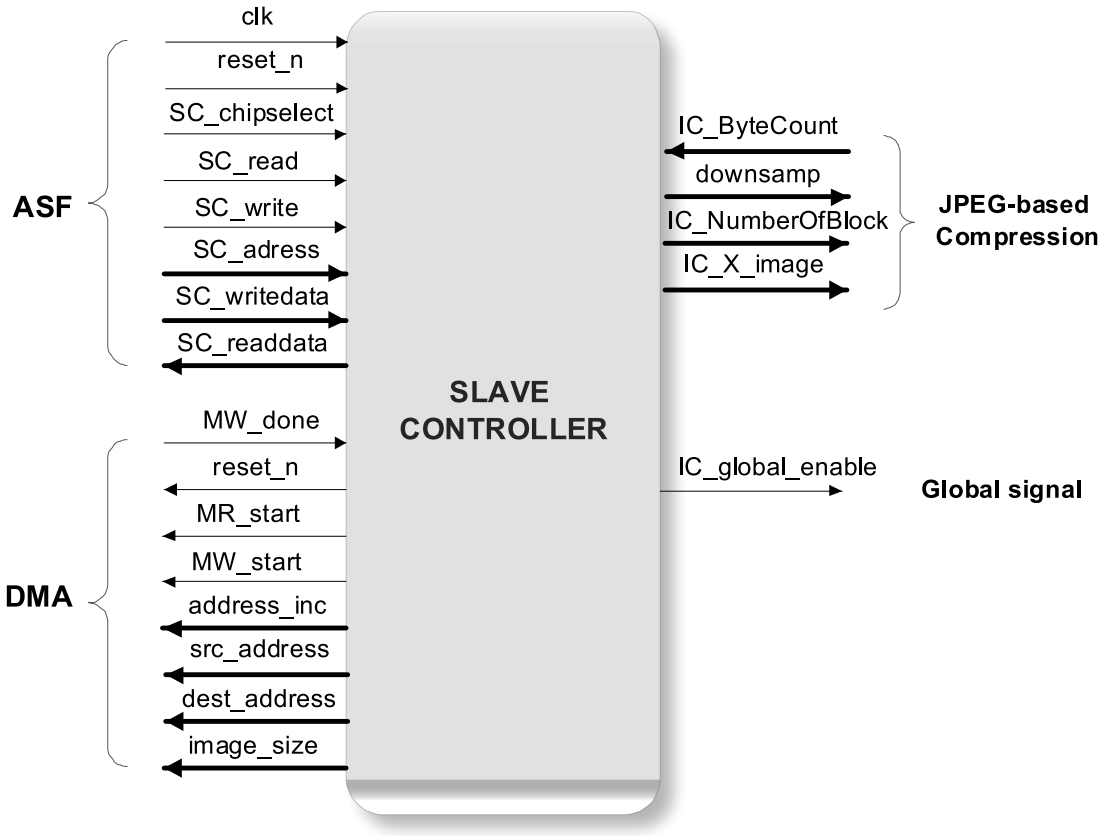

Figure 11. Các giao tiếp I/O của Slave Controller

Tín hiệu	Độ rộng	Hướng	Mô tả
Clk	1	Input	Tín hiệu xung nhịp giúp toàn bộ khối hoạt động đồng bộ với nhau.
Reset_n	1	Input	Tín hiệu Reset đồng bộ tác động mức thấp.
SC_chipselect	1	Input	Tín hiệu điều khiển từ ASF**Error! Reference source not found.**, được sinh ra từ bộ giải mã địa chỉ giúp chọn thiết bị hoạt động. Đây là tín hiệu tiên quyết phải được tích cực trong quá trình ghi hoặc đọc của Slave Control.
SC_read	1	Input	Tín hiệu điều khiển từ ASF, được tích cực trong suốt quá trình CPU yêu cầu đọc dữ liệu từ Slave Control.
SC_write	1	Input	Tín hiệu điều khiển từ ASF, được tích cực trong suốt quá trình CPU yêu cầu ghi dữ liệu từ Slave Control.
SC_address	3	Input	Tín hiệu địa chỉ từ ASF, xác định vị trí offset của thanh ghi cần truy xuất nằm

			trong Slave Control.
SC_writedata	32	Input	Dữ liệu từ ASF, dữ liệu cần ghi vào Slave Control, được đi kèm với tín hiệu SC_write và địa chỉ SC_address.
MW_done	1	Input	Tín hiệu từ khối Master Write báo về Slave Control khi hoàn tất hoạt động nén ảnh và ghi dữ liệu ảnh nén lên DDR2 DRAM
IC_ByteCount	32	Input	Kết quả trả về từ khối Huffman Encoder, cho biết số byte của ảnh đã nén, tham số này được dùng để tính ra MW_lastaddress
IC_global_enable	1	Output	Tín hiệu enable hệ thống
image_size	32	Output	Kích thước ảnh gốc RGB tính theo byte, được truyền cho Master Read để biết kích thước dữ liệu cần đọc
address_inc	3	Output	(Address Increment) giá trị này được truyền cho Master Write để đọc dữ liệu từ FIFO và ghi lên RAM, nó cho biết độ rộng đường dữ liệu để từ đó tăng địa chỉ sau mỗi lần truy xuất cho phù hợp và phụ thuộc vào thanh ghi control[2:0]
downsamp	3	Output	Truyền giá trị này cho phần nén JPEG để thực hiện kiểu giảm mẫu mà user cấu hình trong thanh ghi control[6:4]
IC_NumberOfBlock	20	Output	Giá trị cho biết số block 8x8 (tính theo byte) của ảnh gốc. Giá trị này được truyền cho core nén JPEG
IC_X_image	16	Output	Giá trị cho biết độ rộng của ảnh gốc (tính theo pixel) và được truyền cho phần nén JPEG
SC_readdata	32	Output	Dữ liệu gửi đến ASF, dữ liệu này đáp ứng cho yêu cầu đọc từ CPU.

1.3.2 Master Read Burst

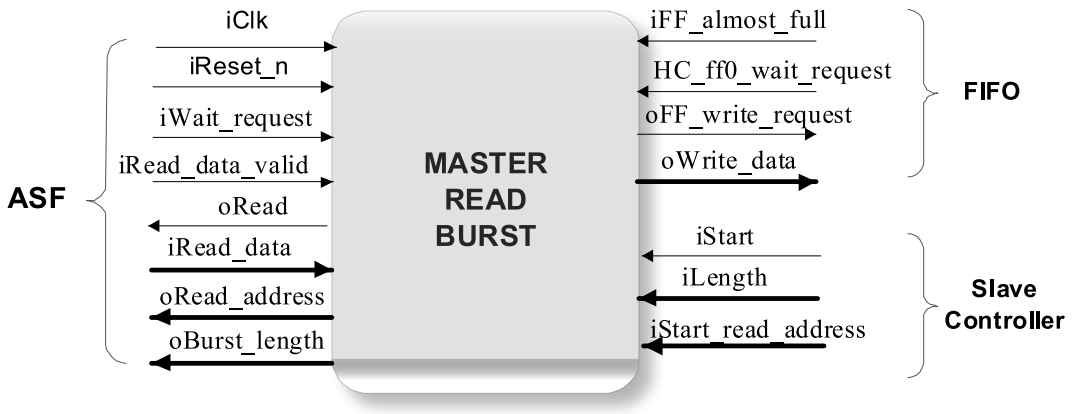

Figure 12. Master Read Burst trong bộ DMA

Master Read Burst sử dụng kiểu truyền Burst để thực hiện quá trình đọc. Với burst length là 64, Master Read Burst cho phép tăng tốc độ đọc lên đáng kể so với kiểu đọc thông thường. Đối với quá trình đọc thông thường, sau mỗi dữ liệu đọc được thì ASF phải gửi tiếp tục các thông tin cần thiết về địa chỉ và bật tín hiệu yêu cầu đọc để quá trình đọc được thực hiện. Quá trình đọc Burst được mô tả trong hình sau.

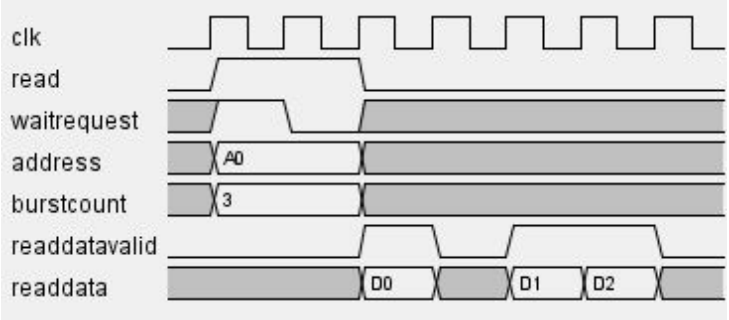

Figure 135. Dạng sóng của quá trình đọc burst

Master Read đọc dữ liệu liên tục burstcount lần và sau đó nếu muốn tiếp tục đọc thì sẽ gửi địa chỉ và tích cực tín hiệu read trong 1 chu kỳ xung clock duy nhất để yêu đọc dữ liệu. Dữ liệu được gói gọn trong tín hiệu readdatavalid nhằm cho biết dữ liệu hợp lệ để đọc vào. Với kiểu đọc này, quá trình đọc tiết kiệm khá nhiều thời gian cho việc gửi các tín hiệu điều khiển đến ASF.

Tín hiệu	Độ rộng	Hướng	Mô tả
iClk	1	Input	Tín hiệu xung nhịp giúp toàn bộ khối hoạt động đồng bộ với nhau.
iReset_n	1	Input	Tín hiệu Reset đồng bộ tác động mức thấp.
iStart	1	Input	Tín hiệu từ Slave Control báo hiệu khởi động Master Read Burst.
iRead_data_valid	1	Input	Tín hiệu từ ASF thông báo dữ liệu hợp lệ để Master đọc vào.
iWait_request	1	Input	Tín hiệu từ ASF thông báo ASF đang bận và yêu cầu chờ.
iFF_almost_full	1	Input	Tín hiệu từ FIFO thông báo dữ liệu trong FIFO đã gần đầy.
HC_ff0_wait_request	1	Input	Tín hiệu gửi đến FIFO nhằm yêu cầu ghi dữ liệu vào FIFO.
iStart_read_address	32	Input	Thông tin từ Slave Control chứa thông tin về địa chỉ vùng nhớ đầu tiên bắt đầu đọc.
iLength	32	Input	Thông tin từ Slave Control chứa thông tin về độ lớn tính theo byte của vùng nhớ cần đọc.
iRead_data	32	Input	Dữ liệu từ ASF gửi đến Master trong quá

			trình đọc.
oWrite_data	32	Output	Dữ liệu gửi đến FIFO, kèm theo tín hiệu oFF_write_request sẽ ghi dữ liệu vào FIFO.
oRead	1	Output	Tín hiệu gửi đến ASF để yêu cầu đọc dữ liệu.
oRead_address	32	Output	Tín hiệu địa chỉ gửi đến ASF cho biết thông tin vùng nhớ cần đọc.
oFF_write_request	1	Output	Tín hiệu yêu cầu ghi vào FIFO dữ liệu mà Master Read đọc được
oBurst_length	8	Output	Thông tin gửi đến ASF cho biết thông tin về độ dài quá trình burst. Tương ứng với tín hiệu BurstCount trên ASF.

1.3.3 Master Write

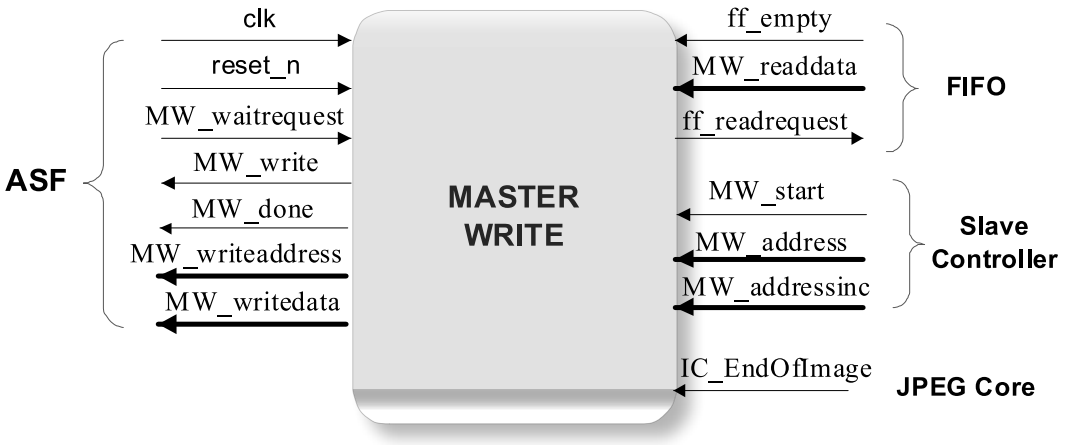

Figure 14. Master Write trong bộ DMA

Master Write sử dụng kiểu truyền Pipeline. Quá trình ghi một dữ liệu sẽ gồm các bước: tích cực tín hiệu write, gửi dữ liệu và địa chỉ cần ghi.

Đối với kiểu ghi pipeline, địa chỉ của mỗi ô nhớ cần ghi lên được đưa ra liên tục trong mỗi chu kì xung clock

Tín hiệu	Độ rộng	Hướng	Mô tả
clk	1	Input	Tín hiệu xung nhịp giúp toàn bộ khối hoạt động đồng bộ với nhau.
reset_n	1	Input	Tín hiệu Reset đồng bộ tác động mức thấp.
MW_start	1	Input	Tín hiệu từ Slave Control báo hiệu khởi động Master Write.
MW_address	32	Input	Thông tin từ Slave Control chứa thông tin về địa chỉ vùng nhớ **đầu tiên** bắt đầu ghi.
MW_addressinc	3	Input	(Address Increment) Thông tin từ tín hiệu address_inc của Slave Controller truyền

			cho Master Write
ff_empty	1	Input	Tín hiệu từ FIFO thông báo dữ liệu trong FIFO đã gần hết.
MW_waitrequest	1	Input	Tín hiệu từ ASF thông báo ASF đang bận và yêu cầu chờ.
IC_EndOfImage	1	Input	Tín hiệu từ phần nén ảnh JPEG gởi tới Master Write cho biết đã hoàn tất quá trình nén
MW_readdata	32	Input	Dữ liệu từ FIFO đáp ứng cho yêu cầu đọc từ Master Write.
MW_done	1	Output	Tín hiệu gửi đến ASF thông báo hoàn tất quá trình ghi của Master Write.
ff_readrequest	1	Output	Tín hiệu gửi đến FIFO nhằm yêu cầu đọc dữ liệu từ FIFO.
MW_writedata	32	Output	Dữ liệu gửi đến ASF trong quá trình ghi.
MW_write	1	Output	Tín hiệu gửi đến ASF để yêu cầu ghi dữ liệu.
MW_writeaddress	32	Output	Tín hiệu địa chỉ gửi đến ASF cho biết thông tin vùng nhớ cần ghi.

1.3.4 FIFO

Figure 6. Ví dụ về một FIFO sử dụng trong thiết kế

Thiết kế sử dụng nhiều FIFO như là bộ nhớ đệm giúp cho việc chuyển dữ liệu trở nên nhanh chóng và đáng tin cậy hơn. Phần mềm Quartus II của Altera hỗ trợ công cụ MegaWizard Plug-In Manager cho phép sử dụng các thiết kế có sẵn của Altera. FIFO được lấy ra từ công cụ này. Quá trình chọn lựa và cấu hình thông số được thực hiện qua các bước sau:

- Trong Quartus II chọn Tools/MegaWizard Plug-In Manager…
- Chọn “Create a new custom megafunction variation” để tạo mới một thành phần nào đó.

- Chọn FIFO và đặt tên file verilog muốn xuất ra.

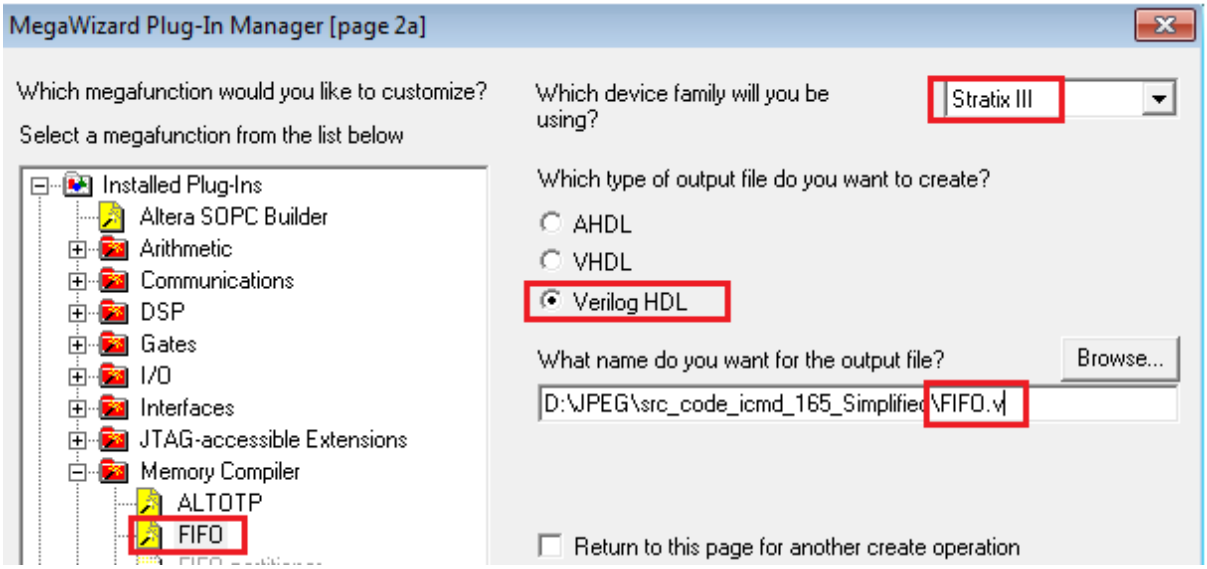

Figure 7. Lựa chọn FIFO trong MegaWizard Plug-In Manager

- Trong tab "Width, Clks, Synchronization", lựa chọn các thông số về độ rộng đường dữ liệu, độ lớn của FIFO

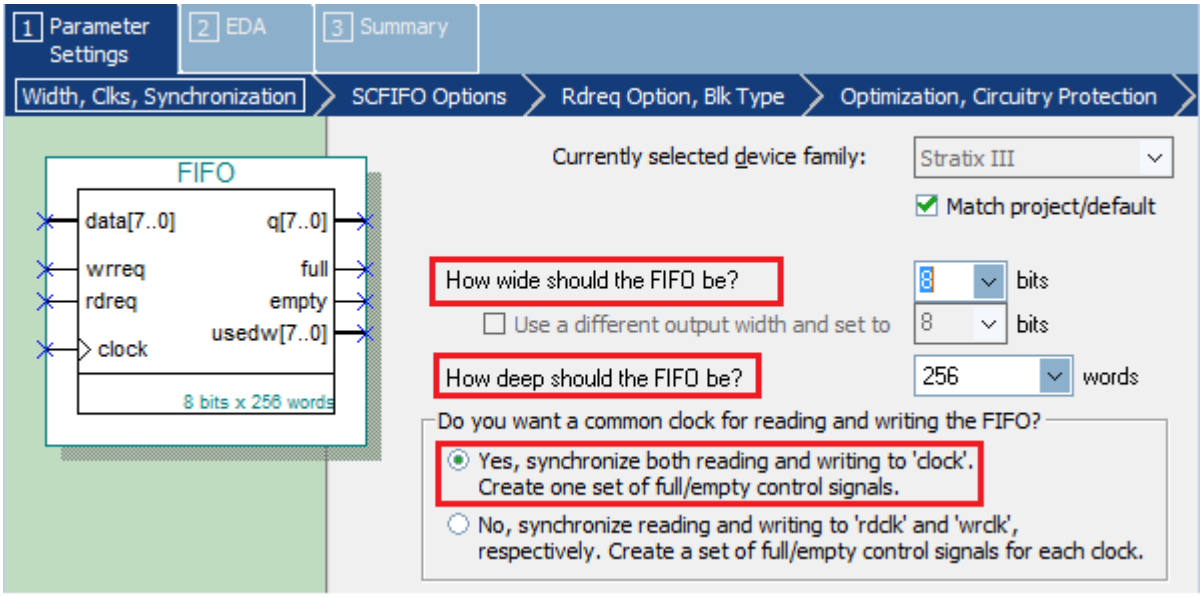

Figure 8. Lựa chọn chế độ cho FIFO

- Lựa chọn các thông số khác cho phù hợp và sau đó Finish để hoàn tất quá trình tạo FIFO. Các tín hiệu full, empty, usedw, almostfull, almostempty cho biết trạng thái của FIFO. Nếu chọn tín hiệu almostfull hay almostempty thì phải chỉ rõ số word bao nhiêu thì

được coi là gần đầy hay gần rỗng. Ta còn có thể chọn thêm tín hiệu reset bất đồng bộ (Asynchronous Reset) nếu cần

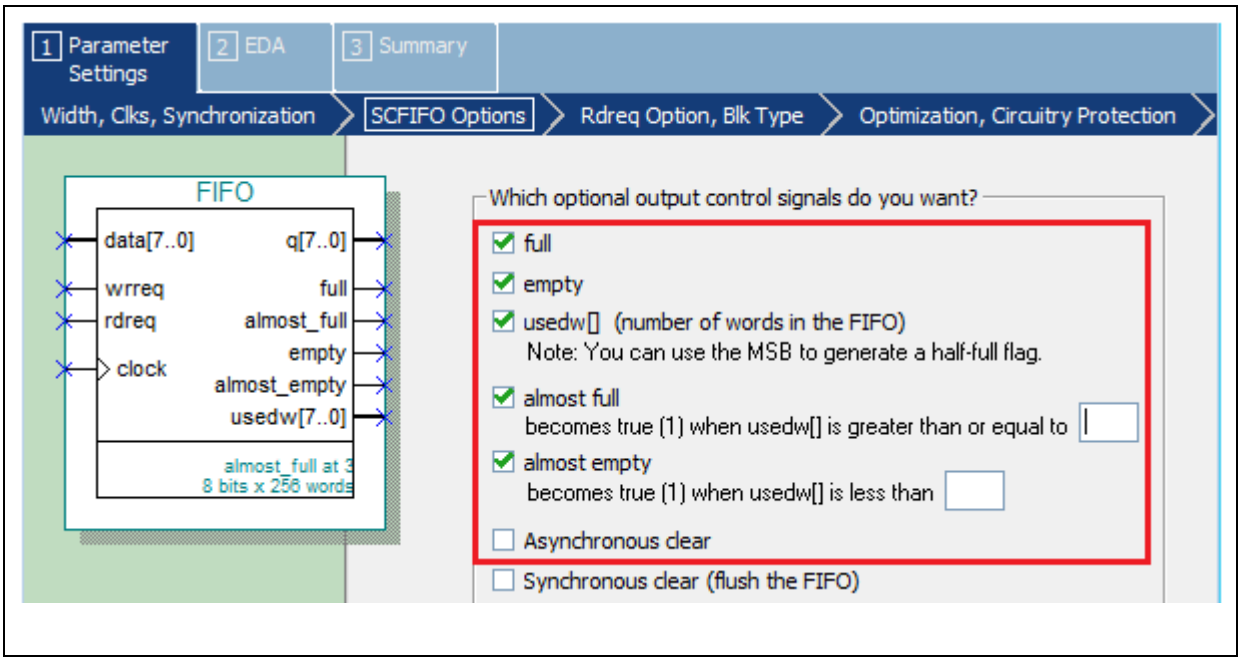

Figure 9. Lựa chọn chế độ cho FIFO (tiếp)

Tín hiệu	Hướng	Mô tả
clock	Input	Tín hiệu xung nhịp giúp toàn bộ khối hoạt động đồng bộ với nhau.
data	Input	Dữ liệu đưa vào FIFO.
sclr	Input	Synchronous Clear, tín hiệu Reset đồng bộ giúp xoá dữ liệu trong FIFO và thiết lập hoạt động từ đầu.
aclr	Input	Asynchronous Clear, tín hiệu Reset **bất** đồng bộ giúp xoá dữ liệu trong FIFO và thiết lập hoạt động từ đầu.
rdreq	Input	Tín hiệu yêu cầu đọc dữ liệu từ FIFO.
wrreq	Input	Tín hiệu yêu cầu ghi dữ liệu vào FIFO.
q	Output	Dữ liệu lấy ra từ FIFO.
almostempty	Output	Tín hiệu báo FIFO gần hết dữ liệu.
almostfull	Output	Tín hiệu báo FIFO gần đầy dữ liệu.
empty	Output	Tín hiệu báo FIFO rỗng.
full	Output	Tín hiệu báo FIFO đầy.
usedw	Output	Thông tin về số Word đã sử dụng.

1.4 PHẦN NÉN ẢNH JPEG (JPEG-BASED COMPRESSION)

Nhận dữ liệu ảnh gốc từ Master Read của bộ DMA và tiến hành nén ảnh theo chuẩn JPEG.

1.4.1 8x8 Block Splitter

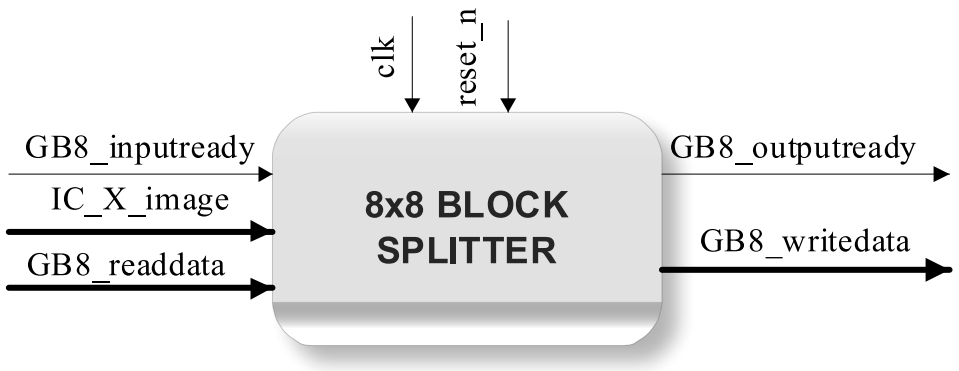

Figure 10. Khối tách dữ liệu thành từng block 8x8 pixels (tức là block 24x8 bytes)

Chịu trách nhiệm tách dữ liệu ảnh gốc thành từng block, mỗi block là 8x8 pixels. Vì mỗi pixel hiện tại gồm 3 thành phần R, G, B, và chiếm 3 bytes (mỗi thành phần chiếm 1 byte) nên block 8x8 sẽ có kích thước là 24x8 bytes

Tín hiệu	Độ rộng	Hướng	Mô tả
clk	1	Input	Tín hiệu xung nhịp giúp toàn bộ khối hoạt động đồng bộ với nhau.
reset_n	1	Input	Tín hiệu Reset đồng bộ tác động mức thấp.
GB8_inputready	1	Input	Tín hiệu thông báo dữ liệu hợp lệ để khối tách block 8x8 đọc vào. Dữ liệu hợp lệ ở trên đường GB8_readdata
IC_X_image	16	Input	Thông tin từ khối Slave Controller cho biết chiều rộng của bức ảnh tính bằng pixel.
GB8_readdata	32	Input	Dữ liệu được đọc từ Master Read
GB8_writedata	32	Output	Dữ liệu ngõ ra của khối tách block 8x8
GB8_outputready	1	Output	Tín hiệu thông báo dữ liệu ngõ ra trên đường GB8_writedata là hợp lệ và có thể được sử dụng ở các khối sau

1.4.2 RGB to YCbCR Converter

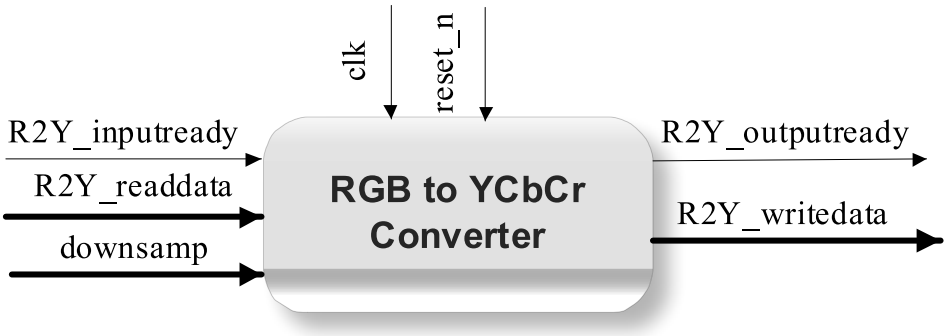

Figure 20. Khối chuyển đổi hệ màu từ RGB sang YCbCr

Khối chuyển đổi hệ màu nhận dữ liệu 32 bit từ khối tách block 8x8, tiến hành tách riêng 3 thành phần R, G, B của từng pixel rồi thực hiện các phép toán để tính ra giá trị các thành phần Y, Cb, Cr trong hệ màu YCbCr. Cuối cùng, tùy vào kiểu giảm mẫu được cấu hình, khối có nhiệm vụ giữ nguyên hoặc loại bỏ 1 số block 8x8 của thành phần Cb và Cr cho phù hợp (figure 1). Ngõ ra của khối chuyển đổi hệ màu là dữ liệu có độ rộng 64 bits, tương ứng với 1 dòng 8 phần tử (mỗi phần tử là thành phần Y, hoặc Cb, hoặc Cr và chiếm 1 byte)

Tín hiệu	Độ rộng	Hướng	Mô tả
clk	1	Input	Tín hiệu xung nhịp giúp toàn bộ khối hoạt động đồng bộ với nhau.
reset_n	1	Input	Tín hiệu Reset đồng bộ tác động mức thấp.
R2Y_inputready	1	Input	Tín hiệu thông báo dữ liệu hợp lệ để khối chuyển hệ màu đọc vào. Dữ liệu hợp lệ ở trên đường R2Y_readdata
downsamp	3	Input	Thông tin từ khối Slave Controller cho biết kiểu giảm mẫu mong muốn là 4:2:2 hay 4:4:4
R2Y _readdata	32	Input	Dữ liệu được đọc từ khối tách block 8x8
R2Y _writedata	32	Output	Dữ liệu ngõ ra của khối chuyển hệ màu từ RGB sang YCbCr
R2Y _outputready	1	Output	Tín hiệu thông báo dữ liệu ngõ ra trên đường R2Y_writedata là hợp lệ và có thể được sử dụng ở các khối sau

Hình sau minh họa từng dòng ngõ ra của khối Converter trong 3 block đầu của kiểu giảm mẫu 4:4:4

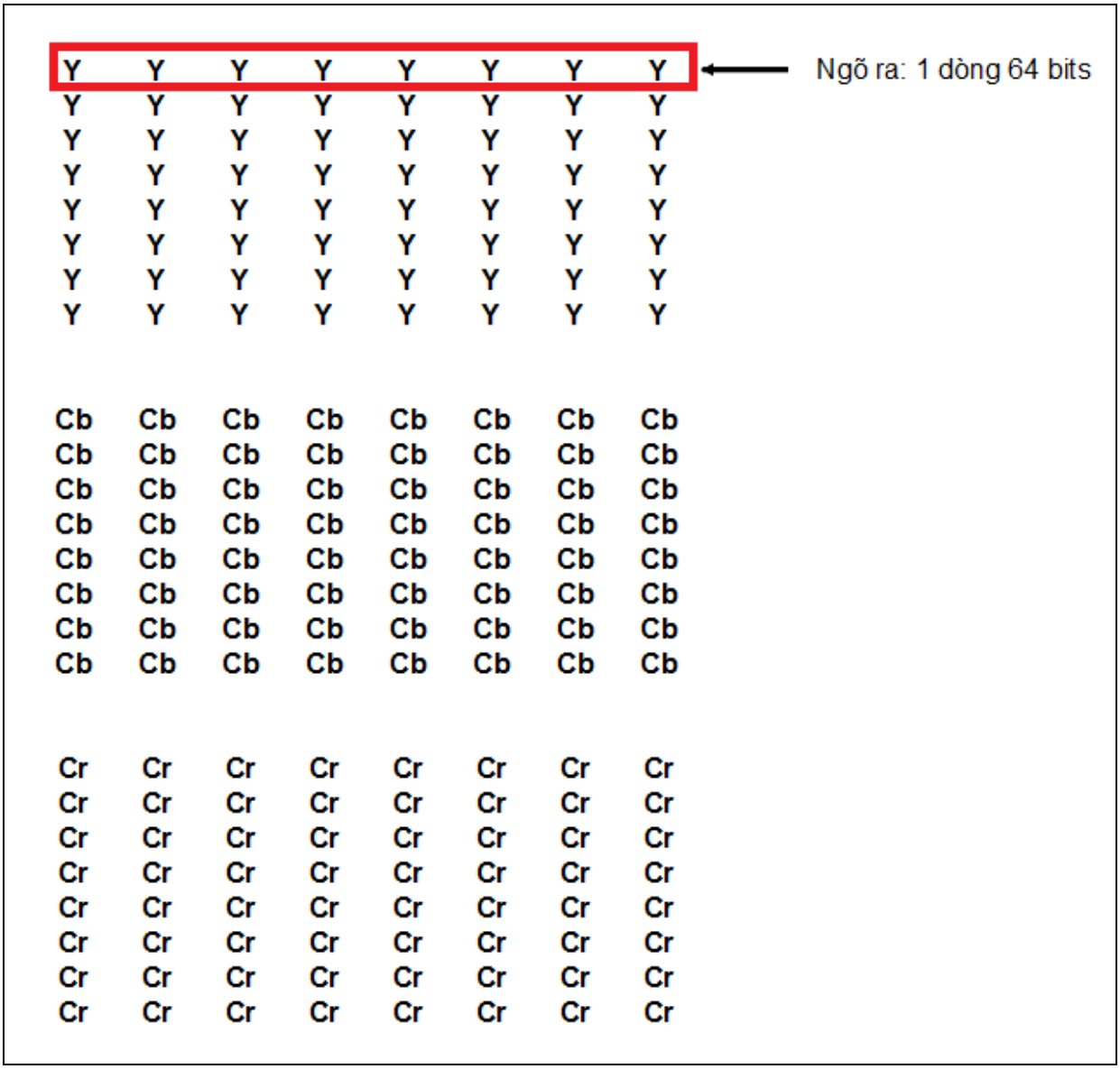

Figure 21. Minh họa ngõ ra từng dòng trong 3 block đầu

1.4.3 2-D BinDCT Processor

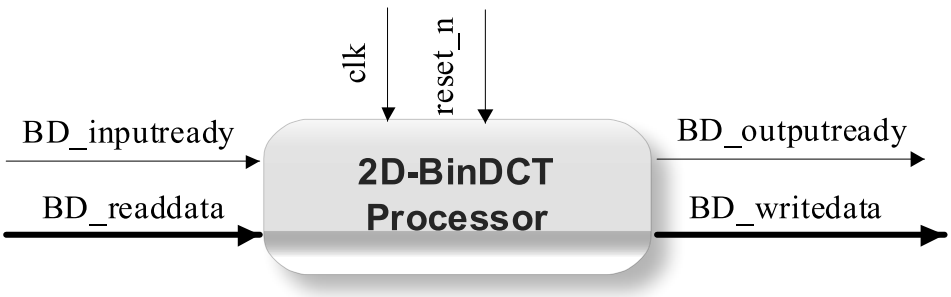

Figure 22. Khối thực hiện biến đổi DCT bằng thuật toán BinDCT

Khối bao gồm bộ biến đổi BinDCT 1 chiều (1D-BinDCT) và bộ chuyển vị ma trận. Dữ liệu 64 bits ngõ vào sẽ qua 2 lần biến đổi BinDCT 1 chiều (figure 2), thực hiện 1 lần xong tiến hành chuyển vị ma trận 8x8 rồi thực hiện DCT lần 2, nên cuối cùng sau 2 lần ta được kết quả biến đổi BinDCT 2 chiều. Ngõ ra sẽ là 1 dòng 8 hệ số DCT và mỗi hệ số có 16 bits (số bit tăng lên cho mỗi phần tử từ 8 lên 16 là do quá trình biến đổi có thực hiện các phép cộng…). Kết quả dữ liệu ra sẽ có độ rộng là 128 bits

Tín hiệu	Độ rộng	Hướng	Mô tả
clk	1	Input	Tín hiệu xung nhịp giúp toàn bộ khối hoạt động đồng bộ với nhau.
reset_n	1	Input	Tín hiệu Reset đồng bộ tác động mức thấp.
BD_inputready	1	Input	Tín hiệu thông báo dữ liệu hợp lệ để đọc vào. Dữ liệu hợp lệ ở trên đường BD_readdata
BD _readdata	64	Input	Dữ liệu được đọc từ khối đổi hệ màu (RGB to YCbCr Converter)
BD _writedata	128	Output	Dữ liệu ngõ ra của khối biến đổi Bin DCT
BD _outputready	1	Output	Tín hiệu thông báo dữ liệu ngõ ra trên đường BD_writedata là hợp lệ và có thể được sử dụng ở các khối sau

1.4.4 Lượng tử hóa và sắp xếp Zigzac

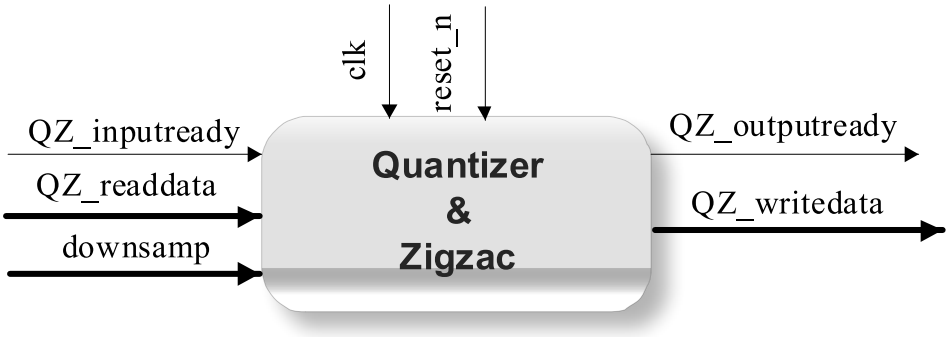

Figure 113. Khối lượng tử hóa và sắp xếp zigzac

Khối chịu trách nhiệm chia từng hệ số DCT trong mỗi ma trận 8x8 với từng hằng số tương ứng trong ma trận lượng tử (được lưu trong ROM). Do đó mà giá trị sau lượng tử sẽ giảm xuống, dẫn đến số bit cần để lưu trữ 1 phần tử chỉ là 13 bits (từ 16 giảm xuống 13), nên dữ liệu ngõ ra sẽ là 1 dòng 104 bits (1 dòng 8 phần tử, mỗi phần tử 13 bits). Phần sắp xếp Zigzac sẽ do 2 bộ nhớ RAM và khối Control Unit điều khiển, theo đó, mỗi phần tử sau lượng tử hóa sẽ được lưu trữ trong RAM, và Control Unit sẽ đưa ra địa chỉ để đọc dữ liệu từ RAM ra ngoài

theo thứ tự zigzac (figure 8). Mỗi RAM sẽ có dung lượng là 13bits x 64, đủ để chứa 1 ma trận 8x8 phần tử

Tín hiệu	Độ rộng	Hướng	Mô tả
clk	1	Input	Tín hiệu xung nhịp giúp toàn bộ khối hoạt động đồng bộ với nhau.
reset_n	1	Input	Tín hiệu Reset đồng bộ tác động mức thấp.
QZ_inputready	1	Input	Tín hiệu thông báo dữ liệu hợp lệ để khối chuyển hệ màu đọc vào. Dữ liệu hợp lệ ở trên đường QZ_readdata
downsamp	3	Input	Thông tin từ khối Slave Controller cho biết kiểu giảm mẫu mong muốn là 4:2:2 hay 4:4:4
QZ _readdata	128	Input	Dữ liệu được đọc từ khối biến đổi BinDCT
QZ _writedata	104	Output	Dữ liệu ngõ ra của khối lượng tử hóa và sắp xếp Zigzac
QZ_outputready	1	Output	Tín hiệu thông báo dữ liệu ngõ ra trên đường QZ_writedata là hợp lệ và có thể được sử dụng ở các khối sau

1.4.5 Khối mã hóa Huffman

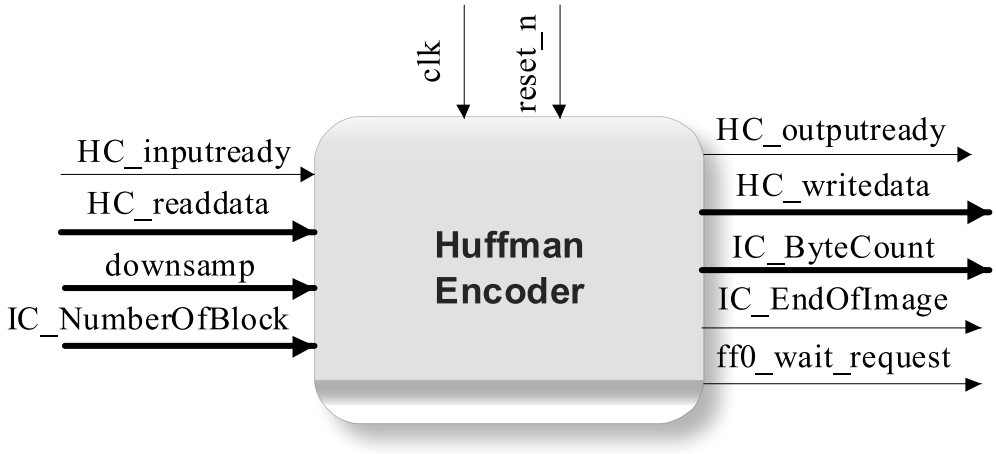

Figure 124. Khối mã hóa Huffman

Bộ mã hóa Huffman gồm 2 phần: phần tiền xử lÝ và phần mã hóa. Phần tiền xử lÝ sẽ loại bỏ các dòng toàn giá trị 0 trong block 8x8, và đánh dấu vị trí cuối cùng của dòng mà còn chứa giá trị khác 0. Sau đó, các phần tử của những dòng được giữ lại sẽ qua bộ biến đổi song song sang nối tiếp để tách riêng từng phần tử trong 1 dòng. Cuối cùng, phần mã hóa sẽ mã hóa Huffman cho những phần tử này

Tín hiệu	Độ rộng	Hướng	Mô tả
Clk	1	Input	Tín hiệu xung nhịp giúp toàn bộ khối hoạt động đồng bộ với nhau.
reset_n	1	Input	Tín hiệu Reset đồng bộ tác động mức thấp.
HC_inputready	1	Input	Tín hiệu thông báo dữ liệu hợp lệ để khối chuyển hệ màu đọc vào. Dữ liệu hợp lệ ở trên đường HC_readdata
IC_NumberOfBlock	20	Input	Thông tin từ khối Slave Controller cho biết số block 8x8 (phần tử) của bức ảnh
downsamp	3	Input	Thông tin từ khối Slave Controller cho biết kiểu giảm mẫu mong muốn là 4:2:2 hay 4:4:4
HC _readdata	104	Input	Dữ liệu được đọc từ khối lượng tử hóa và sắp xếp Zigzac
HC _writedata	32	Output	Dữ liệu ảnh nén JPEG
ff0_wait_request	1	Output	Tín hiệu yêu cầu chờ từ khối Huffman gởi ra cho khối Master Read để tạm ngưng việc đọc dữ liệu vào
IC_EndOfImage	1	Output	Tín hiệu thông báo việc nén ảnh đã hoàn tất
IC_ByteCount	32	Output	Tổng số byte của ảnh nén
HC_outputready	1	Output	Tín hiệu thông báo dữ liệu ngõ ra trên đường HC_writedata là hợp lệ và có thể được sử dụng ở các khối sau

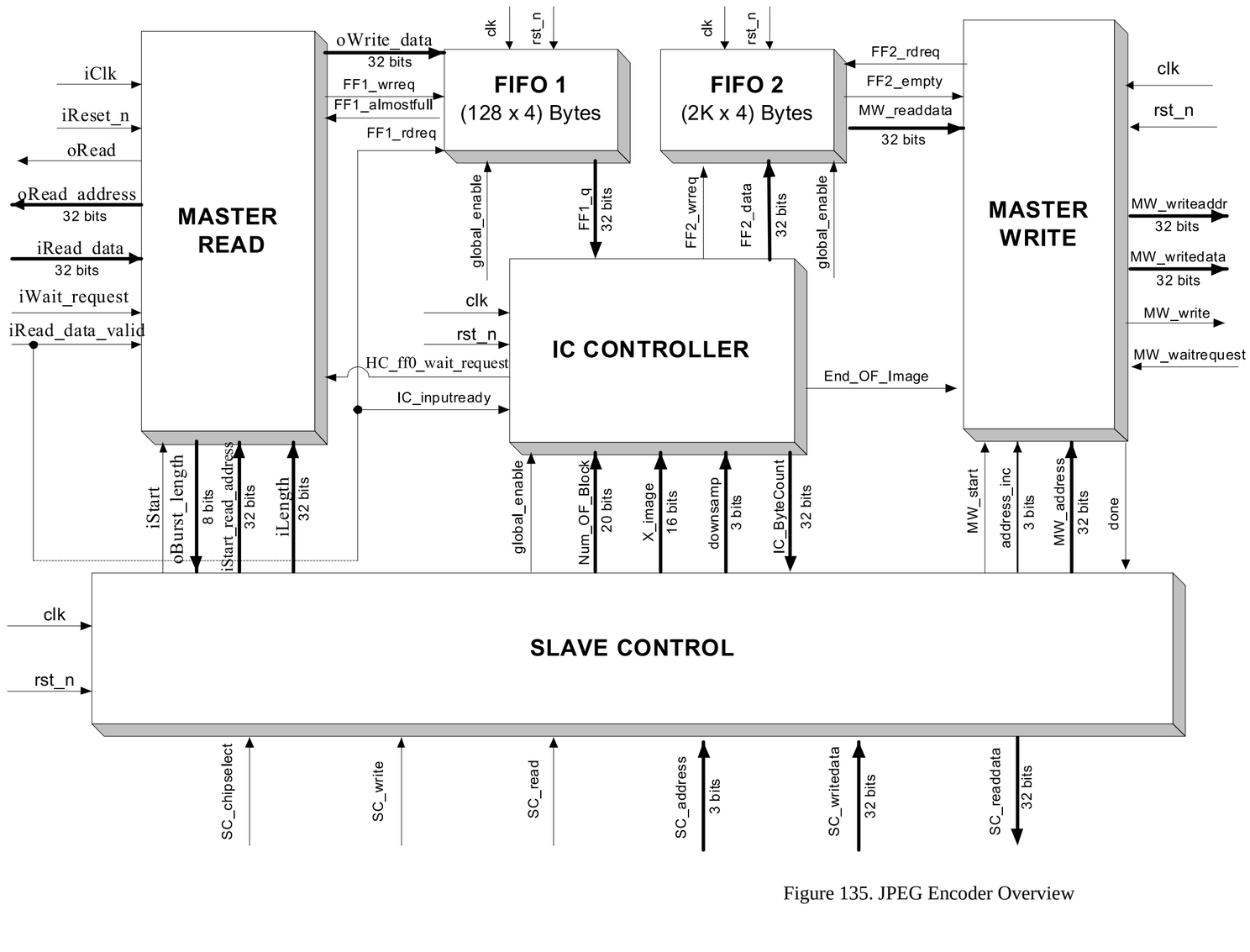

Figure 135. JPEG Encoder Overview

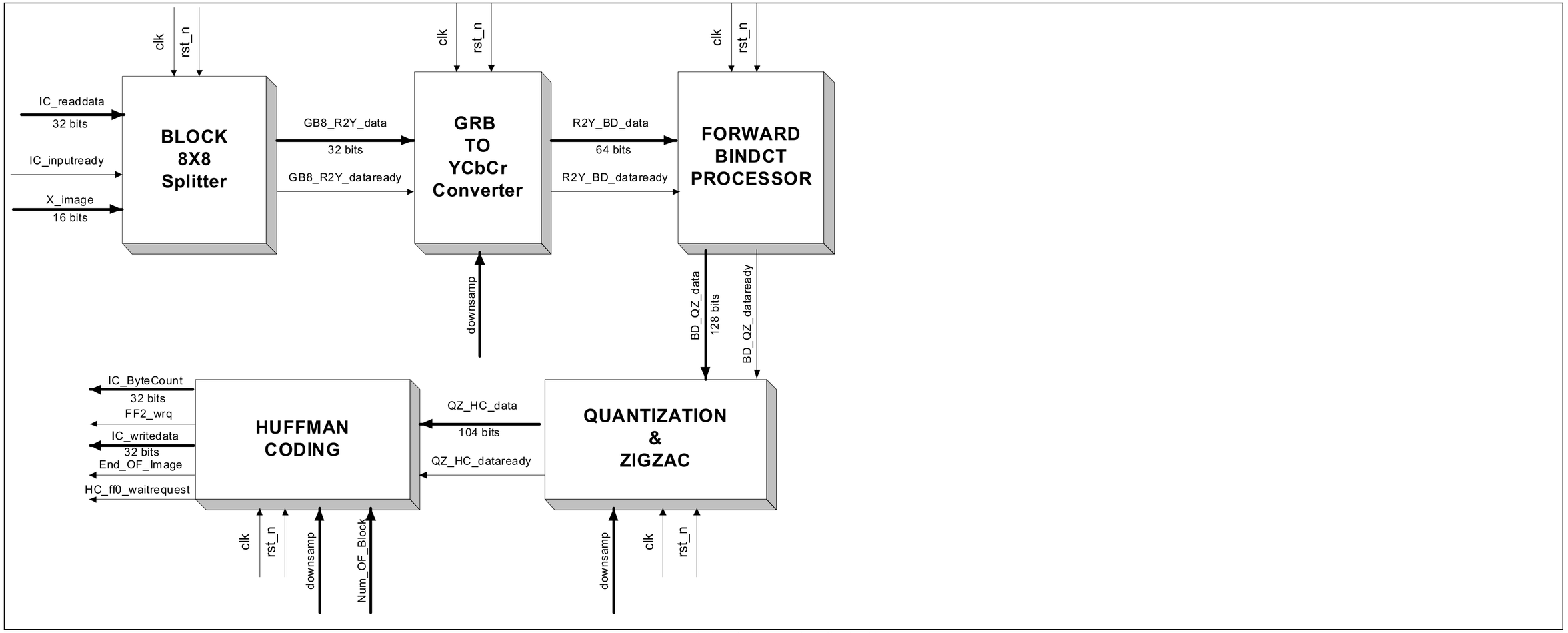

Figure 146. JPEG Encoder Block Diagram

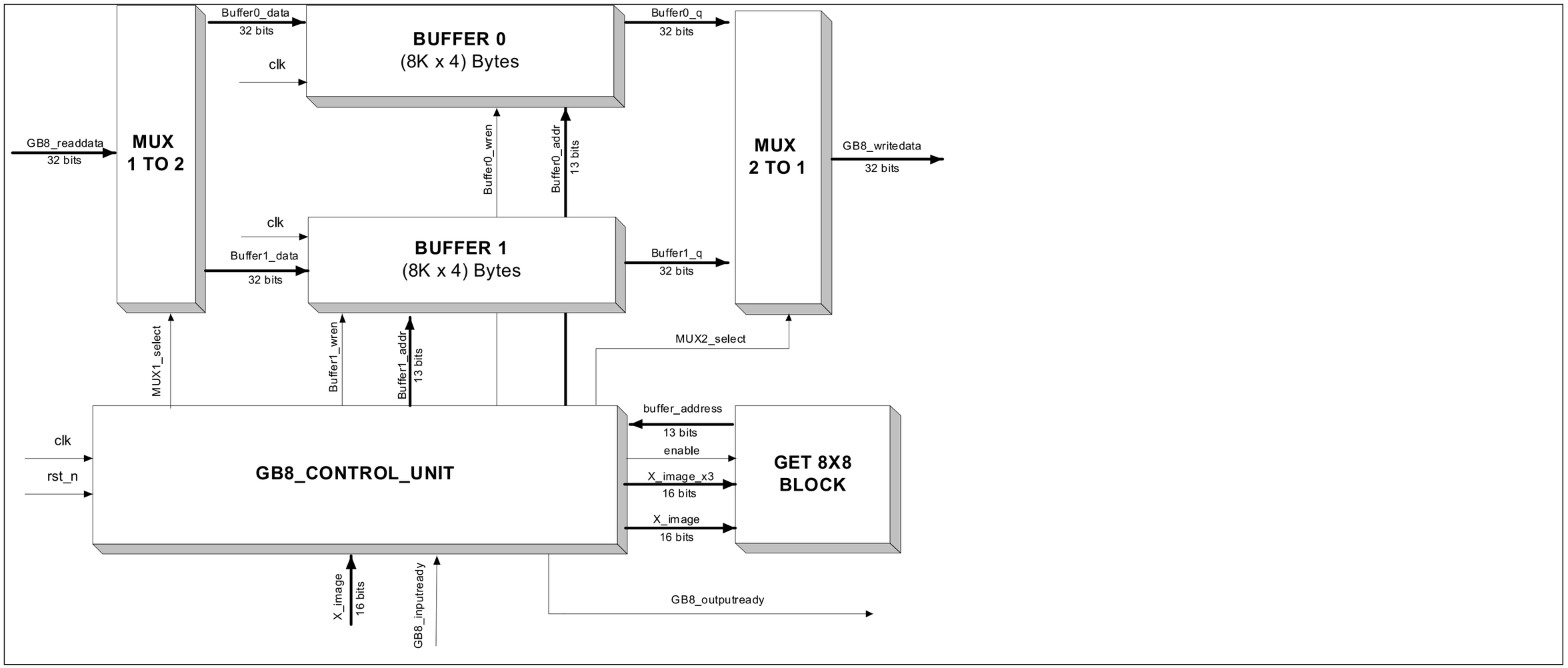

Figure 157. 8x8Block Splitter Diagram

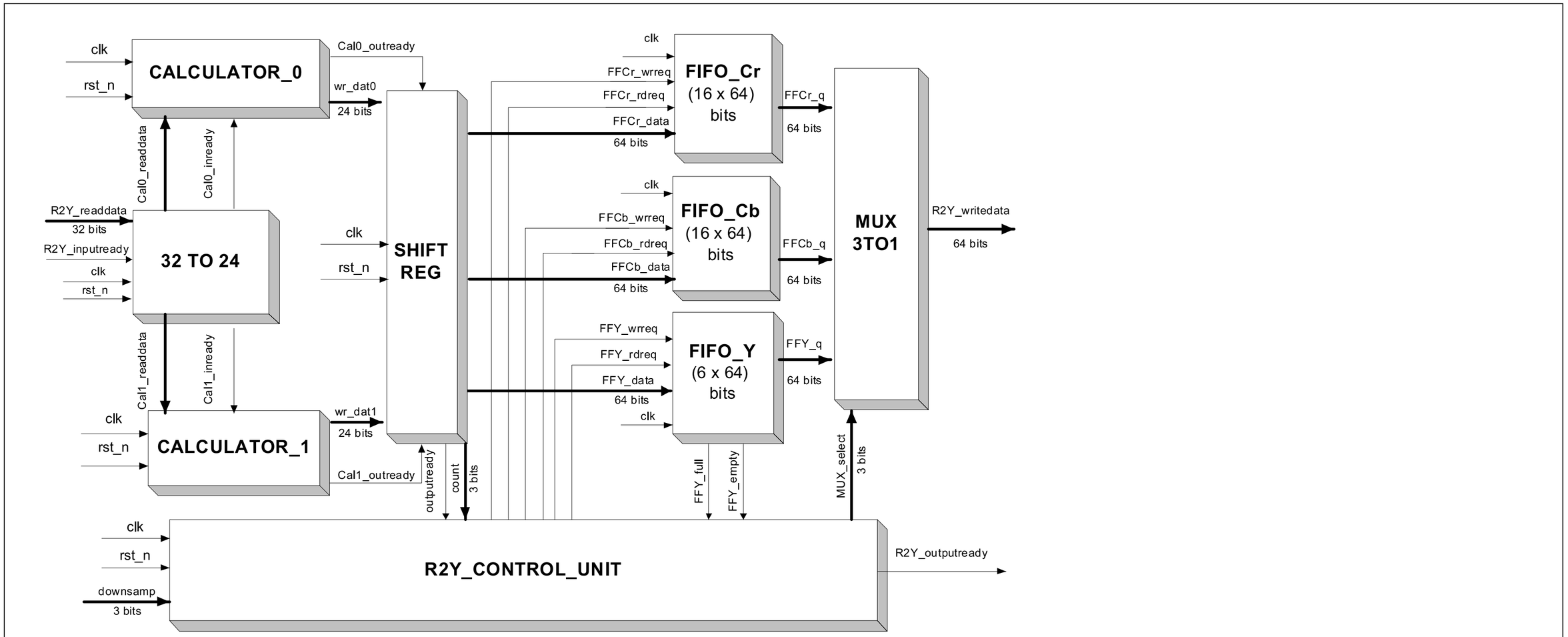

Figure 168. RGB to YCbCr Converter

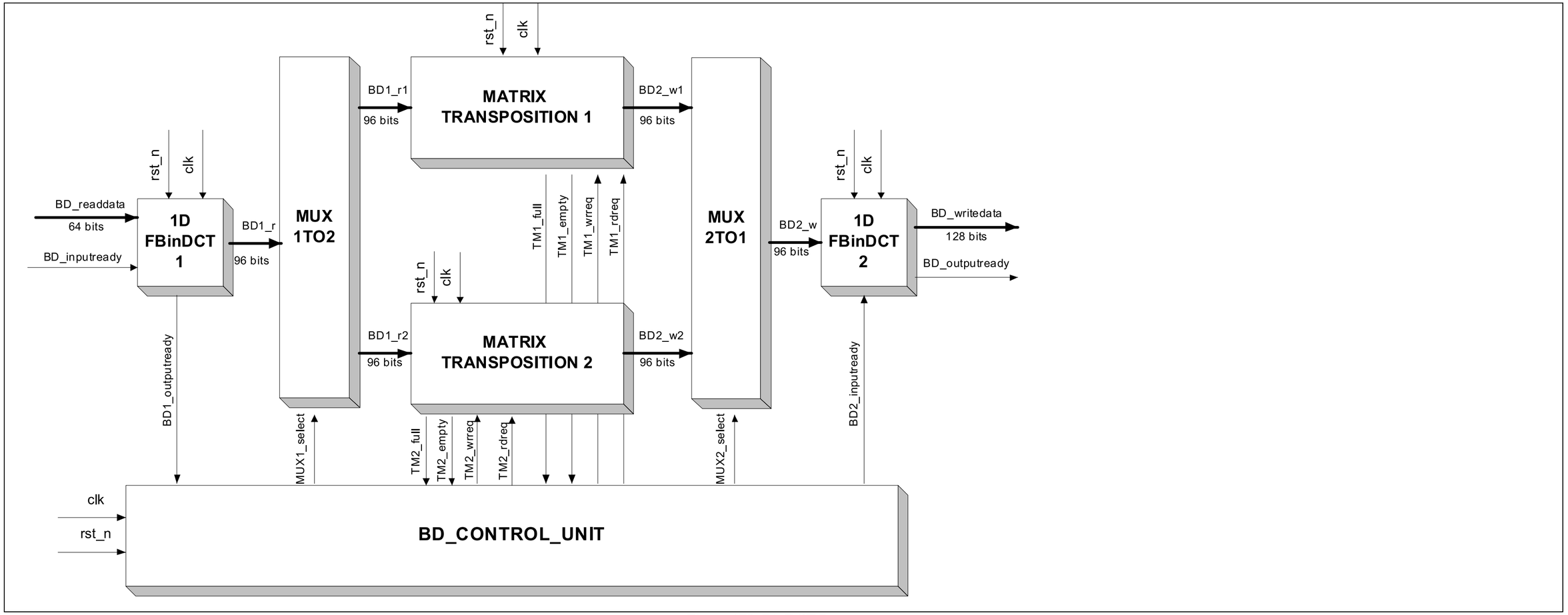

Figure 179. BinDCT Processor

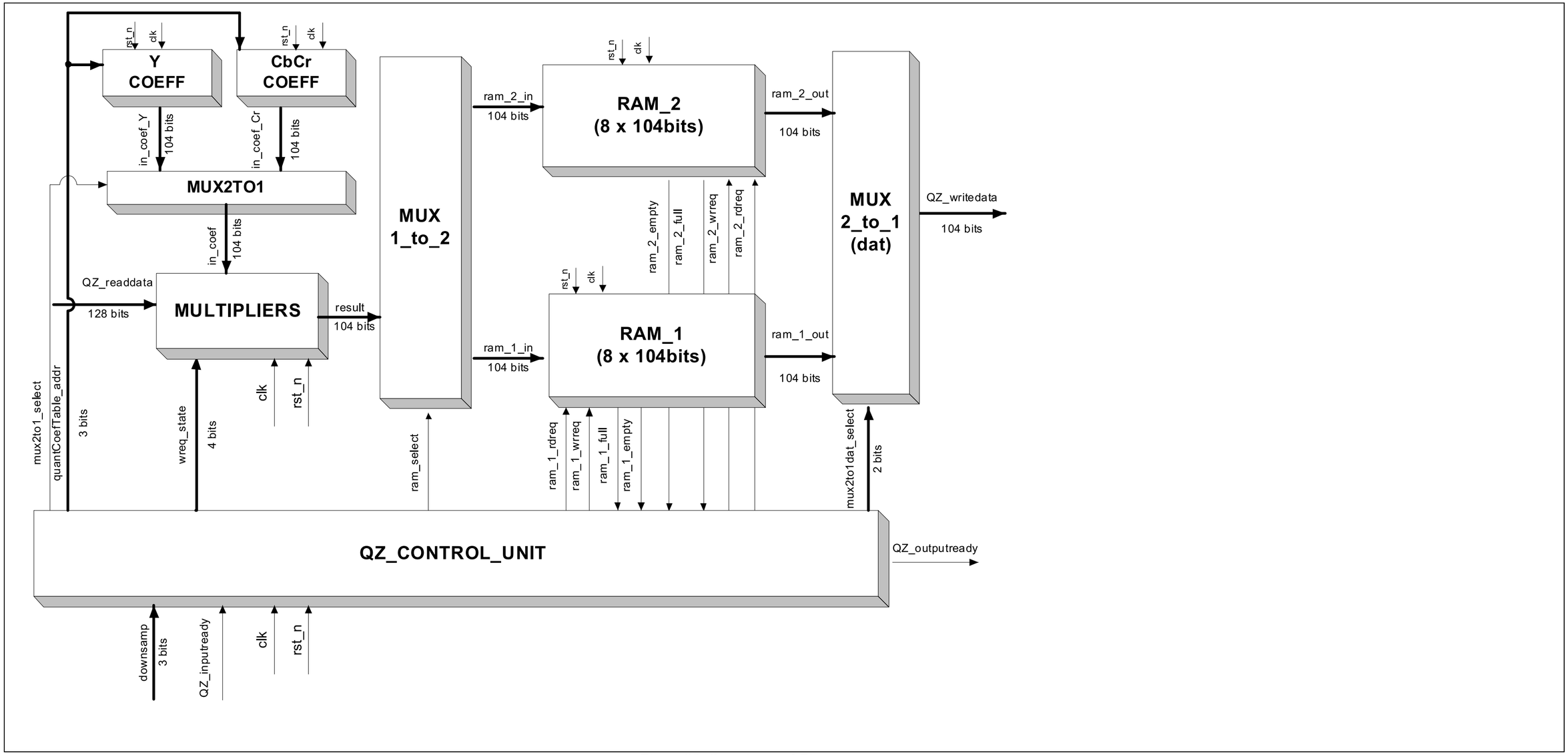

Figure 30. Quantizer and Zigzac Arrangement

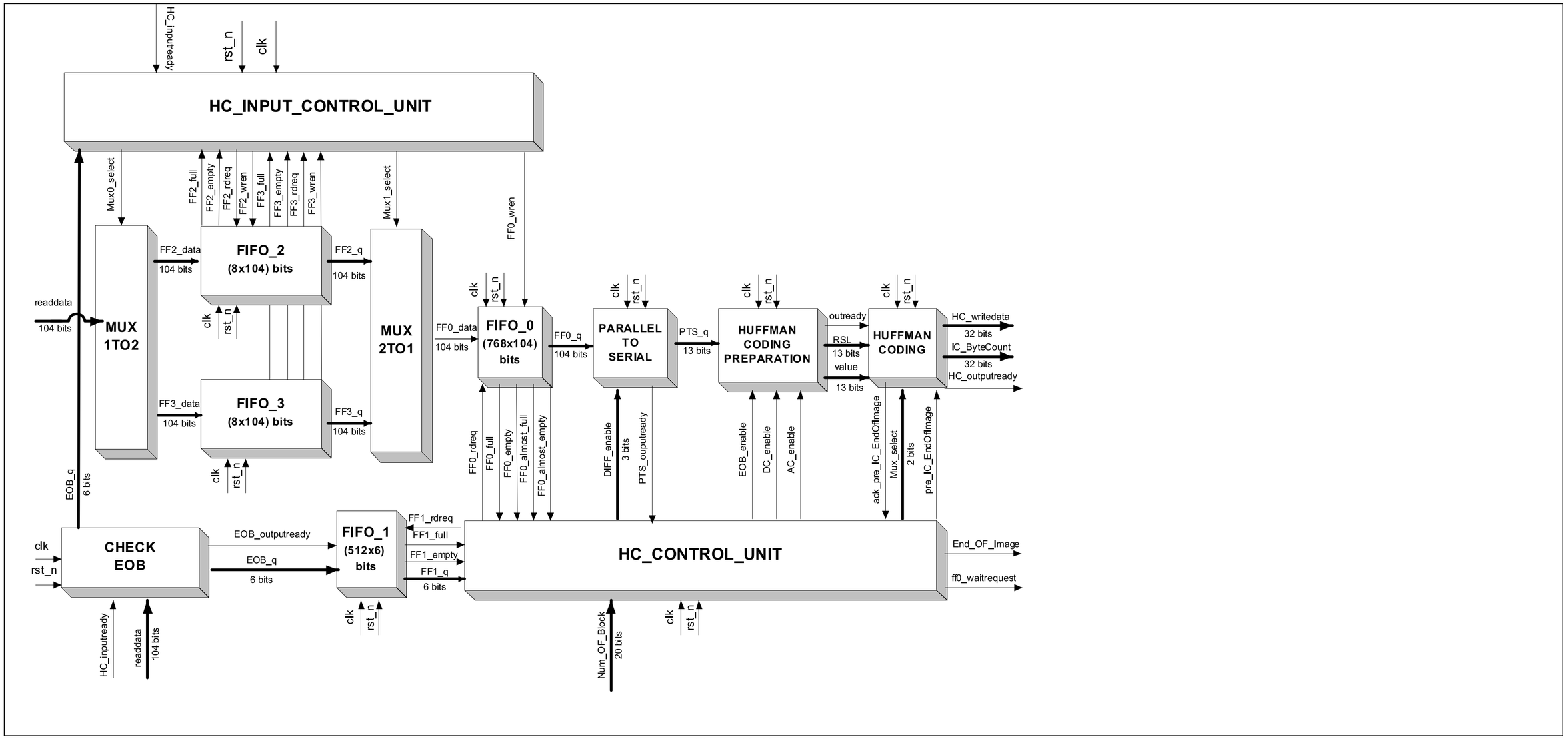

Figure 31. Huffman Encoder