TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN ĐIỆN TỬ - VIỄN THÔNG

----------o0o----------

BÁO CÁO BÀI TẬP LỚN

THIẾT KẾ HỆ THỐNG NHÚNG

Đề tài: Thiết kế DCT 64 điểm

Giảng viên hướng dẫn: TS. Ngô Vũ Đức

Nhóm thực hiện:

Họ và tên	MSSV
Nguyễn Văn Chuyên	20140487
Lê Văn Long	20142662
Vũ Xuân Phong	20143438

Hà Nội, 1/2019

MỤC LỤC

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG BIỂU

DANH MỤC TỪ VIẾT TẮT

Từ viết tắt	**Diễn giải**
DCT	Discrete cosine transform

LỜI MỞ ĐẦU

DCT được sử dụng rộng rãi trong xử lý ảnh, đặc biệt dùng để nén ảnh. DCT hai chiều được ứng dụng trong việc nén ảnh tĩnh và nén từng khung hình video. DCT đa chiều được ứng dụng chủ yếu để nén các luồng video và không gian âm lượng.

Trong chương trình học phần Thiết kế hệ thống nhúng, nhóm chúng em đã chọn đề tài **Thiết kế DCT 64 điểm** để làm đề tài bài tập lớn.

Trong nội dung của bài báo cáo, nhóm chúng em chia báo cáo thành các chương như sau:

- **CHƯƠNG 1: TỔNG QUAN DCT**
- **CHƯƠNG 2: THIẾT KẾ DCT**
- **CHƯƠNG 3: THỰC THI TỪNG KHỐI**
- **CHƯƠNG 4: KẾT LUẬN**

Chúng em xin chân thành cảm ơn TS. Ngô Vũ Đức, là giảng viên hướng dẫn chúng em trong môn Thiết kế hệ thống nhúng. Trong quá trình thực hiện, dù nhóm chúng em đã cô gắng hoàn thiện đề tài nhưng do kiến thức còn hạn hẹp nên khó tránh khỏi những sai sót. Chúng em mong sẽ nhận được sự góp ý của thầy để có thể hoàn thiện đề tài hơn nữa.

Chúng em xin chân thành cảm ơn!

NHÓM SINH VIÊN

CHƯƠNG 1:TỔNG QUAN DCT

Biến đổi Cosine rời rạc (DCT) lần đầu tiên được đề xuất bởi Ahmed et al. (1974), và cho đến nay DCT ngày càng đóng vai trò quan trọng trong việc nén ảnh.

1.1 Giới thiệu

DCT được sử dụng rộng rãi trong xử lý ảnh, đặc biệt dùng để nén. DCT hai chiều được ứng dụng trong việc nén ảnh tĩnh và nén từng khung hình video. DCT đa chiều được ứng dụng chủ yếu để nén các luồng video và không gian âm lượng.

1.1.1 DCT là gì?

Biến đổi Cosine rời rạc (Discrete cosine transform - DCT) là một biến đổi dữ liệu dưới dạng biên độ thành dữ liệu dưới dạng số.

Mục đích của DCT là loại bỏ sự dư thừa dữ liệu trong không gian. Ngoài ra, DCT cũng được ứng dụng để chuyển dữ liệu đa chiều sang miền tần số.

DCT được chia thành hai loại là:

- DCT một chiều
- DCT hai chiều

Thông lượng ảnh hưởng trực tiếp đến chất lượng trải nghiệm của nội dung đa phương tiện. Do đó việc triển khai DCT trên phần cứng sẽ cho thông lượng cao hơn các giải pháp phần mềm. Phần cứng DCT đặc biệt làm giảm tải tính toán từ bộ xử lý và do đó cải thiện hiệu suất của hệ thống đa phương tiện hoàn chỉnh.. Một yếu tố quan trọng khác ảnh hưởng đến chất lượng là hiệu ứng chiều dài thanh ghi hữu hạn ảnh hưởng đến độ chính xác của quá trình biến đổi ngược.

Do đó, động lực để điều tra các thuật toán DCT cụ thể của phần cứng là rõ ràng. Vì thuật toán DCT 2 chiều là điển hình nhất cho nén hình ảnh, trọng tâm chính của chương này sẽ là triển khai phần cứng hiệu quả của nén dựa trên DCT 2 chiều bằng cách giảm số lượng tính toán, tăng độ chính xác của quá trình tái tạo và giảm khu vực chip. Đổi lại điều này làm giảm mức tiêu thụ năng lượng của kỹ thuật nén. Khi số lượng ứng dụng yêu cầu thuật toán DCT chiều cao hơn đang tăng lên,

người ta sẽ chú ý đặc biệt đến các thuật toán có thể dễ dàng mở rộng cho các trường hợp chiều cao hơn. Tiêu chuẩn JPEG đã có từ cuối những năm 1980 và là một giải pháp đầu tiên hiệu quả cho việc tiêu chuẩn hóa hình ảnh nén. Mặc dù JPEG có một số chiến lược rất hữu ích cho việc định lượng và nén DCT, nhưng nó chỉ được phát triển cho mức nén thấp. Kích thước khối DCT 8 × 8 đã được chọn cho tốc độ (hiện không phải là vấn đề, với sự xuất hiện của bộ xử lý nhanh hơn) không phải vì hiệu suất.

Giống như các biến đổi khác, Biến đổi Cosine rời rạc (DCT) cố gắng giải mã dữ liệu hình ảnh. Sau khi giải mã, mỗi hệ số biến đổi có thể được mã hóa độc lập mà không làm mất hiệu quả nén. Phần này mô tả DCT và một số thuộc tính quan trọng của nó.

1.1.2 Ứng dụng của DCT

Biến đổi DCT cũng được ứng dụng để chuyển dữ liệu đa chiều sang miền tần số DCT, trong đó các hoạt động khác nhau, như hình mờ dữ liệu phổ trải rộng, có thể được thực hiện theo cách dễ dàng và hiệu quả hơn.

Vô số bài báo thảo luận về thuật toán DCT đang chứng kiến mạnh mẽ về tầm quan trọng và khả năng ứng dụng của nó.

Việc triển khai phần cứng đặc biệt thú vị để thực hiện các thuật toán song song cao có thể đạt được thông lượng cao hơn nhiều so với các giải pháp phần mềm. Ngoài ra, phần cứng DCT có mục đích đặc biệt sẽ giải phóng tải tính toán từ bộ xử lý và do đó cải thiện hiệu suất của hệ thống đa phương tiện hoàn chỉnh. Thông lượng ảnh hưởng trực tiếp đến chất lượng trải nghiệm của nội dung đa phương tiện. Một yếu tố quan trọng khác ảnh hưởng đến chất lượng là hiệu ứng chiều dài thanh ghi hữu hạn đối với tính chính xác của quá trình biến đổi nghịch đảo.

1.1.3 Đặt vấn đề

Biến đổi Cosine rời rạc là một trong những kỹ thuật biến đổi rộng rãi nhất trong xử lý tín hiệu số. Do đó, động lực cho việc thiết kế kiến trúc biến đổi Cosine rời rạc là rõ ràng. Vì đây cũng là biến đổi chuyên sâu tính toán nhất đòi hỏi nhiều

phép nhân và bổ sung. Xử lý dữ liệu thời gian thực đòi hỏi phải sử dụng phần cứng cho mục đích đặc biệt liên quan đến hiệu quả phần cứng cũng như thông lượng cao. Nhiều thuật toán DCT đã được đề xuất để đạt được DCT tốc độ cao.

1.2 DCT một chiều

Định nghĩa DCT phổ biến nhất của chuỗi 1 chiều dài N là:

$$C(u) = \alpha(u)\sum_{x=0}^{N-1} f(x)\cos\left[\frac{\pi(2x+1)u}{2N}\right]$$

Với u = 0, 1, 2 …, N - 1. Tương tự, phép biến đổi DCT ngược được định nghĩa là:

$$f(x) = \sum_{u=0}^{N-1}\alpha(u)c(u)\cos\left[\frac{\pi(2x+1)u}{2N}\right]$$

Với α (u) là:

$$\alpha(u) = \begin{cases}\sqrt{\frac{1}{N}}, u = 0\\ \sqrt{\frac{2}{N}}, u \neq 0\end{cases}$$

Trong trường hợp u = 0 ta có công thức tính DCT sau:

$$c(u=0) = \sqrt{\frac{1}{N}}\sum_{x=0}^{N-1} f(x)$$

Do đó, hệ số biến đổi đầu tiên (u = 0) là giá trị trung bình của chuỗi mẫu. Trong tài liệu, giá trị này được gọi là Hệ số DC. Tất cả các hệ số biến đổi khác được gọi là Hệ số AC.

Bỏ qua thành phần f (x) và α (u):

$$\sum_{x=0}^{N-1}\cos\left[\frac{\pi(2x+1)u}{2N}\right]$$

Cho N = 8 và các giá trị khác nhau của u. Dạng sóng trên cùng bên trái (u = 0) biểu thị giá trị không đổi (DC), trong khi đó tất cả các dạng sóng khác (u ≠ 0) cung

cấp dạng sóng với tần số tăng dần. Các dạng sóng này được gọi là hàm cơ sở cosine.[1]

1.3 DCT hai chiều

DCT hai chiều là phần mở rộng trực tiếp của DCT một chiều và được đưa ra bởi công thức sau:

$$C(u,v) = \frac{1}{4}\alpha(u)\alpha(v)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)\frac{\cos(2x+1)u\pi}{2N}\frac{\cos(2y+1)v\pi}{2N}$$

Với u, v = 0, 1, 2…, N - 1 và α (u) và α (v)

Biến đổi DCT ngược:

$$f(x,y) = \sum_{u=0}^{N-1}\sum_{v=0}^{N-1}\alpha(u)\alpha(v)C(u,v)\cos\left[\frac{\pi(2x+1)u}{2N}\right]\cos\left[\frac{\pi(2y+1)v}{2N}\right]$$

Với x, y = 0, 1, 2…, N

Các hàm cơ sở DCT hai chiều có thể được tạo bằng cách nhân các hàm cơ sở DCT một chiều theo chiều ngang với tập hợp định hướng theo chiều dọc của các hàm tương tự. Các hàm cơ bản cho N = 8 được hiển thị trong.

Hàm cơ sở thể hiện sự tăng dần về tần số cả theo hướng dọc và ngang. Hàm cơ sở trên cùng bên trái của các kết quả từ phép nhân của thành phần DC với chuyển vị của nó. Do đó, hàm này có một giá trị không đổi và được gọi là hệ số DC.[1]

1.4 Phân tích thuộc tính DCT

Lợi thế chính của việc chuyển đổi hình ảnh là loại bỏ sự dư thừa dữ liệu giữa các pixel lân cận. Điều này dẫn đến các hệ số biến đổi không tương quan có thể được mã hóa độc lập.

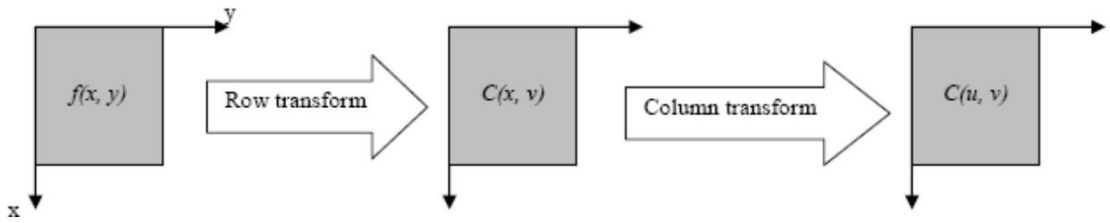
Tự động tương quan chuẩn hóa của các hình ảnh trước và sau DCT. Rõ ràng, biên độ của tự động tương quan sau khi hoạt động DCT rất nhỏ ở tất cả các độ trễ. Do đó, có thể suy ra rằng DCT thể hiện các thuộc tính phân rã tuyệt vời.

1.5 Phân tích DCT

Phương trình biến đổi DCT có thể được biểu diễn dưới dạng

$$C(u,v) = \sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)\cos\left[\frac{\pi(2x+1)u}{2N}\right]\cos\left[\frac{\pi(2y+1)v}{2N}\right]$$

Thuộc tính này, được gọi là khả năng phân tách, có lợi thế nguyên tắc là C(u,v) có thể được tính theo hai bước bằng các thao tác DCT một chiều liên tiếp trên các hàng và cột của hình ảnh. Ý tưởng này được minh họa bằng đồ họa trong Hình 2.7. Các đối số được trình bày có thể được áp dụng giống hệt cho tính toán DCT nghịch đảo.



Hình 1.1 Tính toán DCT hai chiều bằng cách sử dụng thuộc tính tách rời

CHƯƠNG 2:THIẾT KẾ DCT

2.1 Biến đổi công thức

Đầu vào của thiết kế là một ảnh hai chiều 8x8 với các giá trị điểm ảnh từ 0 – 255, để thực hiện DCT 2D 64 điểm ta thực hiện DCT 1D 8 điểm theo hàng và thực hiện DCT 1D 8 điểm theo từng cột. Để thực hiện ta cần biến đổi công thức DCT 1D về dạng tối giản nhất:

Với N = 8 ta có công thức:

$$x(u) = \frac{1}{2} c(u) \sum_{0}^{7} a_i \cos\left[\frac{(2i+1)\pi u}{16}\right]$$

$C(u) = 1/\sqrt{2}$ Với u = 0

$C(u) = 1$ Với u != 0

Giả sử ta có 8 điểm giá trị đầu vào:

a0	a1	a2	a3	a4	a5	a6	a7

Công thức được biến đổi tiếp như sau:

$$x(u) = \frac{c(u)}{2}\left[a_0 \cos(\frac{\pi u}{16}) + a_1 \cos(\frac{3\pi u}{16}) + a_2 \cos(\frac{5\pi u}{16}) + a_3 \cos(\frac{7\pi u}{16}) + a_4 \cos(\frac{9\pi u}{16}) + a_5 \cos(\frac{11\pi u}{16}) + a_6 \cos(\frac{13\pi u}{16}) + a_7 \cos(\frac{15\pi u}{16})\right]$$

Và dựa vào tính chất của hàm cos, thực hiện biến đổi tiếp:

$$x(u) = \frac{c(u)}{2}\left[a_0 \cos(\frac{\pi u}{16}) + a_1 \cos(\frac{3\pi u}{16}) + a_2 \cos(\frac{5\pi u}{16}) + a_3 \cos(\frac{7\pi u}{16}) + a_4 \cos(u\pi - \frac{7\pi u}{16}) + a_5 \cos(u\pi - \frac{5\pi u}{16}) + a_6 \cos(u\pi - \frac{3\pi u}{16}) + a_7 \cos(u\pi - \frac{\pi u}{16})\right]$$

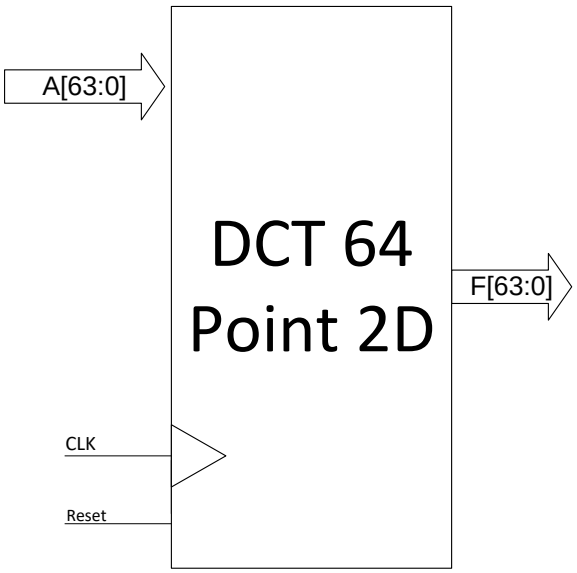
Với giá trị u lẻ:

$$x(u) = \frac{c(u)}{2}\left[(a_0 - a_7)\cos(\frac{\pi u}{16}) + (a_1 - a_6)\cos(\frac{3\pi u}{16}) + (a_2 - a_5)\cos(\frac{5\pi u}{16}) + (a_3 - a_4)\cos(\frac{7\pi u}{16})\right]$$

Với giá trị u chẵn:

$$x(u) = \frac{c(u)}{2}\left[(a_0 + a_7)\cos(\frac{\pi u}{16}) + (a_1 + a_6)\cos(\frac{3\pi u}{16}) + (a_2 + a_5)\cos(\frac{5\pi u}{16}) + (a_3 + a_4)\cos(\frac{7\pi u}{16})\right]$$

2.2 Phân tích yêu cầu



Hình 2.1 Yêu cầu thiết kế DCT 64 Point

DCT 64 point được thiết kế bao gồm:

Input:

- 64 điểm đầu vào từ A0 đến A63. Các giá trị đầu vào được biểu diễn dưới dạng 24bit trong đó có 16bit đại diện cho phần nguyên và 8bit đại diện cho phần thập phân.
- Tín hiệu CLK làm tần số xung nhịp cho chip
- Tín hiệu reset để khởi động hệ thông

Output:

- Đầu ra giá trị sau khi đã qua biến đổi DCT bao gồm 64 điểm từ F0 đến F63.

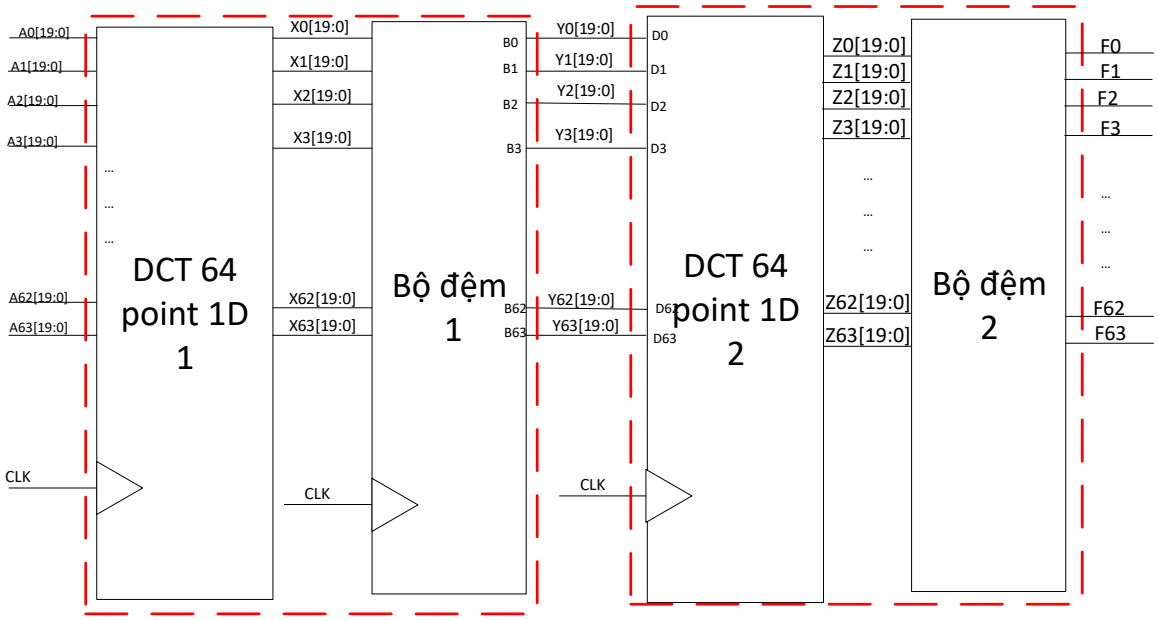
2.3 Thiết kế chi tiết

Bao gồm các thiết kế chi tiết liên quan đế các khối bên trong DCT cũng như tín hiệu vào ra và số bit quy định trong từng khối một.

2.3.1 Thiết kế TOP BLOCK

a) Thiết kế

Thiết kế DCT 64 point 2D bao gồm 2 biến đổi DCT 64 point 1D gộp thanh. Trong thiết kế theo thứ tự từ trái qua phải như hình trên bao gồm.



Hình 2.2 Thiết kế top block

- DCT 64 point 1D: biến đổi DCT 1D thực hiện biến đổi theo hàng.
- Bộ đệm: lưu trữ các giá trị tính toán từ DCT 1D biến đổi theo hàng để từ đó làm đầu vào cho các giá trị biến đổi DCT 1D theo cột
- DCT 64 point 1D: biến đổi DCT 1D thực hiện các giá trị theo cột.

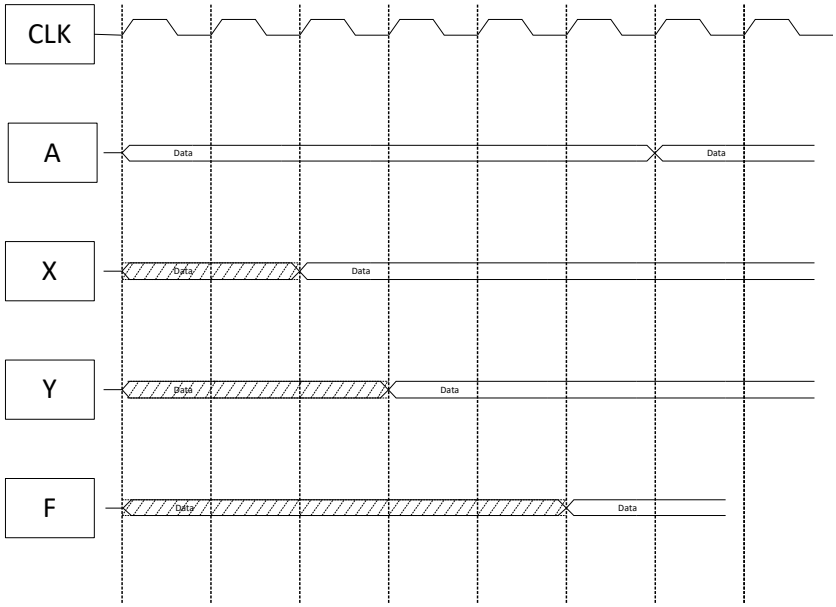
b) Mô tả vào ra

Bảng 2.1 Mô tả vào ra của các khối

Column1	Column2	Column3	Column4
tín hiệu	kết nối	I/O	mô tả
Ai		I	Original image
Bi	Ci	O	Tín hiệu sau DCT 1D được đảo hàng thành cộ
Ci		O	
Di	Ci	I	Tín hiệu vào khối DCT 1D thứ 2
Fi		O	Tín hiệu sau DCT 2D

Bao gồm tín hiệu vào từng khối được mô tả như hình vẽ bên trên.

c) Mô tả Timming



Hình 2.3 Timming cho top block

Tín hiệu thực hiện hết chuyển đổi DCT được thực hiện 5 xung nhịp trong đó hai xung nhịp đầu tiên DCT 64 point 2D sẽ thực hiện tính toán trong DCT 1D và lưu giá trị vào bộ đệm. Xung nhịp thứ 3 sẽ đọc giá trị từ bộ đệm để cho tín hiệu vào khối DCT 1D biến đổi theo cột, hai xung nhịp thứ 4 và 5 sẽ thực hiện biến đổi DCT 1D theo cột và xuất ra dữ liệu ở đầu ra.

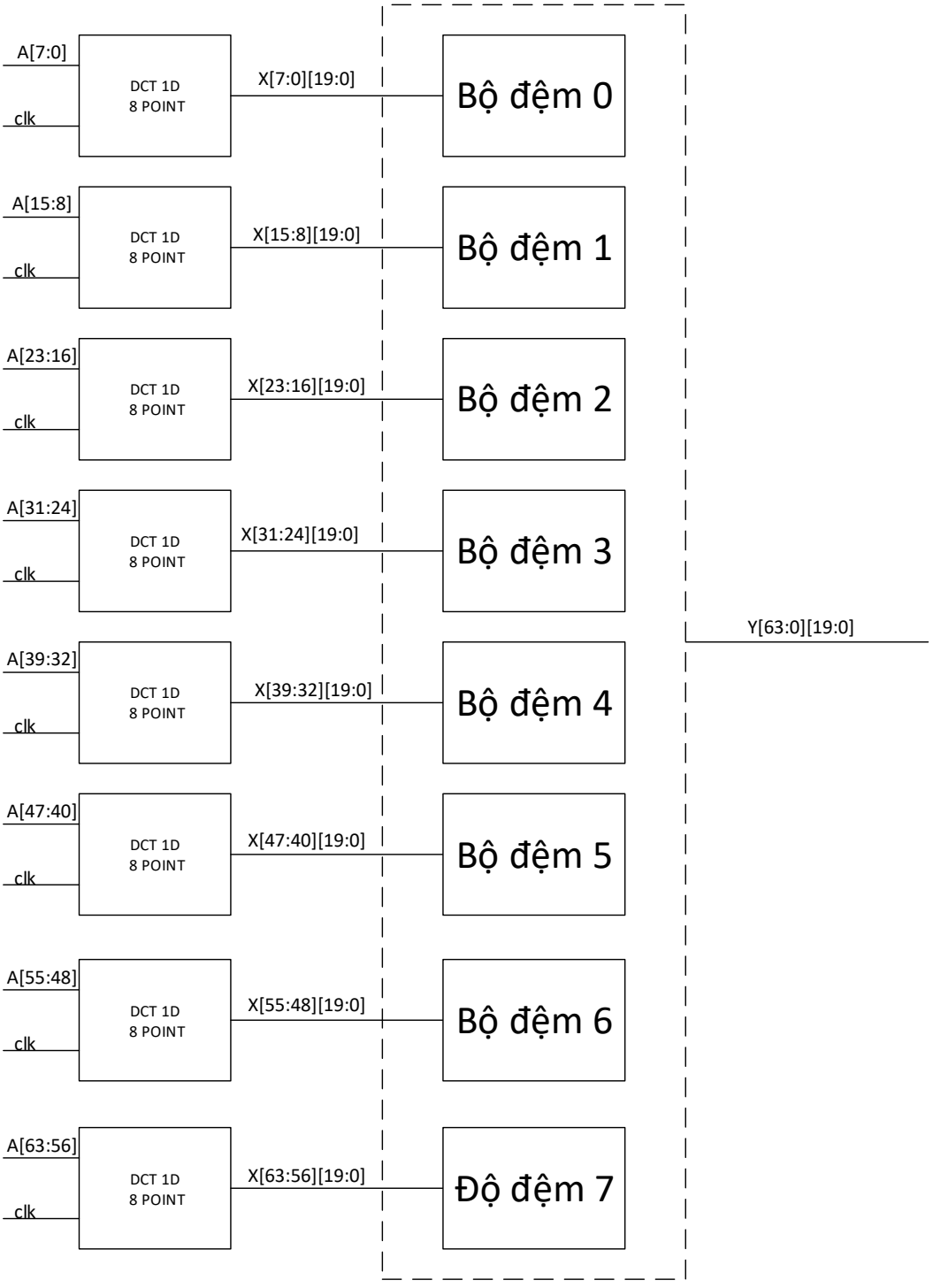
2.3.2 Thiết kế chi tiết DCT 1D

a) Thiết kế

Trong thiết kế DCT 1D nhóm thiết kế cho biến đổi DCT 64 ponit sẽ bao gồm 8 biến đổi DCT 1D 8 điểm gộp thành với tín hiệu xuất ra được lưu trong bộ đệm của từng khối, mỗi khối DCT 1D 8 điểm có giá trị đầu vào ra sẽ phụ trách theo như mô tả từng khối từ trên xuống theo thứ tự dưới đây:

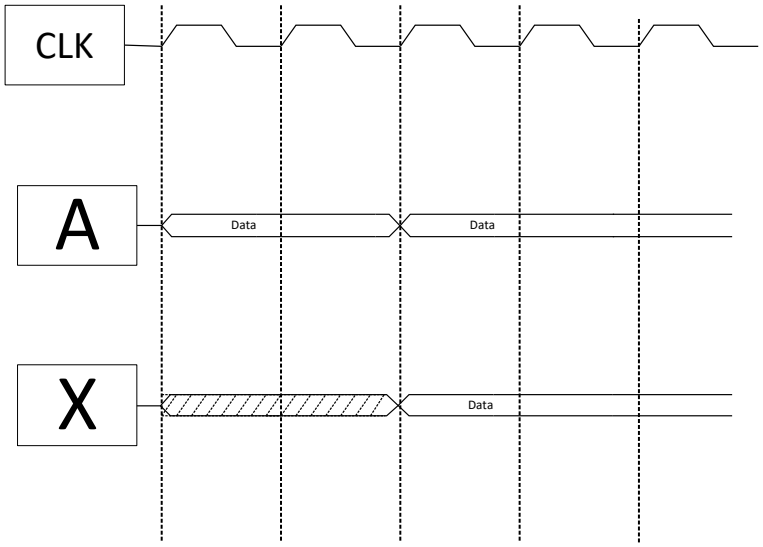
- ***DCT 1D 8 point***: bao gồm các tín hiệu đầu vào và ra từ A0 đến A7, mỗi tín hiệu có độ dài là 20bit bao gồm 1bit dấu, 11bit nguyên và 8bit thập phân.
- ***DCT 1D 8 point***: bao gồm các tín hiệu đầu vào và ra từ A8 đến A15, mỗi tín hiệu có độ dài là 20bit bao gồm 1bit dấu, 11bit nguyên và 8bit thập phân.
- ***DCT 1D 8 point***: bao gồm các tín hiệu đầu vào và ra từ A16 đến A23, mỗi tín hiệu có độ dài là 20bit bao gồm 1bit dấu, 11bit nguyên và 8bit thập phân.

- ***DCT 1D 8 point***: bao gồm các tín hiệu đầu vào và ra từ A24 đến A31, mỗi tín hiệu có độ dài là 20bit bao gồm 1bit dấu, 11bit nguyên và 8bit thập phân.
- ***DCT 1D 8 point***: bao gồm các tín hiệu đầu vào và ra từ A32 đến A39, mỗi tín hiệu có độ dài là 20bit bao gồm 1bit dấu, 11bit nguyên và 8bit thập phân.
- ***DCT 1D 8 point***: bao gồm các tín hiệu đầu vào và ra từ A40 đến A47, mỗi tín hiệu có độ dài là 20bit bao gồm 1bit dấu, 11bit nguyên và 8bit thập phân.
- ***DCT 1D 8 point***: bao gồm các tín hiệu đầu vào và ra từ A48 đến A55, mỗi tín hiệu có độ dài là 20bit bao gồm 1bit dấu, 11bit nguyên và 8bit thập phân.
- ***DCT 1D 8 point***: bao gồm các tín hiệu đầu vào và ra từ A56 đến A63, mỗi tín hiệu có độ dài là 20bit bao gồm 1bit dấu, 11bit nguyên và 8bit thập phân.



Hình 2.4 Thiết kế chi tiết DCT 1D

b) Mô tả Timming



Hình 2.5 Mô tả timming ghép khối DCT 1D và bộ đệm

Để tính timming khi tín hiệu xuất ra từ khối DCT 1D đi vào bộ đệm xuất ra tín hiệu sẽ mất 3 xung nhịp tín hiệu sẽ xuất ra.

2.3.3 Thiết kế chi tiết DCT 8 điểm

a) Thiết kế

Bao gồm 8 khối nhỏ được tính toán bên trong mỗi khối sẽ được tính theo công thức như sau:

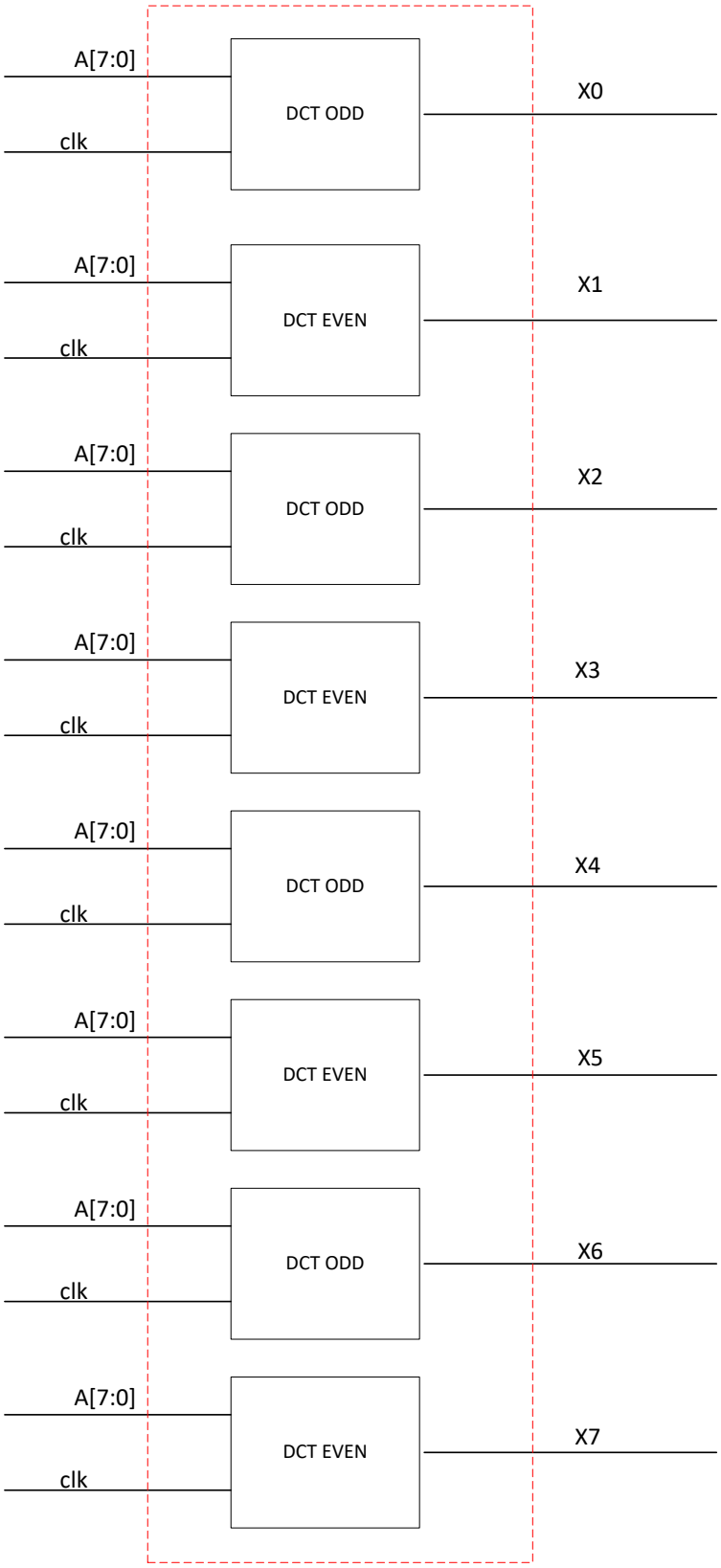
Với u lẻ:

$$x(u)=\frac{C(u)}{2}\times\left[(a_0+a_7)\cos(\frac{\pi u}{16})+(a_1+a_6)\cos(\frac{3\pi u}{16})+(a_2+a_5)\cos(\frac{5\pi u}{16})+(a_3+a_4)\cos(\frac{7\pi u}{16})\right]$$

Với u chẵn:

$$x(u)=\frac{C(u)}{2}\times\left[(a_0-a_7)\cos(\frac{\pi u}{16})+(a_1-a_6)\cos(\frac{3\pi u}{16})+(a_2-a_5)\cos(\frac{5\pi u}{16})+(a_3-a_4)\cos(\frac{7\pi u}{16})\right]$$

Trong đó: $C(u)=\frac{1}{\sqrt{2}}$ $(\forall u=0)$ với và $C(u)=1\,(\forall u\neq 0)$

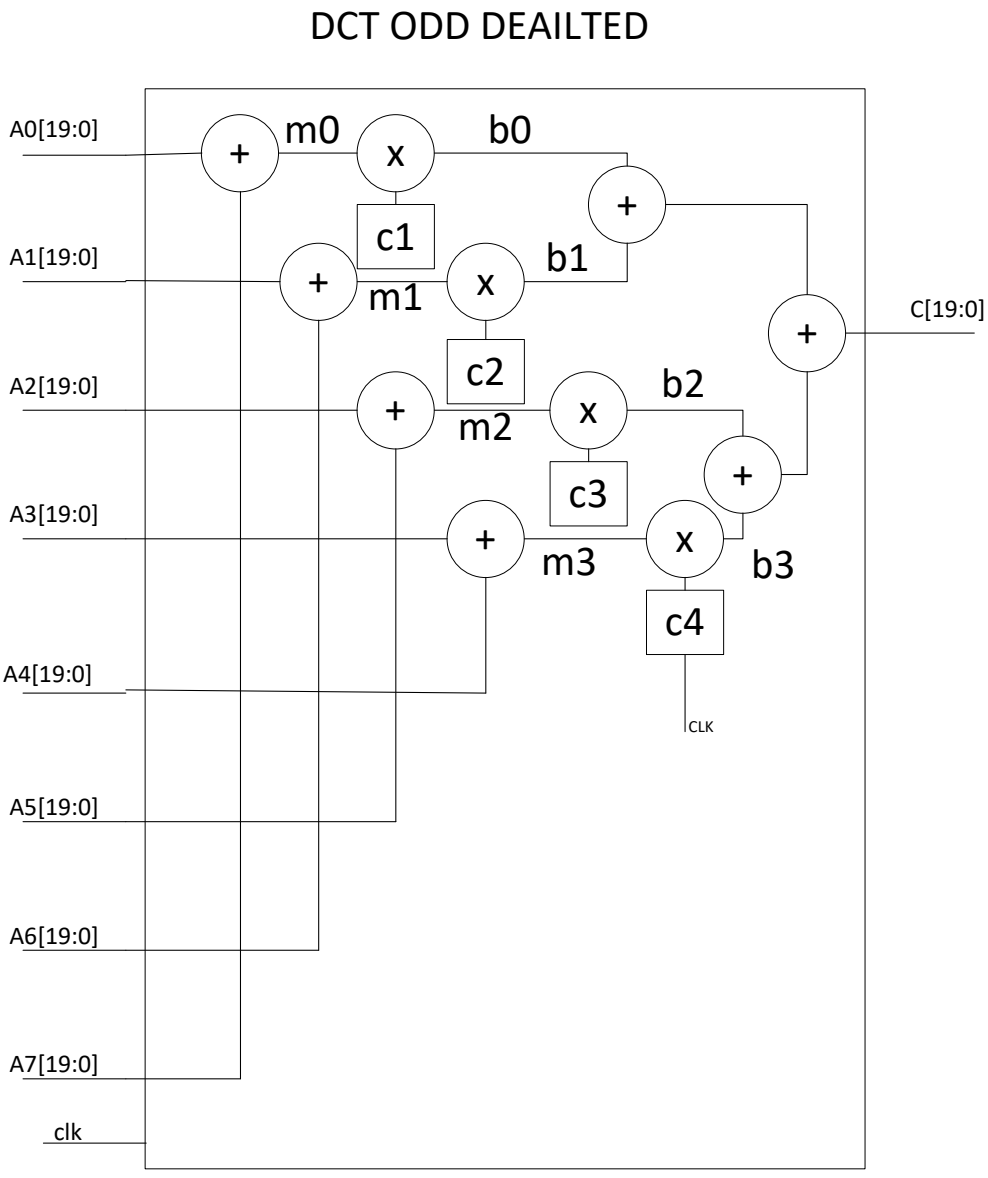


Hình 2.6 Thiết kế top DCT 8 điểm.

b) Thiết kế chi tiết DCT 8 điểm với u chẵn

Áp dụng thiết kế từ công thức:

$$x(u) = \frac{C(u)}{2} \times \left[(a_0 + a_7)\cos(\frac{\pi u}{16}) + (a_1 + a_6)\cos(\frac{3\pi u}{16}) + (a_2 + a_5)\cos(\frac{5\pi u}{16}) + (a_3 + a_4)\cos(\frac{7\pi u}{16}) \right]$$



Hình 2.7 Thiết kế DCT 8 điểm với u chẵn

Trong thiết kế bao gồm 1 bộ nhân cộng tính toán cộng các giá trị đầu vào với nhau theo công thức được một giá trị sẽ nhân giá trị đó với hàm cos đã được tạo sẵn

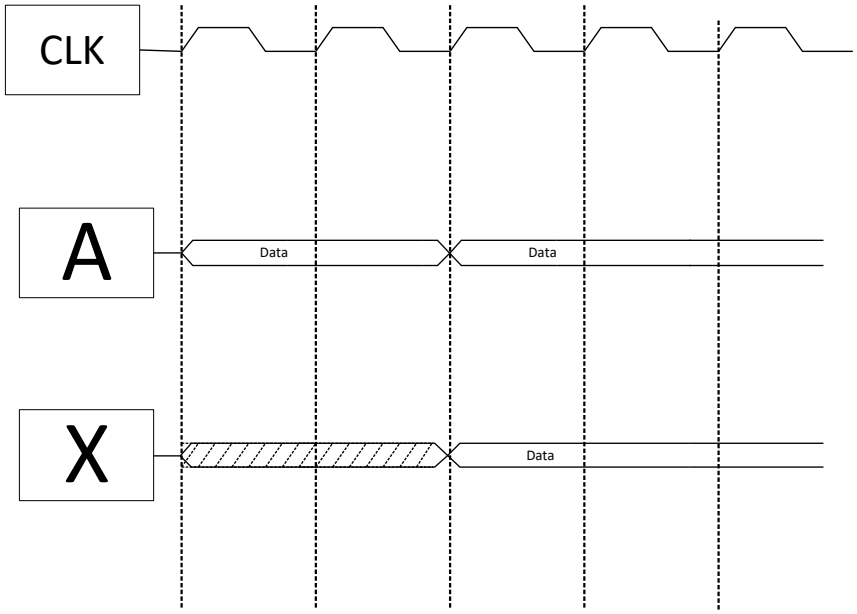
trong ROM khởi tạo để xuất các tín hiệu ra theo hàm cos và cuối cùng các tín hiếu sẽ được tổng hợp lại tại đầu ra theo công thức.

Mô tả tín hiệu vào ra khối DCT 8 điểm với u chẵn:

Bảng 2.2 Mô tả tín hiệu vào ra DCT 8 điểm với U chẵn

tín hiệu	kết nối	I/O	mô tả
Ai[19:0]	đầu vào	I	Giá trị một điểm ảnh
Ck[19:0]	x	O	tín hiệu đầu ra

Mô tả timing DCT 8 điểm: DCT 8 điểm sẽ thực hiện trong hai chu kỳ xung nhịp, chu kỳ đầu tiên DCT 8 điểm sẽ thực hiện đưa giá trị từ ngoài vào và công, chu kỳ thứ 2 DCT 8 điểm sẽ thực hiện nhân giá trị COSIN lấy từ khối ROM ra.



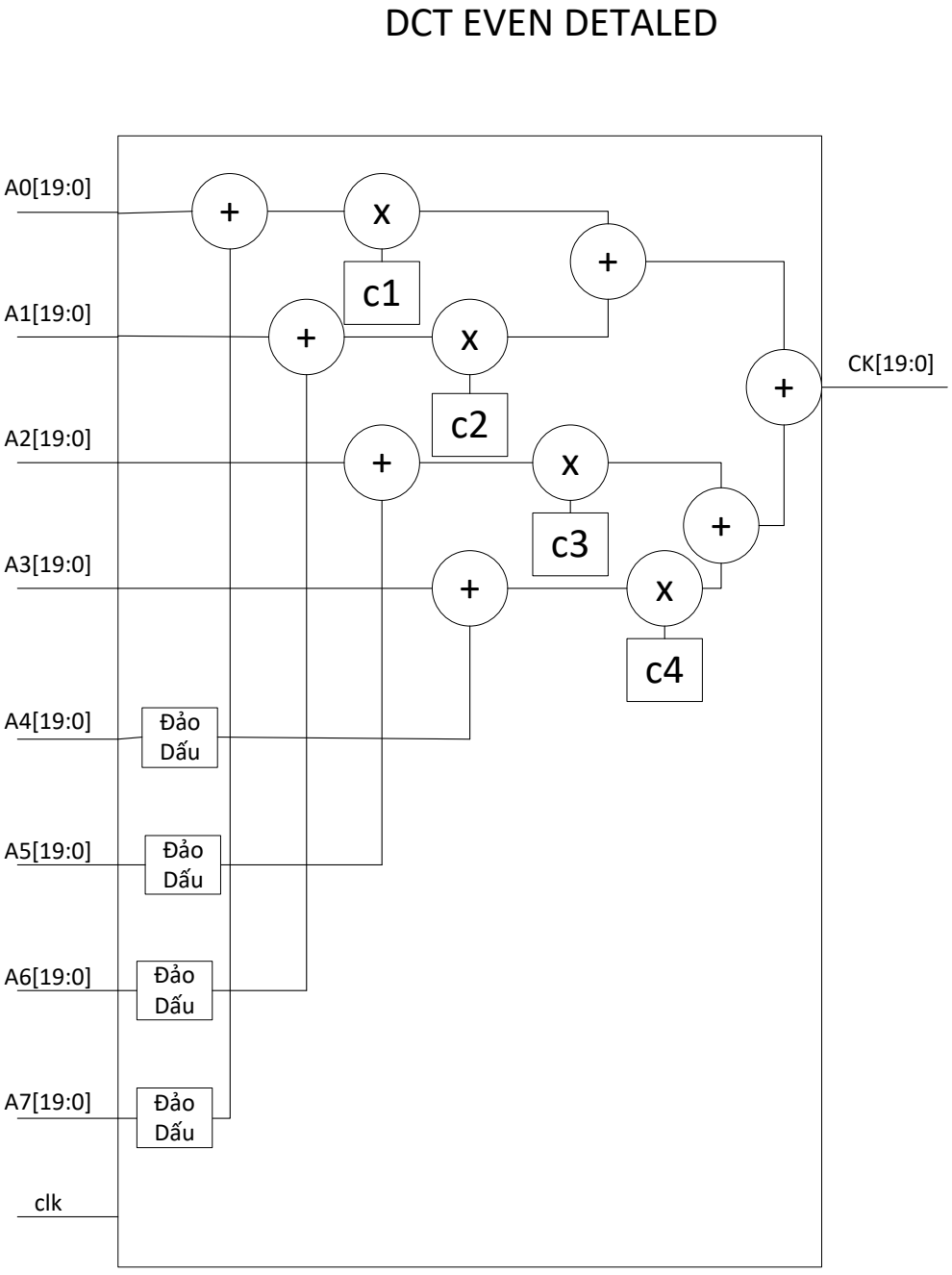
Hình 2.8 Timming cho DCT 1D 8 điểm

c) Thiết kế DCT 1D 8 điểm với U lẻ

Áp dụng với công thức:

$$x(u) = \frac{C(u)}{2} \times \left[(a_0 - a_7)\cos(\frac{\pi u}{16}) + (a_1 - a_6)\cos(\frac{3\pi u}{16}) + (a_2 - a_5)\cos(\frac{5\pi u}{16}) + (a_3 - a_4)\cos(\frac{7\pi u}{16}) \right]$$

Khối DCT 8 điểm có u lẻ được thiết kế tượng tự như khối DCT 8 điểm có u chẵn nhưng do trong công thức có thêm giá trị trì nên trong thiết kế sẽ có thêm tín hiệu trừ nhưng thay vì trữ sẽ dùng là cộng với số đối của số trừ chính là bộ đảo như trong hình vẽ.



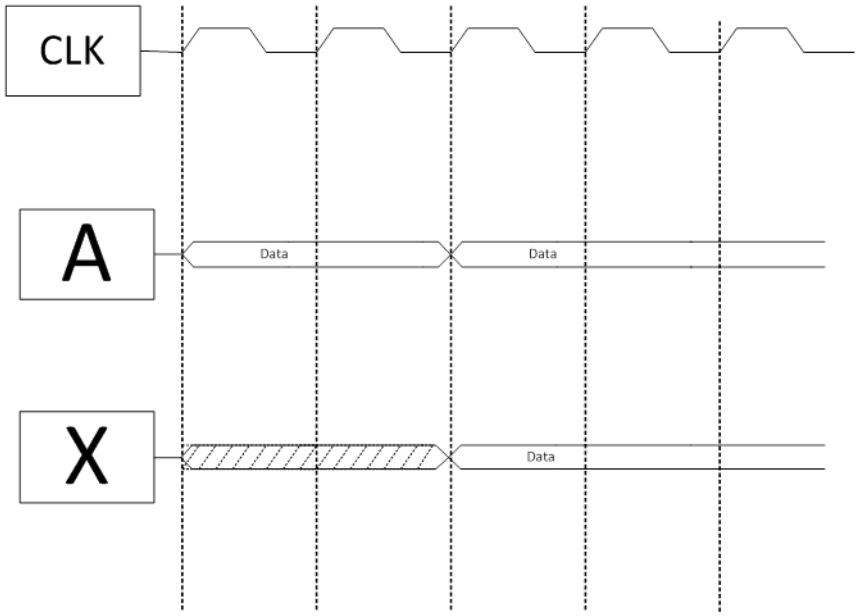
Hình 2.9 Thiết kế chi tiết DCT 8 điểm với U lẻ

Tín hiệu vào ra khối DCT 8 điểm với U lẻ

Bảng 2.3 Mô tả tín hiệu vào ra của DCT 8 điểm với U lẻ

tín hiệu	kết nối	I/O	mô tả
Ai[19:0]	Y	I	Giá trị một điểm ảnh
Ck[19:0]	F	O	đầu ra tín hiệu

Timming DCT 8 điểm với u lẻ tương tự với DCT 8 điểm với u chẵn cũng thực hiện trong cùng 2 chu kỳ xung nhịp.

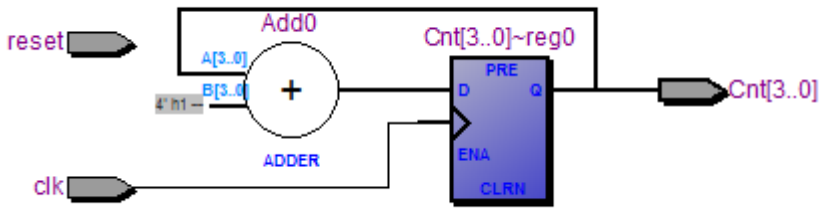


Hình 2.10 Mô tả Timming của DCT 8 điểm với U lẻ

CHƯƠNG 3:THỰC THI VÀ KIỂM THỬ TỪNG KHỐI

3.1 Khối đếm xung CLK.

Nhiệm vụ của khối xung CLK là thực hiện đếm xung CLK theo sườn dương của CLK để các công việc được thực hiện tuần tự và chính xác.



Hình 3.1 Mạch tổng hợp của khối đếm xung CLK

Trong thiết kế tổng thể, khối DCT 1D thứ nhất hoạt động ở xung CLK thực nhất và DCT 1D thữ 2 hoạt động ở xung CLK thứ 2.

3.2 Khối DCT ODD.

3.2.1 Thực thi khối

Như phân tích trong phần **BIẾN ĐỔI CÔNG THỨC** trong khâu thiết kế, công thức DCT 1 chiều với 8 điểm được chia thành trường hợp và được tổng hợp dưới đây:

CÔNG THỨC

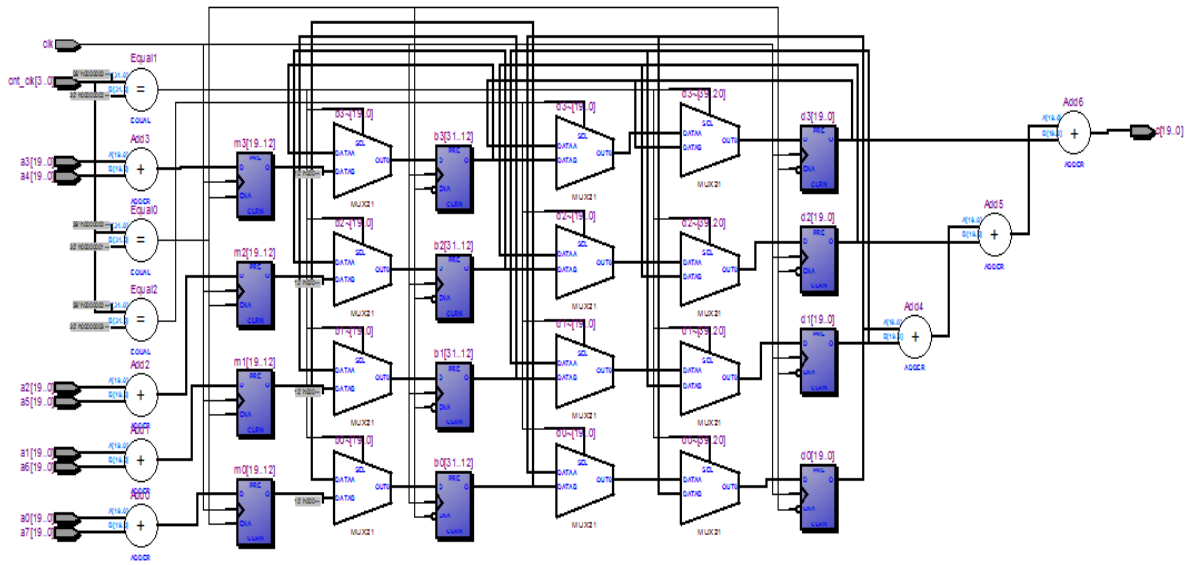
$$x(u)=\frac{C(u)}{2}\times\left[(a_0+a_7)\cos(\frac{\pi u}{16})+(a_1+a_6)\cos(\frac{3\pi u}{16})+(a_2+a_5)\cos(\frac{5\pi u}{16})+(a_3+a_4)\cos(\frac{7\pi u}{16})\right]$$ Với u chẵn

$$x(u)=\frac{C(u)}{2}\times\left[(a_0-a_7)\cos(\frac{\pi u}{16})+(a_1-a_6)\cos(\frac{3\pi u}{16})+(a_2-a_5)\cos(\frac{5\pi u}{16})+(a_3-a_4)\cos(\frac{7\pi u}{16})\right]$$ Với u lẻ

$C(u)=1/\sqrt{2}$ Với u = 0

$C(u)=1$ Với u != 0

Khối DCT ODD thực hiện việc tính toán công thức DCT trong các trường hợp u chẵn.

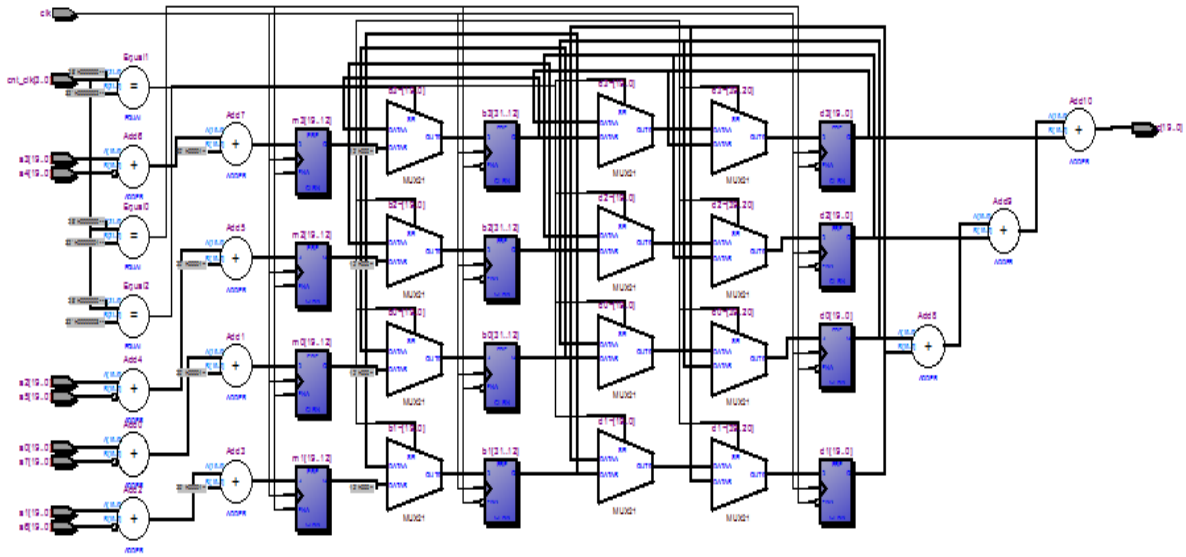


Hình 3.2 Mạch tổng hợp DCT ODD

Giá trị điểm ảnh đầu vào được biểu diễn bởi số 20 bit trong đó có 12 bit phần nguyên và 8 bit phần thập phân, bit đầu tiên là bit dấu. Việc tính toán được thực hiện như mô tả trong phần thiết kế, nhưng sau phép nhân hai số 20bit, kết quả thu được là một số 40 bit, do đó cần phải chuyển đổi về số 20bit. [2]

3.3 Khối DCT EVEN.

Khối DCT EVEN thực hiện công thưc DCT một chiều 8 điểm trong các trường hợp u lẻ.



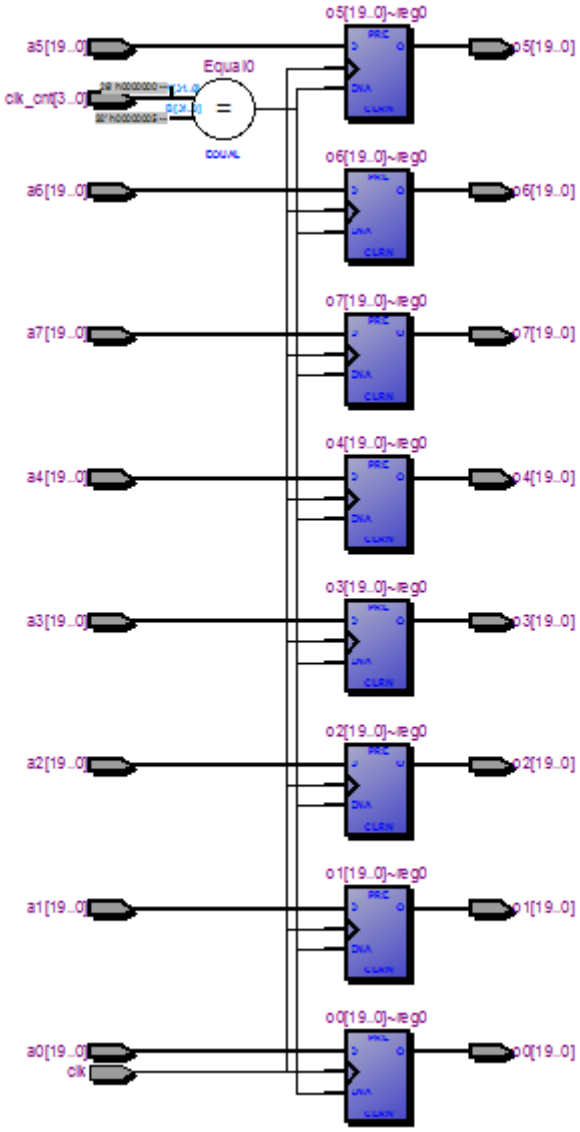
Hình 3.3 Mạch tổ hợp khối DCT EVEN

Việc thực hiện phép toán trừ trong khối DCT EVEN được thực hiện bằng cách đảo dấu và cộng thêm bit 1 và sau đó thực hiện phép cộng bit.

3.4 Khối Buffer

Chức năng của khối này là thực hiện việc chuyển đổi hàng thành cột của ảnh sau khối DCT 1D để thực hiện tiếp khối DCT 1D sau đó.

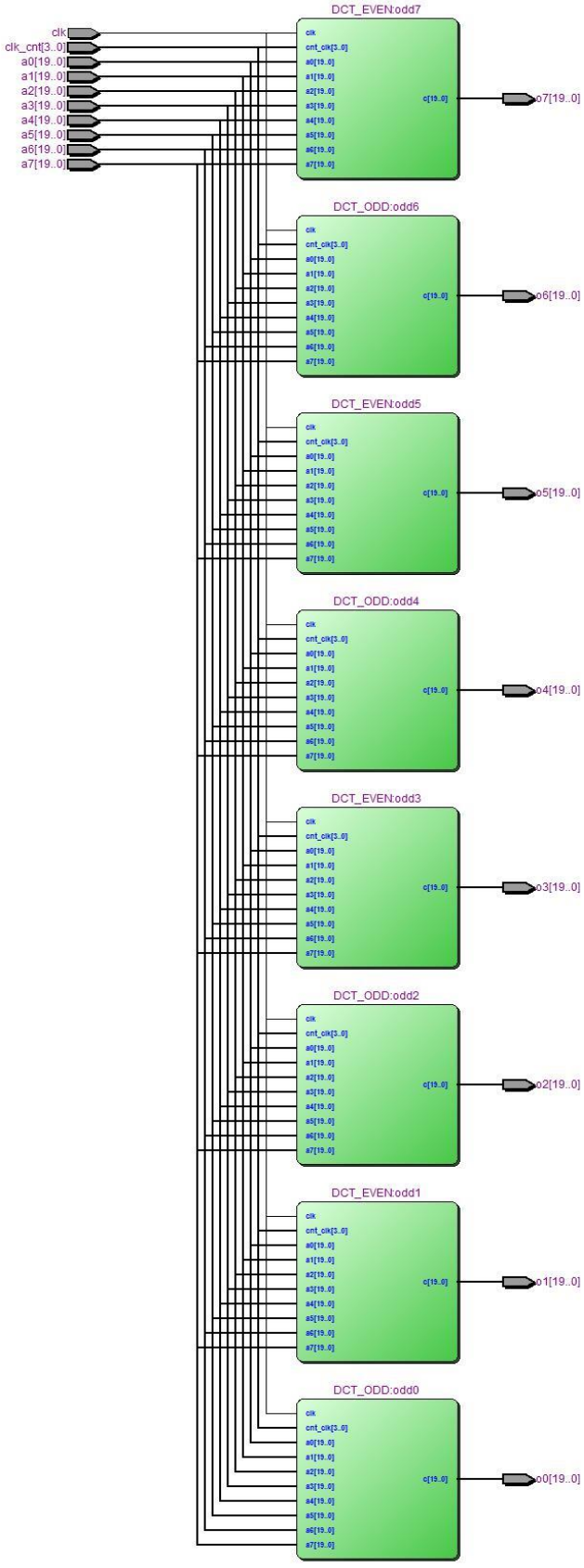
Khối Buffer mỗi lần chuyển đổi 8 tín hiệu.



Hình 3.4 Mạch tổ hợp của khối Buffer

3.5 Khối DCT 1D 8 POINT

Chức năng của khối DCT 1D 8 POINT là thực hiện công thức DCT với đầu vào 8 điểm ảnh và đầu ra là 8 giá trị đã được tính toán.



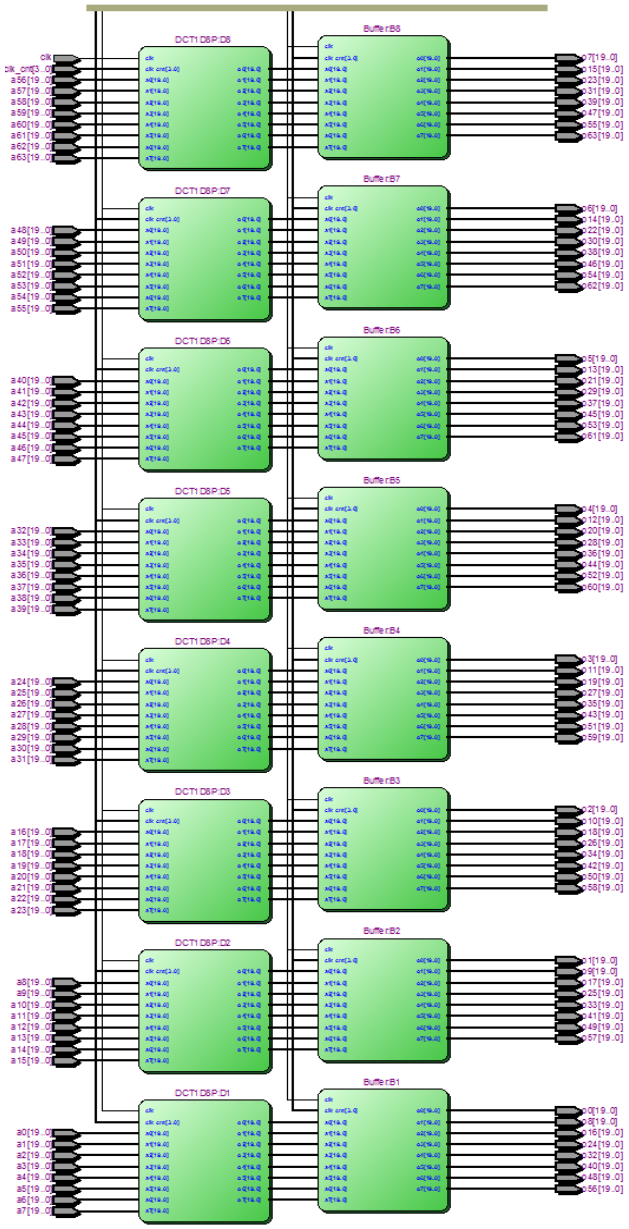
Hình 3.5 Mạch tổ hợp khối DCT 1D 8 POINT

3.6 Khối DCT 1D 64 POINT

Chức năng của khối DCT 1D 64 POINT là thực hiện công thức DCT theo hàng của một khối ảnh pixel 8x8.

Đầu vào của khối là các giá trị a_0 -> a_{63}, đầu ra là các giá trị đã được tính theo công thưc DCT.

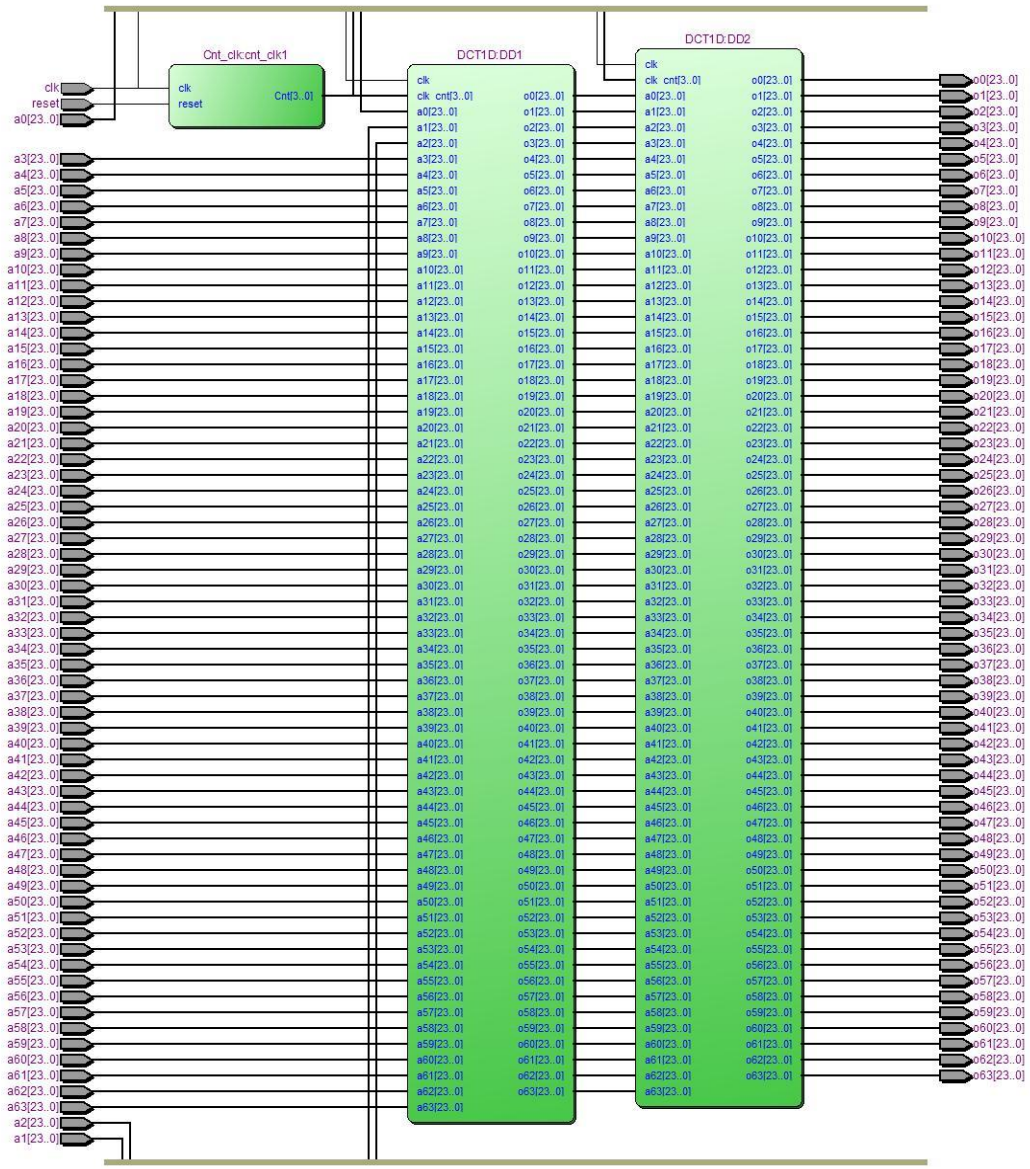
Khối DCT 1D 64 POINT bao gồm 8 khối DCT 1D 8 POINT và 8 Buffer để chuyển đổi hàng thành cột.



Hình 3.6 Mạch tổ hợp khối DCT 1D 64 point

3.7 Khối tổng DCT 2D

Khối DCT 2D bao gồm hai khối DCT 1D 64 POINT nối tiếp với nhau để thực hiện chức DCT 2 chiều một ảnh pixel 8x8.



Hình 3.7 Mạch tổ hợp của khối DCT 2D

.

3.8 Kiểm thử

Để thực hiện kiểm thử, nhóm sử dụng bộ dữ liệu mẫu sau:

130	132	132	129	133	133	134	135
135	133	133	131	133	137	138	136
132	134	133	136	139	142	137	137
137	136	136	135	135	136	135	138
139	138	139	135	136	139	137	140
134	136	137	136	134	136	136	143
137	133	138	136	136	136	139	146
139	137	138	134	133	138	140	146

DCT →

1089.1	-10.4	8.1	-2.4	-0.4	-2.2	4.9	0
-12.5	0.6	-5.9	6.3	-2.9	-1.0	-2.8	0.4
-4.9	3.8	3.9	-1.2	-0.5	0.9	-0.3	1.8
-4.2	1.2	0.7	-3.5	0.3	-0.8	0	1.3
0.6	4.2	2.7	-0.4	-1.1	-0.8	0.4	0.8
2.6	-1.2	-2.4	-1.9	2.9	-0.6	-0.5	0.8
1.1	-1.7	-3.1	-0.4	-0.8	-2.5	-0.9	2.3
3.1	-1.7	-2.6	2.0	-0.1	-0.7	0.6	0.5

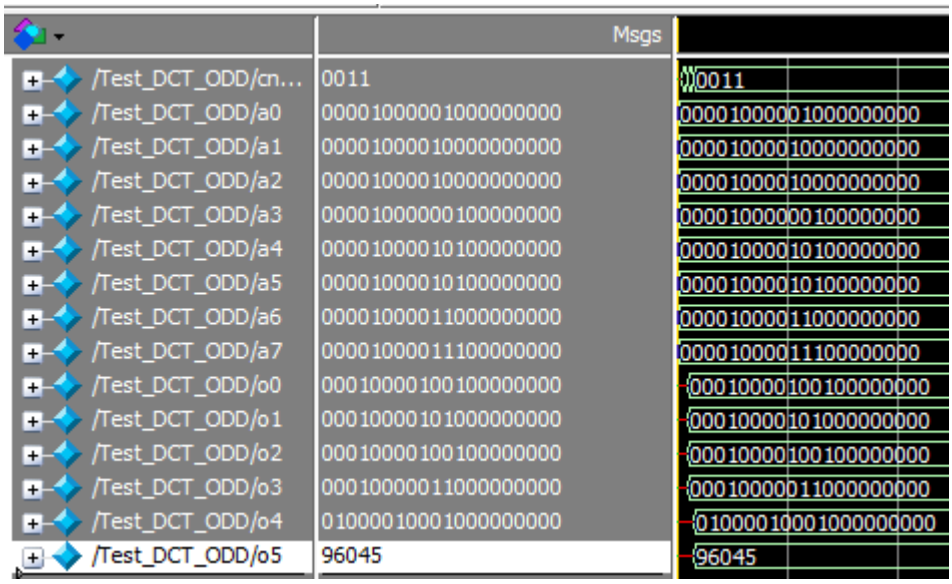
Để kiểm tra các module nhỏ với 8 điểm ảnh, nhóm sử dụng hàng đầu tiên của khối ảnh pixel 8x8 ở trên.

130	132	132	129	133	133	134	135

Gía trị DCT 1 chiều 8 điểm của 8 điểm trên là:

374.06	-3.95	1.58	-0.28	-1.41	-2.17	0.112	1.615

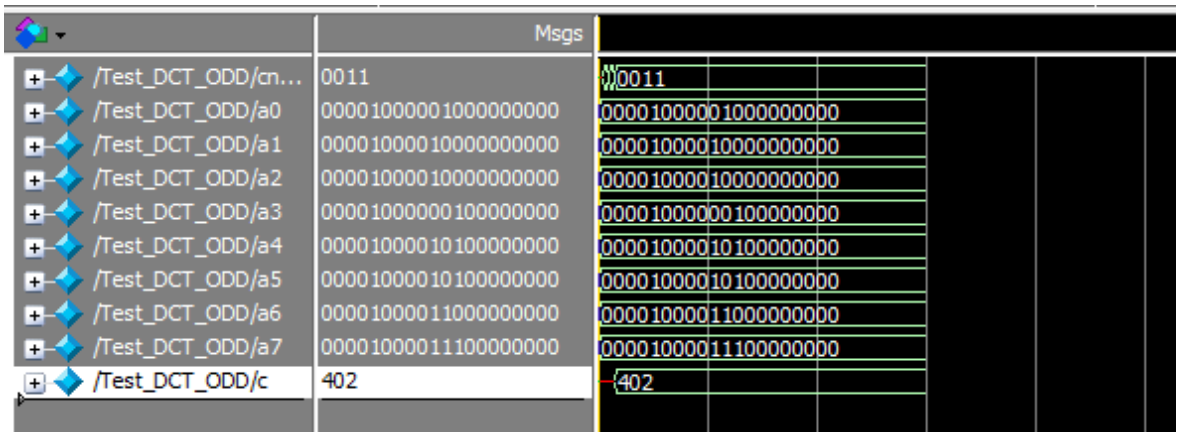
3.8.1 Kiểm thử module DCT_ODD



Hình 3.8 Kiểm thử module DCT_ODD vị trí thứ nhất

Hình ảnh trên thể hiện kết quả của việc tính toán DCT vị trí thứ 0 của 8 điểm ảnh phía trên. Theo quy ước trong phần thiết, mỗi điểm ảnh gồm 24bit trong đó có 16bit là phần nguyên và 8bit đại diện cho phần thập phân. Vậy nên để xác định kết

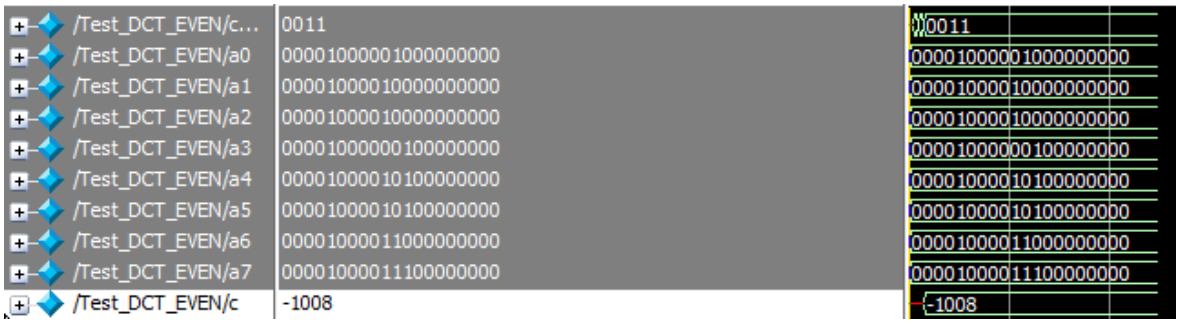
quả ta thực hiện $\frac{96045}{2^8} = 375.176$, giá trị này gần bằng so với giá trị tính toán lý thuyết.



Hình 3.9 Kiểm thử module DCT_ODD vị trí thứ 2

Hình ảnh trên thể hiện kết quả của DCT 1 chiều 8 điểm ở vị trí thứ 2. Tương tự, để thu được kết quả ta thực hiện lấy $\frac{402}{2^8} = 1.57$. Giá trị này là gần đúng với giá trị tính toán theo lý thuyết.

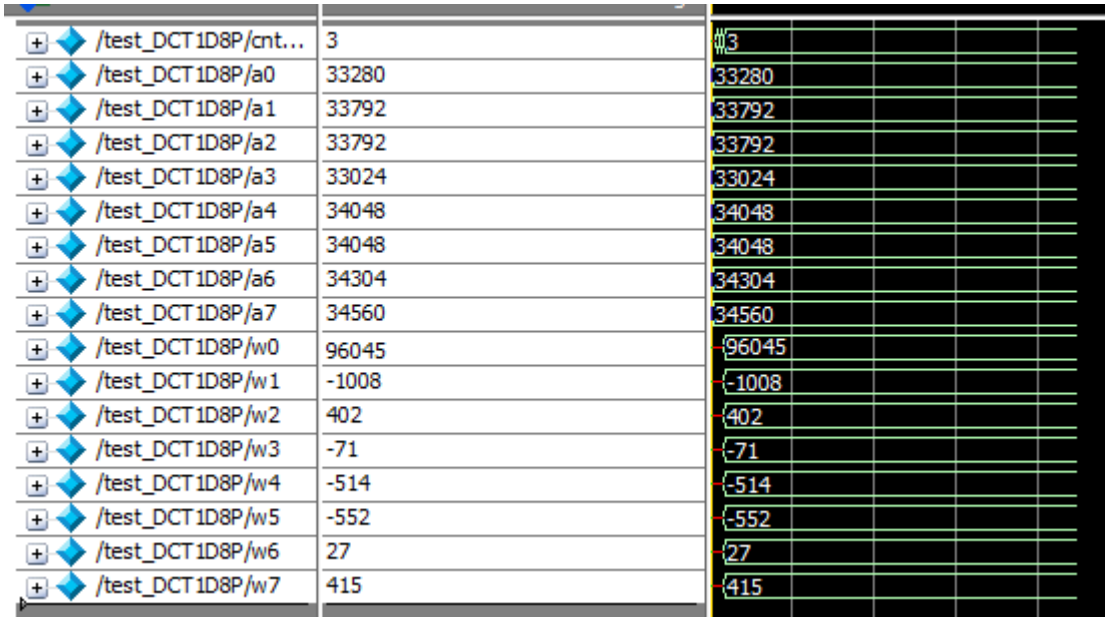
3.8.2 Kiểm thử khối DCT_EVEN



Hình 3.10 Kiểm thử khối DCT_EVEN

Hình ảnh trên thể hiện kết quả của DCT 1 chiều 8 điểm ở vị trí thứ 1. Tương tự, để thu được kết quả ta thực hiện lấy -1008/2^8 = -3.9375. Giá trị này là gần đúng với giá trị tính toán theo lý thuyết.

3.8.3 Kiểm thử khối DCT1D8P



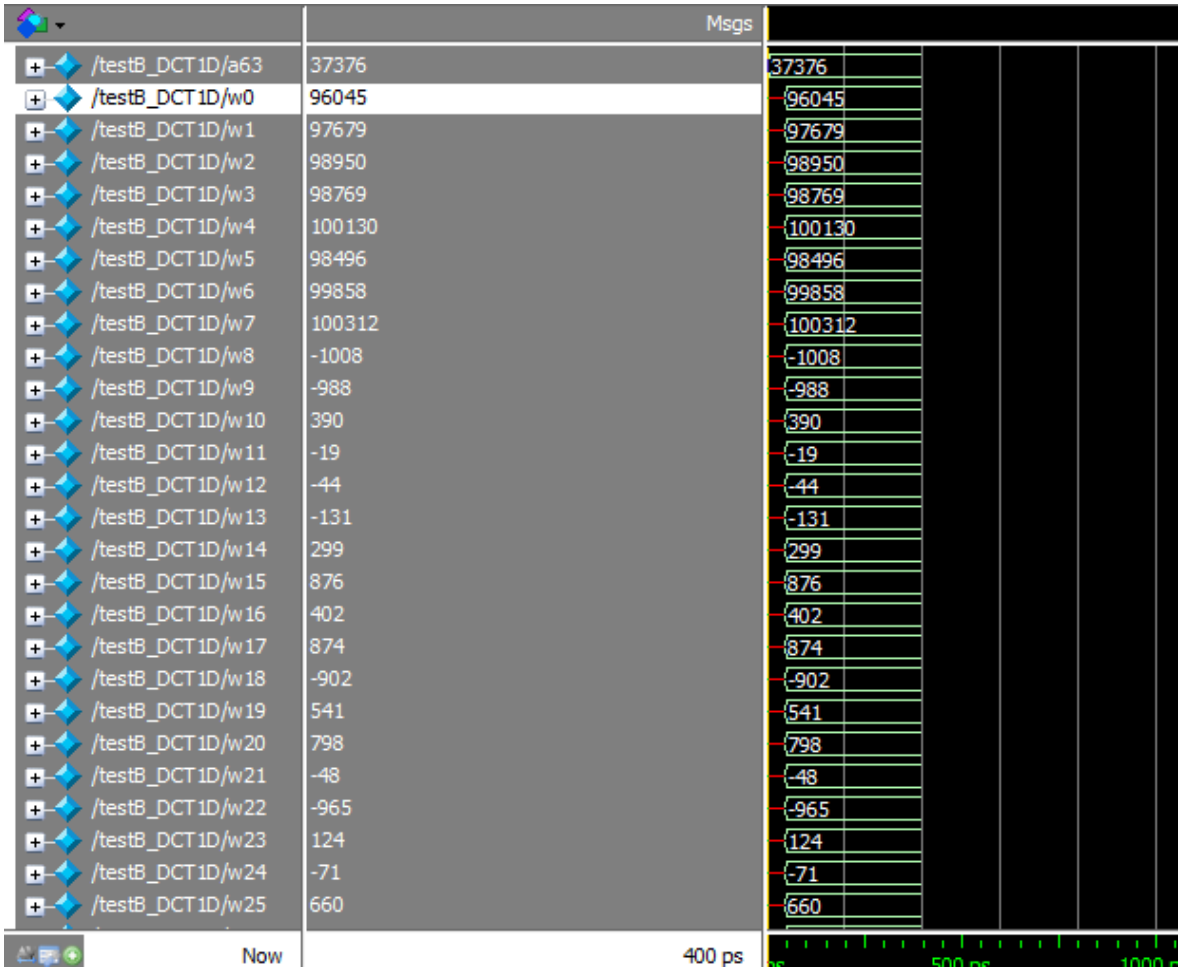
Hình 3.11 Kiểm thử khối DCT1D8P

Hình ảnh trên thể hiện kết quả của DCT 1 chiều 8 điểm của cả 8 điểm ảnh. Tương tự, để thu được kết quả ta thực hiện:

Bảng 3.1 Bảng kết quả DCT 1 chiều 8 điểm

$\frac{96045}{2^8}$	375.176
$\frac{-1008}{2^8}$	-3.9375
$\frac{402}{2^8}$	1.57
$\frac{-71}{2^8}$	-2.277
$\frac{-514}{2^8}$	-2.00
$\frac{-552}{2^8}$	-2.16
$\frac{27}{2^8}$	0.105
$\frac{415}{2^8}$	1.621

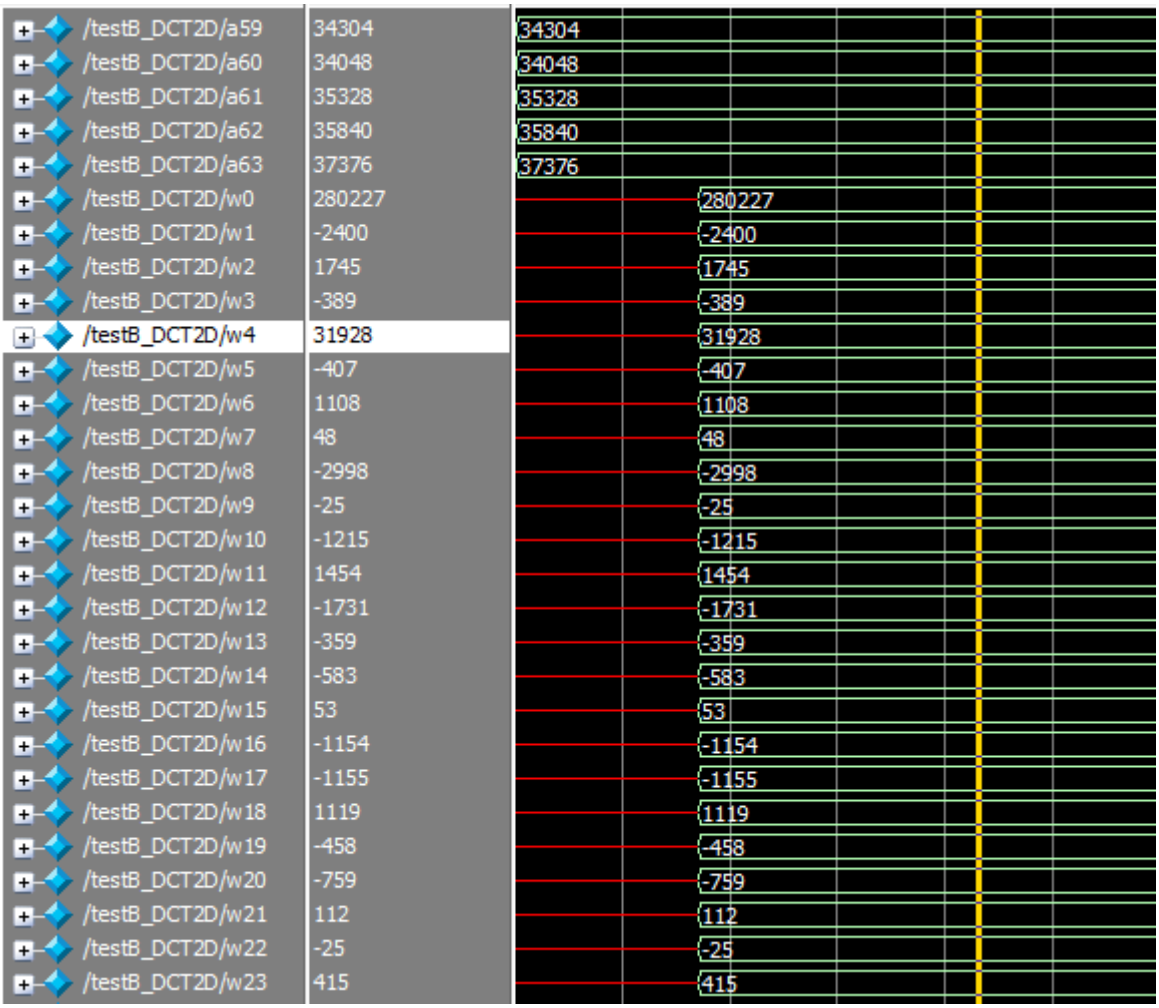
3.8.4 Kiểm thử khối DCT1D



Hình 3.12 Kiểm thử khối DCT1D

Hình ảnh trên thể hiện kết quả của DCT 1 chiều 64 điểm theo chiều ngang của khối ảnh trên.

3.8.5 Kiểm thử khối DCT2D



Hình 3.13 Kiểm thử khối DCT2D

Hình ảnh trên thể hiện kết quả của DCT 2 chiều 64 điểm của khối ảnh trên. Để kiểm tra ta lấy kết quả chia với 2^8. Ví dụ $\frac{280277}{2^8}$ = 1094.83, kết quả là gần bằng với kết quả tính toán lý thuyết.

CHƯƠNG 4: KẾT LUẬN

Trong quá trình thực hiện bài tập lớn, chúng em đã có cơ hội tìm hiểu về DCT giúp giảm tải trong quá trình truyền, thực hiện thiết kế từng khối để thực hiện kĩ thuật DCT 64 điểm và thực hiện lập trình Verilog để mô tả hệ thống.

Qua bài tập lớn chúng em đã học được rất nhiều về việc thiết kế một hệ thống, tuy còn nhiều thiếu sót do thiếu kinh nghiệm nhưng cơ bản chúng em đã hiểu các khâu trong việc phân tích và thiết kế một hệ thống. Đây sẽ là kiến thức nền tảng giúp chúng em trong quá trình học tập, nghiên cứu và phát triển sau này.

TÀI LIỆU THAM KHẢO

[1] K. K. Mahapatra, "Design of 2D Discrete Cosine Transform Using Cordic Architectures in Vhdl Design of 2D Discrete Cosine Transform Using Cordic Architectures in Vhdl," 2007.

[2] "Verilog căn bản: Giới thiệu về ngôn ngữ verilog | Vi mạch - Diễn đàn Vi Mạch, Hệ Thống Nhúng, Trí Tuệ Nhân Tạo." [Online]. Available: http://vimach.net/threads/verilog-can-ban-gioi-thieu-ve-ngon-ngu-verilog.82/?fbclid=IwAR2RbN3OaaFFPS3lbgi1ScV8_4eXgQiyeg2vmO0DMBNL4l3of0dPCfXuegk. [Accessed: 23-Dec-2018].

PHỤ LỤC

1. Code khối đếm xung CLK

```
module Cnt_clk
#(
  parameter SIZE_CNT
)
(
  input clk,
  input reset,
  output reg [SIZE_CNT:0]Cnt
);
initial
begin
 Cnt = 4'b0000;
end
always @(posedge clk)
begin
    Cnt = Cnt+1;
end
     endmodule
```

2. Code khối DCT ODD

```
  module DCT_ODD
#(
  parameter c1,
  parameter c2,
  parameter c3,
  parameter c4,
  parameter CNT_CLK,
  parameter cu
)
```

```
(
  input clk,
  input [3:0] cnt_clk,
  input [23:0] a0, a1, a2, a3, a4, a5, a6, a7,
  output [23:0] c
);

localparam size_cnt = 3;
wire [23:0] w0,w1,w2,w3;
reg [23:0] m0, m1, m2, m3;
reg [23:0] b0, b1, b2, b3;
reg [23:0] d0, d1, d2, d3;

always @(posedge clk)
begin
  if(cnt_clk == CNT_CLK)
  begin
      m0 = a0 + a7;
      m1 = a1 + a6;
      m2 = a2 + a5;
      m3 = a3 + a4;
  end
else if (cnt_clk == CNT_CLK + 1)
  begin
  b0 = m0 * c1;
  b1 = m1 * c2;
  b2 = m2 * c3;
  b3 = m3 * c4;
  end
  else if(cnt_clk == CNT_CLK + 2)
```

```
    begin
        d0 = b0 + b1 + b2 + b3;
        if (d0[23] == 1'b1)
        begin
            d1 = ~d0 + 1'b1;
            d2 = d1 / cu;
            d3 = ~d2 + 1'b1;
        end
        else if(d0[23] == 1'b0)
        begin
            d3 = d0 / cu;
        end
    end
end
assign c = d3;
endmodule
```

3. Code khối DCT EVEN

```
module DCT_EVEN
#(
  parameter c1,
  parameter c2,
  parameter c3,
  parameter c4,
  parameter CNT_CLK,
  parameter cu
)
(
  input clk,
```

```
  input [3:0] cnt_clk,
  input [23:0] a0, a1, a2, a3, a4, a5, a6, a7,
  output [23:0] c
);
localparam size_cnt = 3;

wire [23:0] w0,w1,w2,w3;
reg [23:0] m0, m1, m2, m3;
reg [23:0] b0, b1, b2, b3;
reg [23:0] d0, d1, d2, d3;



always @(posedge clk)
begin
  if(cnt_clk == CNT_CLK)
  begin
     m0 = a0 + ~a7 + 1'b1;
     m1 = a1 + ~a6 + 1'b1;
     m2 = a2 + ~a5 + 1'b1;
     m3 = a3 + ~a4 + 1'b1;
  end
else if (cnt_clk == CNT_CLK + 1)
  begin
  b0 = m0 * c1;
  b1 = m1 * c2;
  b2 = m2 * c3;
  b3 = m3 * c4;
  end
  else if(cnt_clk == CNT_CLK + 2)
  begin
```

```
        d0 = b0 + b1 + b2 + b3;
        if (d0[23] == 1'b1)
        begin
            d1 = ~d0 + 1'b1;
            d2 = d1 / cu;
            d3 = ~d2 + 1'b1;
        end
        else if(d0[23] == 1'b0)
        begin
            d3 = d0 / cu;
        end
    end
end

assign c = d3;
endmodule
```

4. Code khối Buffer

```
module Buffer
#(
  parameter CNT_CLK
)
(
  input clk,
  input [3:0] clk_cnt,
  input [19:0] a0, a1, a2, a3, a4, a5, a6, a7,
  output reg [19:0] o0, o1, o2, o3, o4, o5, o6, o7
);
```

```
always@(posedge clk)
begin
  if(clk_cnt == CNT_CLK)
  begin
      o0 = a0;
      o1 = a1;
      o2 = a2;
      o3 = a3;
      o4 = a4;
      o5 = a5;
      o6 = a6;
      o7 = a7;
  end
end
endmodule
```

5. Code khối DCT 1D 8 POINT

```
module DCT1D8P
#(
  parameter CNT_Clk
)
(
  input clk,
  input [3:0] clk_cnt,
  input [23:0] a0, a1, a2, a3, a4, a5, a6, a7,
  output wire [23:0] o0, o1, o2, o3, o4, o5, o6, o7
);
localparam c0 = 24'b000000000000000000000001;
localparam c1 = 24'b000000000000000000000001;
```

```
localparam c2 = 24'b000000000000000000000001;
localparam c3 = 24'b000000000000000000000001;
localparam c4 = 24'b000000000000000011111011;
localparam c5 = 24'b000000000000000011010100;
localparam c6 = 24'b000000000000000010001110;
localparam c7 = 24'b000000000000000000110001;
localparam c8 = 24'b000000000000000011101100;
localparam c9 = 24'b000000000000000001100001;
localparam c10 = 24'b111111111111111110011111;
localparam c11 = 24'b111111111111111100010100;
localparam c12 = 24'b000000000000000011010100;
localparam c13 = 24'b111111111111111111001111;
localparam c14 =24'b111111111111111100000101 ;
localparam c15 = 24'b111111111111111101110010;
localparam c16 = 24'b000000000000000010110101;
localparam c17 = 24'b111111111111111101001011;
localparam c18 =24'b000000000000000010110101;
localparam c19 =  24'b000000000000000010001110;
localparam c20 = 24'b000000000000000010001110;
localparam c21 = 24'b111111111111111100000101;
localparam c22 =24'b000000000000000000110001;
localparam c23 =  24'b000000000000000011010100;
localparam c24 = 24'b000000000000000001100001;
localparam c25 = 24'b111111111111111100010100;
localparam c26 =24'b000000000000000011101100;
localparam c27 =  24'b111111111111111110011111;
localparam c28 = 24'b000000000000000000110001;
localparam c29 = 24'b111111111111111101110010;
localparam c30 =24'b000000000000000011010100;
localparam c31 =  24'b111111111111111100000101;
```

```
localparam cu0 = 2.82, cu = 512;

wire [23:0] w0, w1, w2, w3, w4, w5, w6, w7;

DCT_ODD
#(.c1(c0),.c2(c1),.c3(c2),.c4(c3),.CNT_CLK(CNT_Clk),.cu(cu0))
odd0 (.clk(clk),.cnt_clk(clk_cnt),.a0(a0),.a1(a1),
.a2(a2),.a3(a3),.a4(a4),.a5(a5),.a6(a6),.a7(a7),.c(w0));

DCT_EVEN
#(.c1(c4),.c2(c5),.c3(c6),.c4(c7),.CNT_CLK(CNT_Clk),.cu(cu))
even0 (.clk(clk),.cnt_clk(clk_cnt),.a0(a0),.a1(a1),
.a2(a2),.a3(a3),.a4(a4),.a5(a5),.a6(a6),.a7(a7),.c(w1));

DCT_ODD
#(.c1(c8),.c2(c9),.c3(c10),.c4(c11),.CNT_CLK(CNT_Clk),.cu(cu))
odd1 (.clk(clk),.cnt_clk(clk_cnt),.a0(a0),.a1(a1),
.a2(a2),.a3(a3),.a4(a4),.a5(a5),.a6(a6),.a7(a7),.c(w2));

DCT_EVEN
#(.c1(c12),.c2(c13),.c3(c14),.c4(c15),.CNT_CLK(CNT_Clk),.cu(cu))
even1 (.clk(clk),.cnt_clk(clk_cnt),.a0(a0),.a1(a1),
.a2(a2),.a3(a3),.a4(a4),.a5(a5),.a6(a6),.a7(a7),.c(w3));

DCT_ODD
#(.c1(c16),.c2(c17),.c3(c18),.c4(c19),.CNT_CLK(CNT_Clk),.cu(cu))
odd2 (.clk(clk),.cnt_clk(clk_cnt),.a0(a0),.a1(a1),
.a2(a2),.a3(a3),.a4(a4),.a5(a5),.a6(a6),.a7(a7),.c(w4));

DCT_EVEN
```

```
#(.c1(c20),.c2(c21),.c3(c22),.c4(c23),.CNT_CLK(CNT_Clk),.cu(cu))
even2 (.clk(clk),.cnt_clk(clk_cnt),.a0(a0),.a1(a1),
.a2(a2),.a3(a3),.a4(a4),.a5(a5),.a6(a6),.a7(a7),.c(w5));

DCT_ODD
#(.c1(c24),.c2(c25),.c3(c26),.c4(c27),.CNT_CLK(CNT_Clk),.cu(cu))
odd3 (.clk(clk),.cnt_clk(clk_cnt),.a0(a0),.a1(a1),
.a2(a2),.a3(a3),.a4(a4),.a5(a5),.a6(a6),.a7(a7),.c(w6));

DCT_EVEN
#(.c1(c28),.c2(c29),.c3(c30),.c4(c31),.CNT_CLK(CNT_Clk),.cu(cu))
even3 (.clk(clk),.cnt_clk(clk_cnt),.a0(a0),.a1(a1),
.a2(a2),.a3(a3),.a4(a4),.a5(a5),.a6(a6),.a7(a7),.c(w7));

assign o0 = w0;
assign o1 = w1;
assign o2 = w2;
assign o3 = w3;
assign o4 = w4;
assign o5 = w5;
assign o6 = w6;
assign o7 = w7;
endmodule
```

6. Code khối DCT 1D

```
    module DCT1D
#(
  parameter CNT_CLK
)
```

```
(
  input clk,
  input [3:0] clk_cnt,
  input [23:0] a0,a1,a2,a3,a4,a5,a6,a7,
  input [23:0] a8,a9,a10,a11,a12,a13,a14,a15,
  input [23:0] a16,a17,a18,a19,a20,a21,a22,a23,
  input [23:0] a24,a25,a26,a27,a28,a29,a30,a31,
  input [23:0] a32,a33,a34,a35,a36,a37,a38,a39,
  input [23:0] a40,a41,a42,a43,a44,a45,a46,a47,
  input [23:0] a48,a49,a50,a51,a52,a53,a54,a55,
  input [23:0] a56,a57,a58,a59,a60,a61,a62,a63,
  output wire [23:0] o0,o1,o2,o3,o4,o5,o6,o7,
  output wire [23:0] o8,o9,o10,o11,o12,o13,o14,o15,
  output wire [23:0] o16,o17,o18,o19,o20,o21,o22,o23,
  output wire [23:0] o24,o25,o26,o27,o28,o29,o30,o31,
  output wire [23:0] o32,o33,o34,o35,o36,o37,o38,o39,
  output wire [23:0] o40,o41,o42,o43,o44,o45,o46,o47,
  output wire [23:0] o48,o49,o50,o51,o52,o53,o54,o55,
  output wire [23:0] o56,o57,o58,o59,o60,o61,o62,o63
);

wire [23:0] w0,w1,w2,w3,w4,w5,w6,w7;
wire [23:0] w8,w9,w10,w11,w12,w13,w14,w15;
wire [23:0] w16,w17,w18,w19,w20,w21,w22,w23;
wire [23:0] w24,w25,w26,w27,w28,w29,w30,w31;
wire [23:0] w32,w33,w34,w35,w36,w37,w38,w39;
wire [23:0] w40,w41,w42,w43,w44,w45,w46,w47;
wire [23:0] w48,w49,w50,w51,w52,w53,w54,w55;
wire [23:0] w56,w57,w58,w59,w60,w61,w62,w63;
```

```
DCT1D8P #(.CNT_Clk(CNT_CLK)) D1
(.clk(clk),.clk_cnt(clk_cnt),.a0(a0),.a1(a1),.a2(a2),.a3(a3),.a4(a4),.a5(
a5),.a6(a6),.a7(a7),
.o0(w0),.o1(w1),.o2(w2),.o3(w3),.o4(w4),.o5(w5),.o6(w6),.o7(w7
);

DCT1D8P #(.CNT_Clk(CNT_CLK)) D2
(.clk(clk),.clk_cnt(clk_cnt),.a0(a8),.a1(a9),.a2(a10),.a3(a11),.a4(a12),
.a5(a13),.a6(a14),.a7(a15),
.o0(w8),.o1(w9),.o2(w10),.o3(w11),.o4(w12),.o5(w13),.o6(w14),.o7(
w15));

DCT1D8P #(.CNT_Clk(CNT_CLK)) D3
(.clk(clk),.clk_cnt(clk_cnt),.a0(a16),.a1(a17),.a2(a18),.a3(a19),.a4(a2
0),.a5(a21),.a6(a22),.a7(a23),
.o0(w16),.o1(w17),.o2(w18),.o3(w19),.o4(w20),.o5(w21),.o6(w22),.
o7(w23));

DCT1D8P #(.CNT_Clk(CNT_CLK)) D4
(.clk(clk),.clk_cnt(clk_cnt),.a0(a24),.a1(a25),.a2(a26),.a3(a27),.a4(a2
8),.a5(a29),.a6(a30),.a7(a31),
.o0(w24),.o1(w25),.o2(w26),.o3(w27),.o4(w28),.o5(w29),.o6(w30),.
o7(w31));

DCT1D8P #(.CNT_Clk(CNT_CLK)) D5
(.clk(clk),.clk_cnt(clk_cnt),.a0(a32),.a1(a33),.a2(a34),.a3(a35),.a4(a3
6),.a5(a37),.a6(a38),.a7(a39),
.o0(w32),.o1(w33),.o2(w34),.o3(w35),.o4(w36),.o5(w37),.o6(w38),.
o7(w39));
```

```
DCT1D8P #(.CNT_Clk(CNT_CLK)) D6
(.clk(clk),.clk_cnt(clk_cnt),.a0(a40),.a1(a41),.a2(a42),.a3(a43),.a4(a4
4),.a5(a45),.a6(a46),.a7(a47),
.o0(w40),.o1(w41),.o2(w42),.o3(w43),.o4(w44),.o5(w45),.o6(w46),.
o7(w47));

DCT1D8P #(.CNT_Clk(CNT_CLK)) D7
(.clk(clk),.clk_cnt(clk_cnt),.a0(a48),.a1(a49),.a2(a50),.a3(a51),.a4(a5
2),.a5(a53),.a6(a54),.a7(a55),
.o0(w48),.o1(w49),.o2(w50),.o3(w51),.o4(w52),.o5(w53),.o6(w54),.
o7(w55));

DCT1D8P #(.CNT_Clk(CNT_CLK)) D8
(.clk(clk),.clk_cnt(clk_cnt),.a0(a56),.a1(a57),.a2(a58),.a3(a59),.a4(a6
0),.a5(a61),.a6(a62),.a7(a63),
.o0(w56),.o1(w57),.o2(w58),.o3(w59),.o4(w60),.o5(w61),.o6(w62),.
o7(w63));

Buffer #(.CNT_CLK(CNT_CLK+4)) B1
(.clk(clk),.clk_cnt(clk_cnt),.a0(w0),.a1(w1),.a2(w2),.a3(w3),.a4(w4),.
a5(w5),.a6(w6),.a7(w7),
.o0(o0),.o1(o8),.o2(o16),.o3(o24),.o4(o32),.o5(o40),.o6(o48),.o7(o56
));

Buffer #(.CNT_CLK(CNT_CLK+4)) B2
(.clk(clk),.clk_cnt(clk_cnt),.a0(w8),.a1(w9),.a2(w10),.a3(w11),.a4(w
12),.a5(w13),.a6(w14),.a7(w15),
.o0(o1),.o1(o9),.o2(o17),.o3(o25),.o4(o33),.o5(o41),.o6(o49),.o7(o57
));
```

```
Buffer #(.CNT_CLK(CNT_CLK+4)) B3
(.clk(clk),.clk_cnt(clk_cnt),.a0(w16),.a1(w17),.a2(w18),.a3(w19),.a4(
w20),.a5(w21),.a6(w22),.a7(w23),
.o0(o2),.o1(o10),.o2(o18),.o3(o26),.o4(o34),.o5(o42),.o6(o50),.o7(o5
8));

Buffer #(.CNT_CLK(CNT_CLK+4)) B4
(.clk(clk),.clk_cnt(clk_cnt),.a0(w24),.a1(w25),.a2(w26),.a3(w27),.a4(
w28),.a5(w29),.a6(w30),.a7(w31),
.o0(o3),.o1(o11),.o2(o19),.o3(o27),.o4(o35),.o5(o43),.o6(o51),.o7(o5
9));

Buffer #(.CNT_CLK(CNT_CLK+4)) B5
(.clk(clk),.clk_cnt(clk_cnt),.a0(w32),.a1(w33),.a2(w34),.a3(w35),.a4(
w36),.a5(w37),.a6(w38),.a7(w39),
.o0(o4),.o1(o12),.o2(o20),.o3(o28),.o4(o36),.o5(o44),.o6(o52),.o7(o6
0));

Buffer #(.CNT_CLK(CNT_CLK+4)) B6
(.clk(clk),.clk_cnt(clk_cnt),.a0(w40),.a1(w41),.a2(w42),.a3(w43),.a4(
w44),.a5(w45),.a6(w46),.a7(w47),
.o0(o5),.o1(o13),.o2(o21),.o3(o29),.o4(o37),.o5(o45),.o6(o53),.o7(o6
1));

Buffer #(.CNT_CLK(CNT_CLK+4))
B7(.clk(clk),.clk_cnt(clk_cnt),.a0(w48),.a1(w49),.a2(w50),.a3(w51),.
a4(w52),.a5(w53),.a6(w54),.a7(w55),
.o0(o6),.o1(o14),.o2(o22),.o3(o30),.o4(o38),.o5(o46),.o6(o54),.o7(o6
2));
```

```
Buffer #(.CNT_CLK(CNT_CLK+4)) B8
(.clk(clk),.clk_cnt(clk_cnt),.a0(w56),.a1(w57),.a2(w58),.a3(w59),.a4(
w60),.a5(w61),.a6(w62),.a7(w63),
.o0(o7),.o1(o15),.o2(o23),.o3(o31),.o4(o39),.o5(o47),.o6(o55),.o7(o6
3));


endmodule
```

7. Code khối DCT 2D

```
module DCT2D
(
  input clk, reset,
  input [3:0] clk_cnt,
  input [23:0] a0,a1,a2,a3,a4,a5,a6,a7,
  input [23:0] a8,a9,a10,a11,a12,a13,a14,a15,
  input [23:0] a16,a17,a18,a19,a20,a21,a22,a23,
  input [23:0] a24,a25,a26,a27,a28,a29,a30,a31,
  input [23:0] a32,a33,a34,a35,a36,a37,a38,a39,
  input [23:0] a40,a41,a42,a43,a44,a45,a46,a47,
  input [23:0] a48,a49,a50,a51,a52,a53,a54,a55,
  input [23:0] a56,a57,a58,a59,a60,a61,a62,a63,
  output wire [23:0] o0,o1,o2,o3,o4,o5,o6,o7,
  output wire [23:0] o8,o9,o10,o11,o12,o13,o14,o15,
       output wire [23:0] o16,o17,o18,o19,o20,o21,o22,o23,
     output wire [23:0] o24,o25,o26,o27,o28,o29,o30,o31,
  output wire [23:0] o32,o33,o34,o35,o36,o37,o38,o39,
  output wire [23:0] o40,o41,o42,o43,o44,o45,o46,o47,
  output wire [23:0] o48,o49,o50,o51,o52,o53,o54,o55,
  output wire [23:0] o56,o57,o58,o59,o60,o61,o62,o63
```

```
);
localparam size_cnt = 3;
localparam clk_DD1 = 1, clk_DD2 = 5;
wire [23:0] w0,w1,w2,w3,w4,w5,w6,w7;
wire [23:0] w8,w9,w10,w11,w12,w13,w14,w15;
wire [23:0] w16,w17,w18,w19,w20,w21,w22,w23;
wire [23:0] w24,w25,w26,w27,w28,w29,w30,w31;
wire [23:0] w32,w33,w34,w35,w36,w37,w38,w39;
wire [23:0] w40,w41,w42,w43,w44,w45,w46,w47;
wire [23:0] w48,w49,w50,w51,w52,w53,w54,w55;
wire [23:0] w56,w57,w58,w59,w60,w61,w62,w63;

Cnt_clk   #(.SIZE_CNT(size_cnt))                    cnt_clk1
  (.clk(clk),.reset(reset),.Cnt(cnt_clk));

DCT1D #(.CNT_CLK(clk_DD1)) DD1 (.clk(clk),.clk_cnt(clk_cnt),
.a0(a0),.a1(a1),.a2(a2),.a3(a3),.a4(a4),.a5(a5),.a6(a6),.a7(a7),
.a8(a8),.a9(a9),.a10(a10),.a11(a11),.a12(a12),.a13(a13),.a14(a14),.a1
5(a15),
.a16(a16),.a17(a17),.a18(a18),.a19(a19),.a20(a20),.a21(a21),.a22(a22
),.a23(a23),
.a24(a24),.a25(a25),.a26(a26),.a27(a27),.a28(a28),.a29(a29),.a30(a30
),.a31(a31),
.a32(a32),.a33(a33),.a34(a34),.a35(a35),.a36(a36),.a37(a37),.a38(a38
),.a39(a39),
.a40(a40),.a41(a41),.a42(a42),.a43(a43),.a44(a44),.a45(a45),.a46(a46
),.a47(a47),
.a48(a48),.a49(a49),.a50(a50),.a51(a51),.a52(a52),.a53(a53),.a54(a54
),.a55(a55),
.a56(a56),.a57(a57),.a58(a58),.a59(a59),.a60(a60),.a61(a61),.a62(a62
```

```
),.a63(a63),
.o0(w0),.o1(w1),.o2(w2),.o3(w3),.o4(w4),.o5(w5),.o6(w6),.o7(w7),
.o8(w8),.o9(w9),.o10(w10),.o11(w11),.o12(w12),.o13(w13),.o14(w1
4),.o15(w15),
.o16(w16),.o17(w17),.o18(w18),.o19(w19),.o20(w20),.o21(w21),.o2
2(w22),.o23(w23),
.o24(w24),.o25(w25),.o26(w26),.o27(w27),.o28(w28),.o29(w29),.o3
0(w30),.o31(w31),
.o32(w32),.o33(w33),.o34(w34),.o35(w35),.o36(w36),.o37(w37),.o3
8(w38),.o39(w39),
.o40(w40),.o41(w41),.o42(w42),.o43(w43),.o44(w44),.o45(w45),.o4
6(w46),.o47(w47),
.o48(w48),.o49(w49),.o50(w50),.o51(w51),.o52(w52),.o53(w53),.o5
4(w54),.o55(w55),
.o56(w56),.o57(w57),.o58(w58),.o59(w59),.o60(w60),.o61(w61),.o6
2(w62),.o63(w63));

DCT1D #(.CNT_CLK(clk_DD2)) DD2 (.clk(clk),.clk_cnt(clk_cnt),
.a0(w0),.a1(w1),.a2(w2),.a3(w3),.a4(w4),.a5(w5),.a6(w6),.a7(w7),
.a8(w8),.a9(w9),.a10(w10),.a11(w11),.a12(w12),.a13(w13),.a14(w14
),.a15(w15),
.a16(w16),.a17(w17),.a18(w18),.a19(w19),.a20(w20),.a21(w21),.a22
(w22),.a23(w23),
.a24(w24),.a25(w25),.a26(w26),.a27(w27),.a28(w28),.a29(w29),.a30
(w30),.a31(w31),
.a32(w32),.a33(w33),.a34(w34),.a35(w35),.a36(w36),.a37(w37),.a38
(w38),.a39(w39),
.a40(w40),.a41(w41),.a42(w42),.a43(w43),.a44(w44),.a45(w45),.a46
(w46),.a47(w47),
.a48(w48),.a49(w49),.a50(w50),.a51(w51),.a52(w52),.a53(w53),.a54
```

```
(w54),.a55(w55),
.a56(w56),.a57(w57),.a58(w58),.a59(w59),.a60(w60),.a61(w61),.a62
(w62),.a63(w63),
.o0(o0),.o1(o1),.o2(o2),.o3(o3),.o4(o4),.o5(o5),.o6(o6),.o7(o7),
.o8(o8),.o9(o9),.o10(o10),.o11(o11),.o12(o12),.o13(o13),.o14(o14),.o
15(o15),
.o16(o16),.o17(o17),.o18(o18),.o19(o19),.o20(o20),.o21(o21),.o22(o
22),.o23(o23),
.o24(o24),.o25(o25),.o26(o26),.o27(o27),.o28(o28),.o29(o29),.o30(o
30),.o31(o31),
.o32(o32),.o33(o33),.o34(o34),.o35(o35),.o36(o36),.o37(o37),.o38(o
38),.o39(o39),
.o40(o40),.o41(o41),.o42(o42),.o43(o43),.o44(o44),.o45(o45),.o46(o
46),.o47(o47),
.o48(o48),.o49(o49),.o50(o50),.o51(o51),.o52(o52),.o53(o53),.o54(o
54),.o55(o55),
.o56(o56),.o57(o57),.o58(o58),.o59(o59),.o60(o60),.o61(o61),.o62(o
62),.o63(o63)
);

endmodule
```

8. Code testbench DTC_ODD.

```
module Test_DCT_ODD();


reg clk;
reg [3:0] cnt_clk;
reg [23:0] a0, a1, a2, a3, a4, a5, a6, a7;
```

```
wire [23:0] c;

localparam c1 = 20'b00000000000000000001, c2 = 20'b00000000000000000001, c3 = 20'b00000000000000000001, c4 = 20'b00000000000000000001;

localparam CNT_Clk = 1;

localparam cu0 = 2.82, cu = 512;


DCT_ODD #(.c1(c1),.c2(c2),.c3(c3),.c4(c4),.CNT_CLK(CNT_Clk),.cu(cu0)) tb_DCT_ODD_EVEN(.clk(clk), .cnt_clk(cnt_clk),

.a0(a0), .a1(a1), .a2(a2), .a3(a3), .a4(a4), .a5(a5), .a6(a6), .a7(a7),

.c(c));


initial

begin

 clk = 1;

 forever  #5 clk = ~clk;

end


initial

begin

    a0 =  24'b000101111011100101101;//24'b000010000001000000000;

    a1 =  24'b000101111110110001111;//24'b000010000010000000000;

    a2 =  24'b000110000001010000110;//24'b000010000010000000000;

    a3 =  24'b000110000000111010001;//24'b000010000000100000000;

    a4 =  24'b000110000011100100010;//24'b000010000010100000000;

    a5 =  24'b000110000000011000000;//24'b000010000010100000000;

    a6 =  24'b000110000011000010010;//24'b000010000011000000000;

    a7 =  24'b000110000011111011000;//24'b000010000011100000000;

    #10;
```

```
    cnt_clk = 1;
    #10;
    cnt_clk = 2;
    #10;
    cnt_clk = 3;
end
endmodule
```

9. Code testbench DCT EVEN.

```
module Test_DCT_EVEN();


reg clk;
reg [3:0] cnt_clk;
reg [24:0] a0, a1, a2, a3, a4, a5, a6, a7;
wire [24:0] c;
localparam c1 = 20'b00000000000011111011, c2 = 20'b00000000000011010100, c3 = 20'b00000000000010001110, c4 = 20'b00000000000000110001;
localparam CNT_Clk = 1;
localparam cu0 = 2.82, cu = 512;


DCT_EVEN #(.c1(c1),.c2(c2),.c3(c3),.c4(c4),.CNT_CLK(CNT_Clk),.cu(cu)) tb_DCT_ODD_EVEN(.clk(clk), .cnt_clk(cnt_clk),
.a0(a0), .a1(a1), .a2(a2), .a3(a3), .a4(a4), .a5(a5), .a6(a6), .a7(a7),
.c(c)
);


initial
```

```
begin
 clk = 1;
 forever  #5 clk = ~clk;
end


initial
begin
    a0 = 24'b000010000001000000000;//130
    a1 = 24'b000010000010000000000;
    a2 = 24'b000010000010000000000;
    a3 =  24'b000010000000100000000;
    a4 =  24'b000010000010100000000;
    a5 =  24'b000010000010100000000;
    a6 =  24'b000010000011000000000;
    a7 =  24'b000010000011100000000;
    #10;
    cnt_clk = 1;
    #10;
    cnt_clk = 2;
    #10;
    cnt_clk = 3;
end
endmodule
```

10. Code testbench DCT1D8P.

```
module test_DCT1D8P();


reg clk;
```

```
reg [3:0] cnt_clk;
reg [23:0] a0, a1, a2, a3, a4, a5, a6, a7;
wire [23:0] w0, w1, w2, w3, w4, w5, w6, w7;
localparam CNT_Clk = 1;

DCT1D8P #(.CNT_Clk(CNT_Clk)) P (.clk(clk),.clk_cnt(cnt_clk),.a0(a0),.a1(a1),.a2(a2),.a3(a3),.a4(a4),.a5(a5),.a6(a6),.a7(a7),
.o0(w0),.o1(w1),.o2(w2),.o3(w3),.o4(w4),.o5(w5),.o6(w6),.o7(w7));

initial
begin
 clk = 1;
 forever  #5 clk = ~clk;
end

initial
begin
    a0 =  24'b000101111011100101101;//24'b000010000001000000000;
    a1 =  24'b000101111110110001111;//24'b000010000010000000000;
    a2 =  24'b000110000001010000110;//24'b000010000010000000000;
    a3 =  24'b000110000000111010001;//24'b000010000000100000000;
    a4 =  24'b000110000011100100010;//24'b000010000010100000000;
    a5 =  24'b000110000000011000000;//24'b000010000010100000000;
    a6 =  24'b000110000011000010010;//24'b000010000011000000000;
    a7 =  24'b000110000011111011000;//24'b000010000011100000000;
    #10;
    cnt_clk = 1;
```

```
    #10;
    cnt_clk = 2;
    #10;
    cnt_clk = 3;
end
endmodule
```

11. Code testbench DCT1D.

```
module testB_DCT1D();


    reg     clk;
    reg [3:0] cnt_clk;
    reg [23:0] a0,a1,a2,a3,a4,a5,a6,a7;
    reg [23:0] a8,a9,a10,a11,a12,a13,a14,a15;
    reg [23:0] a16,a17,a18,a19,a20,a21,a22,a23;
    reg [23:0] a24,a25,a26,a27,a28,a29,a30,a31;
    reg [23:0] a32,a33,a34,a35,a36,a37,a38,a39;
    reg [23:0] a40,a41,a42,a43,a44,a45,a46,a47;
    reg [23:0] a48,a49,a50,a51,a52,a53,a54,a55;
    reg [23:0] a56,a57,a58,a59,a60,a61,a62,a63;
    wire [23:0] w0,w1,w2,w3,w4,w5,w6,w7;
    wire [23:0] w8,w9,w10,w11,w12,w13,w14,w15;
    wire [23:0] w16,w17,w18,w19,w20,w21,w22,w23;
    wire [23:0] w24,w25,w26,w27,w28,w29,w30,w31;
    wire [23:0] w32,w33,w34,w35,w36,w37,w38,w39;
    wire [23:0] w40,w41,w42,w43,w44,w45,w46,w47;
    wire [23:0] w48,w49,w50,w51,w52,w53,w54,w55;
    wire [23:0] w56,w57,w58,w59,w60,w61,w62,w63;
```

```
    localparam CNT_Clk= 1;


DCT1D #(.CNT_CLK(CNT_Clk)) testB_DCT2D (.clk(clk),.clk_cnt(cnt_clk),
.a0(a0),.a1(a1),.a2(a2),.a3(a3),.a4(a4),.a5(a5),.a6(a6),.a7(a7),
.a8(a8),.a9(a9),.a10(a10),.a11(a11),.a12(a12),.a13(a13),.a14(a14),.a15(a15),
.a16(a16),.a17(a17),.a18(a18),.a19(a19),.a20(a20),.a21(a21),.a22(a22),.a23(a23),
.a24(a24),.a25(a25),.a26(a26),.a27(a27),.a28(a28),.a29(a29),.a30(a30),.a31(a31),
.a32(a32),.a33(a33),.a34(a34),.a35(a35),.a36(a36),.a37(a37),.a38(a38),.a39(a39),
.a40(a40),.a41(a41),.a42(a42),.a43(a43),.a44(a44),.a45(a45),.a46(a46),.a47(a47),
.a48(a48),.a49(a49),.a50(a50),.a51(a51),.a52(a52),.a53(a53),.a54(a54),.a55(a55),
.a56(a56),.a57(a57),.a58(a58),.a59(a59),.a60(a60),.a61(a61),.a62(a62),.a63(a63),
.o0(w0),.o1(w1),.o2(w2),.o3(w3),.o4(w4),.o5(w5),.o6(w6),.o7(w7),
.o8(w8),.o9(w9),.o10(w10),.o11(w11),.o12(w12),.o13(w13),.o14(w14),.o15(w15),
.o16(w16),.o17(w17),.o18(w18),.o19(w19),.o20(w20),.o21(w21),.o22(w22),.o23(w
23),
.o24(w24),.o25(w25),.o26(w26),.o27(w27),.o28(w28),.o29(w29),.o30(w30),.o31(w
31),
.o32(w32),.o33(w33),.o34(w34),.o35(w35),.o36(w36),.o37(w37),.o38(w38),.o39(w
39),
.o40(w40),.o41(w41),.o42(w42),.o43(w43),.o44(w44),.o45(w45),.o46(w46),.o47(w
47),
.o48(w48),.o49(w49),.o50(w50),.o51(w51),.o52(w52),.o53(w53),.o54(w54),.o55(w
55),
.o56(w56),.o57(w57),.o58(w58),.o59(w59),.o60(w60),.o61(w61),.o62(w62),.o63(w
63));


initial
begin
    clk =0;
    forever #5 clk = ~clk;
```

```
end

initial
begin
    a0 = 24'b1000001000000000;//130
    a1 = 24'b1000010000000000;
    a2 = 24'b1000010000000000;
    a3 =  24'b1000000100000000;
    a4 =  24'b1000010100000000;
    a5 =  24'b1000010100000000;
    a6 =  24'b1000011000000000;
    a7 =  24'b1000011100000000;

    a8 =  24'b1000011100000000;
    a9 =  24'b1000010100000000;
    a10  =  24'b1000010100000000;
    a11  =  24'b1000001100000000;
    a12 =  24'b1000010100000000;
    a13 =  24'b1000100100000000;
    a14 =  24'b1000101000000000;
    a15 =  24'b1000100000000000;

    a16 =  24'b1000010000000000;
    a17 =  24'b1000011000000000;
    a18 =  24'b1000010100000000;
    a19 =  24'b1000100000000000;
    a20 =  24'b1000101100000000;
    a21 =  24'b1000111000000000;
```

```
a22 =  24'b1000100100000000;
a23 =  24'b1000100100000000;

a24 =  24'b1000100100000000;
a25 =  24'b1000100000000000;
a26 =  24'b1000100000000000;
a27 =  24'b1000011100000000;
a28 =  24'b1000011100000000;
a29 =  24'b1000100000000000;
a30 =  24'b1000011100000000;
a31 =  24'b1000101000000000;

a32 =  24'b1000101100000000;
a33 =  24'b1000101000000000;
a34 =  24'b1000101100000000;
a35 =  24'b1000011100000000;
a36 =  24'b1000100000000000;
a37 =  24'b1000101100000000;
a38 =  24'b1000100100000000;
a39 =  24'b1000110000000000;

a40 =  24'b1000011000000000;
a41 =  24'b1000100000000000;
a42 =  24'b1000100100000000;
a43 =  24'b1000100000000000;
a44 =  24'b1000011000000000;
a45 =  24'b1000100000000000;
a46 =  24'b1000100000000000;
```

```
a47 =  24'b1000100000000000;

a48 =  24'b1000100000000000;
a49 =  24'b1000010100000000;
a50 =  24'b1000101000000000;
a51 =  24'b1000100000000000;
a52 =  24'b1000100000000000;
a53 =  24'b1000100000000000;
a54 =  24'b1000101100000000;
a55 =  24'b1001001000000000;



a56 =  24'b1000101100000000;
a57 =  24'b1000100100000000;
a58 =  24'b1000101000000000;
a59 =  24'b1000011000000000;
a60 =  24'b1000010100000000;
a61 =  24'b1000101000000000;
a62 =  24'b1000110000000000;
a63 =  24'b1001001000000000;

#10;
cnt_clk = 1;
#10;
cnt_clk = 2;
#10;
cnt_clk = 3;
#10;
```

```
    cnt_clk = 4;
    #10;
    cnt_clk = 5;
end
endmodule
```

12. Code testbench DCT2D.

```
module testB_DCT2D();


    reg    clk, reset;
    reg [23:0] a0,a1,a2,a3,a4,a5,a6,a7;
    reg [23:0] a8,a9,a10,a11,a12,a13,a14,a15;
    reg [23:0] a16,a17,a18,a19,a20,a21,a22,a23;
    reg [23:0] a24,a25,a26,a27,a28,a29,a30,a31;
    reg [23:0] a32,a33,a34,a35,a36,a37,a38,a39;
    reg [23:0] a40,a41,a42,a43,a44,a45,a46,a47;
    reg [23:0] a48,a49,a50,a51,a52,a53,a54,a55;
    reg [23:0] a56,a57,a58,a59,a60,a61,a62,a63;
    wire [23:0] w0,w1,w2,w3,w4,w5,w6,w7;
    wire [23:0] w8,w9,w10,w11,w12,w13,w14,w15;
    wire [23:0] w16,w17,w18,w19,w20,w21,w22,w23;
    wire [23:0] w24,w25,w26,w27,w28,w29,w30,w31;
    wire [23:0] w32,w33,w34,w35,w36,w37,w38,w39;
    wire [23:0] w40,w41,w42,w43,w44,w45,w46,w47;
    wire [23:0] w48,w49,w50,w51,w52,w53,w54,w55;
    wire [23:0] w56,w57,w58,w59,w60,w61,w62,w63;


DCT2D testB_DCT2D (.clk(clk),.reset(reset),
```

```
.a0(a0),.a1(a1),.a2(a2),.a3(a3),.a4(a4),.a5(a5),.a6(a6),.a7(a7),
.a8(a8),.a9(a9),.a10(a10),.a11(a11),.a12(a12),.a13(a13),.a14(a14),.a15(a15),
.a16(a16),.a17(a17),.a18(a18),.a19(a19),.a20(a20),.a21(a21),.a22(a22),.a23(a23),
.a24(a24),.a25(a25),.a26(a26),.a27(a27),.a28(a28),.a29(a29),.a30(a30),.a31(a31),
.a32(a32),.a33(a33),.a34(a34),.a35(a35),.a36(a36),.a37(a37),.a38(a38),.a39(a39),
.a40(a40),.a41(a41),.a42(a42),.a43(a43),.a44(a44),.a45(a45),.a46(a46),.a47(a47),
.a48(a48),.a49(a49),.a50(a50),.a51(a51),.a52(a52),.a53(a53),.a54(a54),.a55(a55),
.a56(a56),.a57(a57),.a58(a58),.a59(a59),.a60(a60),.a61(a61),.a62(a62),.a63(a63),
.o0(w0),.o1(w1),.o2(w2),.o3(w3),.o4(w4),.o5(w5),.o6(w6),.o7(w7),
.o8(w8),.o9(w9),.o10(w10),.o11(w11),.o12(w12),.o13(w13),.o14(w14),.o15(w15),
.o16(w16),.o17(w17),.o18(w18),.o19(w19),.o20(w20),.o21(w21),.o22(w22),.o23(w
23),
.o24(w24),.o25(w25),.o26(w26),.o27(w27),.o28(w28),.o29(w29),.o30(w30),.o31(w
31),
.o32(w32),.o33(w33),.o34(w34),.o35(w35),.o36(w36),.o37(w37),.o38(w38),.o39(w
39),
.o40(w40),.o41(w41),.o42(w42),.o43(w43),.o44(w44),.o45(w45),.o46(w46),.o47(w
47),
.o48(w48),.o49(w49),.o50(w50),.o51(w51),.o52(w52),.o53(w53),.o54(w54),.o55(w
55),
.o56(w56),.o57(w57),.o58(w58),.o59(w59),.o60(w60),.o61(w61),.o62(w62),.o63(w
63));


initial
begin
    clk =0;
    forever #5 clk = ~clk;
end


initial
```

```
begin
    a0 = 24'b1000001000000000;//130
    a1 = 24'b1000010000000000;
    a2 = 24'b1000010000000000;
    a3 =  24'b1000000100000000;
    a4 =  24'b1000010100000000;
    a5 =  24'b1000010100000000;
    a6 =  24'b1000011000000000;
    a7 =  24'b1000011100000000;

    a8 =  24'b1000011100000000;
    a9 =  24'b1000010100000000;
    a10  =  24'b1000010100000000;
    a11  =  24'b1000001100000000;
    a12 =  24'b1000010100000000;
    a13 =  24'b1000100100000000;
    a14 =  24'b1000101000000000;
    a15 =  24'b1000100000000000;

    a16 =  24'b1000010000000000;
    a17 =  24'b1000011000000000;
    a18 =  24'b1000010100000000;
    a19 =  24'b1000100000000000;
    a20 =  24'b1000101100000000;
    a21 =  24'b1000111000000000;
    a22 =  24'b1000100100000000;
    a23 =  24'b1000100100000000;
```

```
a24 =  24'b1000100100000000;
a25 =  24'b1000100000000000;
a26 =  24'b1000100000000000;
a27 =  24'b1000011100000000;
a28 =  24'b1000011100000000;
a29 =  24'b1000100000000000;
a30 =  24'b1000011100000000;
a31 =  24'b1000101000000000;

a32 =  24'b1000101100000000;
a33 =  24'b1000101000000000;
a34 =  24'b1000101100000000;
a35 =  24'b1000011100000000;
a36 =  24'b1000100000000000;
a37 =  24'b1000101100000000;
a38 =  24'b1000100100000000;
a39 =  24'b1000110000000000;

a40 =  24'b1000011000000000;
a41 =  24'b1000100000000000;
a42 =  24'b1000100100000000;
a43 =  24'b1000100000000000;
a44 =  24'b1000011000000000;
a45 =  24'b1000100000000000;
a46 =  24'b1000100000000000;
a47 =  24'b1000100000000000;

a48 =  24'b1000100000000000;
```

```
    a49 =  24'b1000010100000000;
    a50 =  24'b1000101000000000;
    a51 =  24'b1000100000000000;
    a52 =  24'b1000100000000000;
    a53 =  24'b1000100000000000;
    a54 =  24'b1000101100000000;
    a55 =  24'b1001001000000000;



    a56 =  24'b1000101100000000;
    a57 =  24'b1000100100000000;
    a58 =  24'b1000101000000000;
    a59 =  24'b1000011000000000;
    a60 =  24'b1000010100000000;
    a61 =  24'b1000101000000000;
    a62 =  24'b1000110000000000;
    a63 =  24'b1001001000000000;

end
endmodule
```