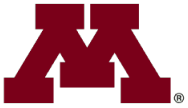


UNIVERSITY OF MINNESOTA

5327 VLSI DESIGN LAB

Project Report 1

Submitted by : *Samson Hruday Chinta*

Email : *chint078@umn.edu*

Submitted to :

Prof. Yu Cao ,

Balagopal Anoop

Department of Electrical and Computer Engineering

UMN ,Twincities

Contents

VLSI DESIGN LABORATORY REPORT 1

1.1 Objectives

This report documents the design and synthesis of a 2D Discrete Cosine Transform (DCT) module implemented in Verilog. The DCT is a mathematical transformation commonly used in digital signal processing (DSP), especially in image and video compression applications. Its primary function is to convert spatial data (such as an image) into frequency data, which allows efficient data representation by focusing on the most significant frequency components.

The objective of this project is to develop a high-performance 2D DCT module, with an operating frequency target around 800 MHz, that maintains precision and operates with minimized area(yet to be optimized) and gate count, making it suitable for use in real-time applications.

1.2 Theory

The DCT is a popular transform in DSP, notably in the field of data compression. It divides a signal into frequency components, allowing for effective data encoding by emphasizing high-energy, low-frequency components while rejecting less relevant, high-frequency ones. This feature is particularly useful in applications requiring high storage and transmission efficiency, such as image, video, and audio compression.

Comparison with Other Transforms:

Discrete Fourier Transform (DFT): The DFT provides a more general frequency analysis, including both real and imaginary (complex) components. However, the DCT, focusing only on real values, is often computationally simpler and more efficient in applications like compression, where complex components are unnecessary.

Discrete Wavelet Transform (DWT): The DWT is another common transform in signal and image processing, especially in recent compression standards like JPEG 2000. Unlike the DCT, which captures global frequency information, the DWT provides a localized frequency representation. DWTs can be better for images with sharp changes, while DCTs are often more efficient for smooth regions.

Karhunen-Loève Transform (KLT): The KLT is theoretically optimal in data decorrelation, but its adaptive nature and computational intensity limit its practical use in real-time applications. The DCT, on the other hand, offers a good balance of efficiency and decorrelation ability, making it highly suitable for standardized compression algorithms.

Applications of DCT 1. Image Compression: The 2D DCT is a core operation in standards like JPEG, where it converts image blocks into frequency components, enabling reduction of data without significant loss in quality.
2. Video Compression: Used in formats like MPEG and H.264, DCT enables efficient encoding by reducing redundancy across frames.
3.Signal Processing: The DCT's ability to represent signals in the frequency domain makes it suitable for noise reduction, filtering, and other DSP operations.
4.Pattern Recognition: DCT is used in feature extraction, where frequency characteristics play a role in identifying patterns in signals.

1.3 Computation Of DCT Using Various Approaches

There are numerous architectures for computing DCT and each of them variers with the application and the each of them have different algorithms.

Approach 1: Matrix Multiplication

The 2D DCT is computed by applying a 1D DCT transformation first on the rows and then on the columns of the input matrix. For an $N \times N$ input matrix $\mathbf{X}$, the 2D DCT is defined as:

$$\mathbf{Y} = \mathbf{C} \cdot \mathbf{X} \cdot \mathbf{C}^T$$

where:

- $\mathbf{Y}$ is the resulting $N \times N$ DCT matrix,
- $\mathbf{C}$ is the DCT transform matrix of size $N \times N$, and
- $\mathbf{C}^T$ denotes the transpose of $\mathbf{C}$.

Step-by-Step Computation
1. Define the DCT Matrix $\mathbf{C}$: The elements of $\mathbf{C}$ are defined as:

$$C_{i,j} = \sqrt{\frac{2}{N}} \cos\left(\frac{\pi(2j\ 1)i}{2N}\right), \quad i, j = 0, 1, \ldots, N-1$$

with $C_{0,j} = \sqrt{\frac{1}{N}}$ to account for the scaling factor in the first row.

2.Apply Row-wise Transformation: Multiply the input matrix $\mathbf{X}$ by $\mathbf{C}$ to obtain an intermediate matrix:

$$\mathbf{Z} = \mathbf{C} \cdot \mathbf{X}$$

3. Apply Column-wise Transformation: Multiply the result by the transpose of $\mathbf{C}$ to complete the transformation:

$$\mathbf{Y} = \mathbf{Z} \cdot \mathbf{C}^T$$

4. The resulting matrix $\mathbf{Y}$ contains the 2D DCT coefficients, which represent the input data in the frequency domain. Low-frequency components are concentrated in the top-left corner of $\mathbf{Y}$.This can be extremely at a disadvantage as it is very inefficient for hardware implementations and involves a lot of adds and multiplications.

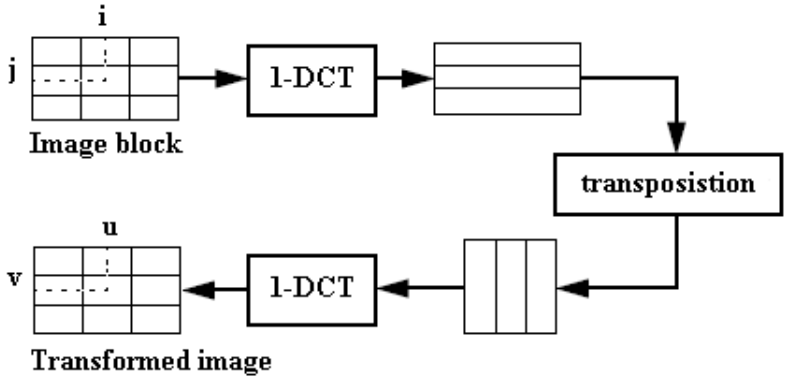


Figure 1: Generic DCT Transformation

Approach 2: Butterfly Approach

The Butterfly structure is a popular method for efficiently computing Discrete Cosine Transform (DCT) coefficients, especially in hardware implementations. Butterfly-based DCT structures divide the computation into stages with simple operations (like additions, subtractions, and shifts), reducing the complexity of multipliers. This approach enables high-speed DCT computations with minimized area and power consumption, making it highly suitable for hardware implementations, such as in VLSI circuits for image and video compression.

The butterfly approach for 1D DCT is efficient for hardware implementations, reducing the need for multipliers by using simpler additions and subtractions, making it suitable for VLSI applications.

Step-by-Step Butterfly Structure for 1D DCT

For an 8-point input vector $x = [x_0, x_1, \ldots, x_7]$, the butterfly approach applies stages of additions and subtractions to simplify the computation. The DCT output coefficients X_k are calculated as follows:

1. Initial Pairwise Addition and Subtraction (Stage 1):

$$\begin{aligned} s_0 &= x_0\ x_7, & s_4 &= x_0 - x_7, \\ s_1 &= x_1\ x_6, & s_5 &= x_1 - x_6, \\ s_2 &= x_2\ x_5, & s_6 &= x_2 - x_5, \\ s_3 &= x_3\ x_4, & s_7 &= x_3 - x_4. \end{aligned}$$

2.Secondary Butterfly Computations (Stage 2):

$$\begin{aligned} t_0 &= s_0\ s_3, & t_2 &= s_0 - s_3, \\ t_1 &= s_1\ s_2, & t_3 &= s_1 - s_2. \end{aligned}$$

3. Final DCT Output Calculation:

$$\begin{aligned} X_0 &= \alpha_0 \cdot t_0, \\ X_1 &= \alpha_1 \cdot (t_3\ s_7), \\ X_2 &= \alpha_2 \cdot t_1, \\ X_3 &= \alpha_3 \cdot (t_2\ s_6), \\ X_4 &= \alpha_4 \cdot t_2, \\ X_5 &= \alpha_5 \cdot (t_3 - s_7), \\ X_6 &= \alpha_6 \cdot t_3, \\ X_7 &= \alpha_7 \cdot (t_1 - s_5). \end{aligned}$$

Example Visualization of Butterfly Structure

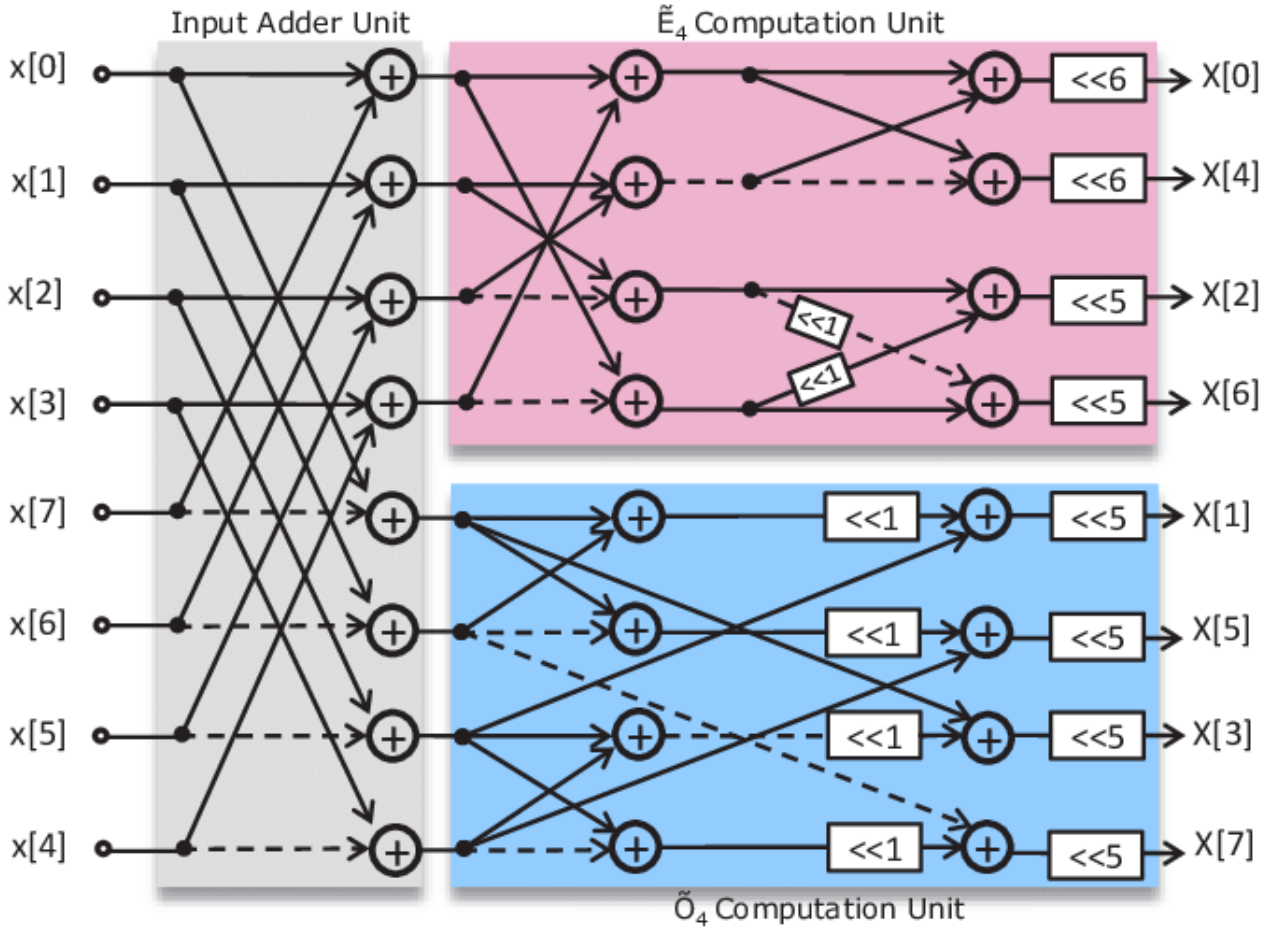


Figure 2: 8 point 1D dct

1.4 Proposed Architecture

This architecture is a Mixture of above mentioned approaches Implements a 32 point 2D DCT through two 1D DCT passes.
The first pass computes the DCT row-wise.The output of the first pass is transposed and fed to the second pass, which computes the DCT column-wise.The overall design ensures that each step is modular and leverages butterfly arithmetic structures to minimize computational complexity. But Computing 1D is based on the Butterfly Approximation . Computed 1D DCT is an approximation of loefflers 8 point DCT computation but extented to 32 point by instantiating lower level modules .This architecture also has FSM to control the flow of design .which is explained below in detailed .

Key components include:
1. FSM for state control.
2. 1D DCT modules (dct32, dct16, dct8, dct4)
3. Scaling units
4. Transpose module
5. Counter for timing and synchronization

FSM (Finite State Machine):

The FSM coordinates the stages of the DCT pipeline, ensuring each computation occurs in the correct order. The FSM manages the following states:
Idle: The system waits for a load signal to start the DCT process.
1D_DCT: The first 1D DCT pass is applied to the input data (rows).
Transpose_Start: Intermediate results are loaded into the transpose buffer.
Transpose_Done: The transposed data is extracted from the buffer for the second DCT pass.
2D_DCT: The second 1D DCT pass (columns) is applied to the transposed data.
The FSM transitions are governed by the counter's output and the availability of input data, ensuring synchronization between stages.

1D DCT Modules (dct32, dct16, dct8, dct4)

The 1D DCT modules break the computation into smaller segments. Each module follows a butterfly structure to compute sums and differences of pairs of inputs.DCT32 Computes the 32-point DCT by splitting the input into 16-point segments . Internally calls DCT16 for these segments. The 16-point DCT follows a similar pattern, where inputs are split into smaller groups for parallel processing.Intermediate sums and differences are computed, and these values are passed to the next stage DCT8.The 8-point DCT processes smaller blocks of input. It uses butterfly arithmetic to compute the final DCT coefficients efficiently. This decomposition allows parallel processing and minimizes multiplications, following a method similar to the Loeffler algorithm.

Transpose Module

Purpose: Transposes the intermediate output from the first DCT pass TO prepare for the second DCT pass.he transpose operation reorders the data, ensuring that rows become columns for the second pass. FSM Control: The FSM triggers trans_start and trans_done to handle the transfer of data to/from the transpose buffer.

Scaling Module

Scaling Modules are used for Fixed point Approximations . scaling1: Scales the output from the first DCT stage. scaling2: Applies final scaling to the output after the second DCT stage. Both modules use bit-shifting for multiplication by powers of 2, which minimizes hardware

Shift-Add Modules

These modules implement multiplication using shift-and-add techniques to reduce the number of multipliers.These may cause some error in results but can give almost similar results This approach minimizes hardware complexity while ensuring high-speed operation.

1.5 Control and Data Flow Synchronization

The architecture ensures synchronization between different modules using the FSM and counter.

Data Flow:

- Input data flows into the first DCT stage (dct32).
- The intermediate results are scaled and transposed.
- The transposed data goes through the second DCT pass.

FSM Synchronization:

Each stage begins only after the previous one completes. The FSM uses the counter to determine when to move to the next stage. Control signals ensure that only the required modules are active at each stage, saving power.

Example Visualization of Butterfly Structure

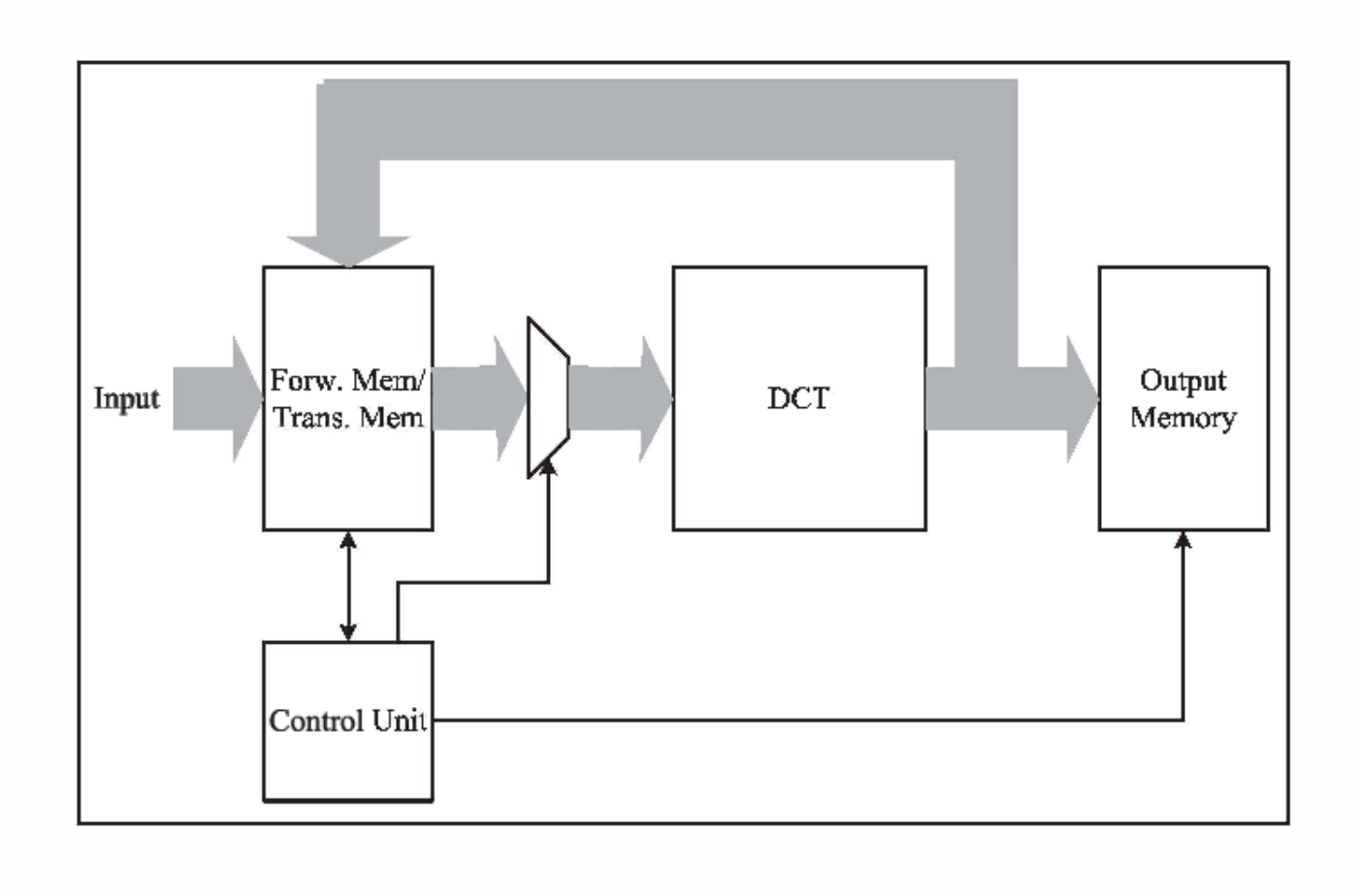


Figure 3: Overall outline of 2D Dct

1.6 RTL and Testbench

1.Verilog /System Verilog Code

```
module dct
#(
    // Parameters for bit-SIZEs at different stages in the pipeline
    parameter SIZE_IN = 16,          // Bit SIZE for the input signal x
    parameter SIZE_Y = 32,         // Bit SIZE for intermediate signals ... after first DCT
    parameter SIZE_X2 = 16,        // Bit SIZE for scaling input after ... first DCT
    parameter SIZE_Y2 = 32,        // Bit SIZE for intermediate signals ... after second DCT
    parameter SIZE_YOUT = 32,      // Bit SIZE for the final output ... signal y

    // FSM state definitions for the control signals in the pipeline
    parameter IDLE = 3'b000,
    parameter 1D_DCT = 3'b001,
    parameter TRANSPOSE_START = 3'b010,
    parameter TRANSPOSE_DONE = 3'b011,
    parameter 2D_DCT = 3'b100
)
(
    // Port definitions for the module
    input wire signed [SIZE_X-1:0] X_IN[0:31],  // 32-element input ... array, each element SIZE_X bits
    input wire clk, rst,                        // Clock and reset ... signals
    input wire load,                            // Load signal to start ... the DCT process
    input wire [1:0] size,                      // Size control for ... scaling

    output wire signed [SIZE_YOUT-1:0]Y_OUT[0:31] // 32-element output ... array, each element SIZE_YOUT bits
);

    // Intermediate signals for the first stage, scaling, transpose, and... second stage
    wire signed [SIZE_Y-1:0] stage1_out [0:31];   // Output of first DCT... stage
    wire signed [SIZE_Y-1:0] scaling_out [0:31]; // Output after first ... scaling
    wire signed [SIZE_Y2-1:0] stage2_out [0:31];  // Output of second ... DCT stage
    wire signed [SIZE_Y2-1:0] scaling_out [0:31];// Output after final ...
```

```
        scaling
32
33      // Control signals for the FSM to manage each stage in the pipeline
34      wire load_1dct;
35      wire transpose_start;
36      wire transpose_done;
37      wire load_2dct;
38
39      // FSM instance to manage control signals
40      // The FSM coordinates which stage to execute next in the 2D DCT ...
            pipeline.
41      fsm fsm1 (
42          .clk(clk),
43          .rst(rst),
44          .load(load),
45          .load_1dct(load_1dct),
46          .transpose_start(transpose_start),
47          .transpose_done(transpose_done),
48          .load_2dct(load_2dct)
49      );
50
51      // 1D DCT (First stage)
52      // This module applies the 1D DCT transform to the input data `x`.
53      // The output is stored in `stage1_out`.
54      dct32
55      #(
56          .SIZE_X(SIZE_X),
57          .SIZE_Y(SIZE_Y)
58      )
59      dct32_stage1 (
60          .clk(clk),
61          .rst(rst),
62          .load(load_1dct),
63          .x(x),
64          .y(stage1_out)
65      );
66
67      // Scaling after the first DCT stage
68      // Adjusts `stage1_out` based on the `size` parameter and outputs ...
            `scaling1_out`.
69      scaling1
70      #(
71          .SIZE(SIZE_Y)
72      )
73      scaling_stage1 (
74          .size(size),
75          .x(stage1_out),
76          .y(scaling1_out)
```

```
77      );
78
79      // Transpose operation between 1D DCTs
80      // Transposes `scaling1_out` to reorder the data, making it ready ...
            for the next DCT stage.
81      transpose
82      #(
83          .SIZE(SIZE_Y)
84      )
85      transpose1 (
86          .clk(clk),
87          .rst(rst),
88          .load(transpose_start),
89          .unload(transpose_done),
90          .x(scaling1_out),
91          .y(scaling2_out)
92      );
93
94      // 2D DCT (Second stage)
95      // Applies a second 1D DCT to the transposed data `scaling2_out`, ...
            completing the 2D DCT process.
96      dct32
97      #(
98          .SIZE_X(SIZE_Y2),
99          .SIZE_Y(SIZE_Y2)
100     )
101     dct32_stage2 (
102         .clk(clk),
103         .rst(rst),
104         .load(load_2dct),
105         .x(scaling2_out),
106         .y(stage2_out)
107     );
108
109     // Final scaling and output
110     // Adjusts `stage2_out` to produce the final output `y`.
111     scaling2
112     #(
113         .SIZE(SIZE_Y2)
114     )
115     final_scaling (
116         .size(size),
117         .x(stage2_out),
118         .y(y)
119     );
120
121 endmodule
122
```

```
123 module fsm (
124     input wire clk, rst,         // Clock and Reset signals
125     input wire load,             // Input signal to trigger the FSM
126 
127     output wire load_1dct,       // Signal to trigger 1D DCT computation
128     output wire transpose_start,     // Signal to trigger matrix ...
        transpose
129     output wire transpose_done,   // Signal to unload transposed matrix
130     output wire load_2dct        // Signal to trigger 2D DCT computation
131 );
132 
133     // State encoding using local parameters for readability
134     localparam IDLE = 3'd0,
135                1D_DCT = 3'd1,
136                transpose_start = 3'd2,
137                transpose_done = 3'd3,
138                TWO_DCT = 3'd4;
139 
140     // State and delay values for each stage
141     localparam DCT_DELAY = 8,         // Delay for 1D DCT operation
142                TRANS_DELAY = 32;      // Delay for transpose
143 
144     reg [2:0] state;                  // Current state of the FSM
145     reg [2:0] state_next;             // Next state of the FSM
146 
147     wire last;                        // Signal to indicate last count ...
        from counter
148     wire [8:0] count;                 // Counter output
149 
150     // Instantiate the counter to handle delays in each stage
151     counter counter1 (
152         .clk (clk),
153         .rst (rst),
154         .load (load),
155         .count (count),
156         .n (78),                      // Total number of clock cycles per...
            FSM cycle
157         .last (last)
158     );
159 
160     // Assign signals based on FSM states
161     assign load_1dct = (state == 1D_DCT) || (state == transpose_start);
162     assign transpose_start = (state == transpose_start);
163     assign transpose_done = (state == transpose_done);
164     assign load_2dct = (state == TWO_DCT);
165 
166     // Sequential logic to update the FSM state
167     always @(posedge clk) begin
```

```
168         if (rst) begin
169             state <= IDLE;              // Reset the FSM to IDLE state
170         end else begin
171             state <= state_next;        // Move to the next state
172         end
173     end
174 
175     // Combinational logic to determine the next FSM state
176     always @(*) begin
177         state_next = state;             // Default: maintain current state
178 
179         case (state)
180             IDLE: begin
181                 if (load) state_next = 1D_DCT;    // Transition from ...
                        IDLE to 1D DCT
182             end
183             1D_DCT: begin
184                 if (count == 8) state_next = transpose_start;  // Move ...
                        to matrix transpose after 1D DCT
185             end
186             transpose_start: begin
187                 if (count == 41) state_next = transpose_done;  // Wait ...
                        for transpose to complete
188             end
189             transpose_done: begin
190                 if (count == 73) state_next = TWO_DCT;   // Move to 2D ...
                        DCT after unloading
191             end
192             TWO_DCT: begin
193                 if (count == 80) state_next = IDLE;     // Return to ...
                        IDLE after 2D DCT
194             end
195             default: begin
196                 state_next = IDLE;  // If undefined state, reset to IDLE
197             end
198         endcase
199     end
200 
201 endmodule
202 
203 
204 
205 
206 
207 
208 
209 
210 
```

```
211 module dct32
212 (
213     input  wire [SIZE_X-1:0] x [0:31],  // Array of inputs x0 to x31
214     input  wire clk, rst, load,
215     output wire signed [SIZE_Y-1:0] y [0:31]  // Array of outputs y0 to ...
        y31
216 );
217 
218     // Internal registers
219     reg signed [SIZE_X-1:0] temp_in [0:31];
220     reg signed [SIZE_Y-1:0] temp1 [0:15];
221     reg signed [SIZE_Y-1:0] temp2 [0:15];
222 
223     wire signed [SIZE_Y-1:0] a_next [0:15];
224     wire signed [SIZE_Y-1:0] b_next [0:15];
225 
226     // Instantiate dct16
227     dct16
228     #(
229         .SIZE_X (SIZE_X ),
230         .SIZE_Y (SIZE_Y)
231     )
232     dct16_1 (
233         .clk      (clk),
234         .rst      (rst),
235         .load     (load),
236         .x        (temp1),  // Pass array to dct16 instance
237         .y        ({y[0], y[2], y[4], y[6], y[8], y[10], y[12], y[14], y...
            [16], y[18], y[20], y[22], y[24], y[26], y[28], y[30]})
238     );
239 
240     // Clock and reset logic
241     always_ff @(posedge clk) begin
242         if (rst) begin
243             temp_in <= '{default: 0};
244             temp1 <= '{default: 0};
245             temp2 <= '{default: 0};
246         end else if (load) begin
247             temp_in <= x;  // Load all inputs
248         end else begin
249             temp1 <= a_next;  // Update temp1 values
250             temp2 <= b_next;  // Update temp2 values
251         end
252     end
253 
254     // Generate the next values for temp1 and temp2
255     assign a_next = '{
256         temp_in[0] + temp_in[31], temp_in[1] + temp_in[30], temp_in[2] +...
```

```
            temp_in[29], temp_in[3] + temp_in[28],
257         temp_in[4] + temp_in[27], temp_in[5] + temp_in[26], temp_in[6] +...
            temp_in[25], temp_in[7] + temp_in[24],
258         temp_in[8] + temp_in[23], temp_in[9] + temp_in[22], temp_in[10] ...
            + temp_in[21], temp_in[11] + temp_in[20],
259         temp_in[12] + temp_in[19], temp_in[13] + temp_in[18], temp_in...
            [14] + temp_in[17], temp_in[15] + temp_in[16]
260     };
261
262     assign b_next = '{
263         temp_in[0] - temp_in[31], temp_in[1] - temp_in[30], temp_in[2] -...
            temp_in[29], temp_in[3] - temp_in[28],
264         temp_in[4] - temp_in[27], temp_in[5] - temp_in[26], temp_in[6] -...
            temp_in[25], temp_in[7] - temp_in[24],
265         temp_in[8] - temp_in[23], temp_in[9] - temp_in[22], temp_in[10] ...
            - temp_in[21], temp_in[11] - temp_in[20],
266         temp_in[12] - temp_in[19], temp_in[13] - temp_in[18], temp_in...
            [14] - temp_in[17], temp_in[15] - temp_in[16]
267     };
268
269     // Instantiate shift_add32
270     shift_add32
271     #(
272         .SIZE (SIZE_Y)
273     )
274     shift_addder(
275         .clk   (clk),
276         .rst   (rst),
277         .b0    (temp2[0]), .b1  (temp2[1]),  .b2  (temp2[2]),  .b3  (...
            temp2[3]),
278         .b4    (temp2[4]), .b5  (temp2[5]),  .b6  (temp2[6]),  .b7  (...
            temp2[7]),
279         .b8    (temp2[8]), .b9  (temp2[9]),  .b10 (temp2[10]), .b11 (...
            temp2[11]),
280         .b12   (temp2[12]), .b13 (temp2[13]), .b14 (temp2[14]), .b15 (...
            temp2[15]),
281         .y1    (y[1]), .y3    (y[3]), .y5    (y[5]), .y7    (y[7]),
282         .y9    (y[9]), .y11   (y[11]), .y13   (y[13]), .y15   (y[15]),
283         .y17   (y[17]), .y19   (y[19]), .y21   (y[21]), .y23   (y[23]),
284         .y25   (y[25]), .y27   (y[27]), .y29   (y[29]), .y31   (y[31])
285     );
286
287 endmodule
288
289
290
291
292
```

```
293 module DCT4(
294     input wire signed [SIZE_X-1:0] x [0:3],      // Array of inputs x0 to...
        x3
295     input wire clk, rst, load,
296     output reg signed [SIZE_Y-1:0] y [0:3]       // Array of outputs y0 ...
        to y3
297 );
298 
299     // Internal registers for storing inputs
300     reg signed [SIZE_X-1:0] temp1 [0:3];         // Register array for ...
        inputs x0 to x3
301 
302     // Intermediate wires
303     wire signed [SIZE_Y-1:0] a [0:1];
304     wire signed [SIZE_Y-1:0] b [0:1];
305     wire signed [SIZE_Y-1:0] t0_64, t0_83, t0_36, t1_64, t1_83, t1_36;
306 
307     // Next-state wires for outputs
308     wire signed [SIZE_Y-1:0] y_d [0:3];
309 
310     // Clock and reset logic for temp1 and output y
311     always_ff @(posedge clk) begin
312         if (rst) begin
313             y <= '{default: 0};
314             temp1 <= '{default: 0};
315         end else begin
316             y <= y_d;                   // Update outputs on each clock ...
                cycle
317             if (load) temp1 <= x;       // Load inputs on load signal
318         end
319     end
320 
321     // Calculate a and b intermediate values
322     assign a[0] = temp1[0] + temp1[3];
323     assign a[1] = temp1[1] + temp1[2];
324     assign b[0] = temp1[0] - temp1[3];
325     assign b[1] = temp1[1] - temp1[2];
326 
327     // Calculate t terms using shifts for scaling
328     assign t0_64 = {a[0], 6'b000000};
329     assign t0_83 = {b[0], 6'b000000} + {b[0], 4'b0000} + {b[0], 1'b0} + ...
        b[0];
330     assign t0_36 = {b[0], 5'b00000} + {b[0], 2'b00};
331     assign t1_64 = {a[1], 6'b000000};
332     assign t1_83 = {b[1], 6'b000000} + {b[1], 4'b0000} + {b[1], 1'b0} + ...
        b[1];
333     assign t1_36 = {b[1], 5'b00000} + {b[1], 2'b00};
334 
```

```
335     // Compute y_d based on the intermediate terms
336     assign y_d[0] = t0_64 + t1_64;
337     assign y_d[1] = t1_36 + t0_83;
338     assign y_d[2] = t0_64 - t1_64;
339     assign y_d[3] = t0_36 + t1_83;
340 endmodule
341 
342 module shift_adder(
343     input logic [SIZE-1:0] b,       // Input array
344     output logic [SIZE-1:0] y1,      // Output signals
345     output logic [SIZE-1:0] y3,
346     output logic [SIZE-1:0] y5,
347     output logic [SIZE-1:0] y7,
348     output logic [SIZE-1:0] y9,
349     output logic [SIZE-1:0] y11,
350     output logic [SIZE-1:0] y13,
351     output logic [SIZE-1:0] y15
352 );
353     logic [15:0] y1_d, y3_d, y5_d, y7_d, y9_d, y11_d, y13_d, y15_d;
354 
355     // Function to calculate the combined value based on the input
356     function logic [SIZE-1:0] calculate_y(logic [15:0] b);
357         logic [15:0] result;
358         result = 0;
359 
360         foreach (b[i]) begin
361             // Bit shifting based on index
362             result += {b[i], 1'b0} << (i + 1);
363             result += {b[i], 2'b00} << (i + 2);
364             result += {b[i], 3'b000} << (i + 3);
365             result += {b[i], 4'b0000} << (i + 4);
366             result += {b[i], 5'b00000} << (i + 5);
367             result += {b[i], 6'b000000} << (i + 6);
368         end
369         return result;
370     endfunction
371     // Assign calculated values to y_d variables
372     always_comb begin
373         y1_d = calculate_y(b);
374         y3_d = calculate_y(b);
375         y5_d = calculate_y(b);
376         y7_d = calculate_y(b);
377         y9_d = calculate_y(b);
378         y11_d = calculate_y(b);
379         y13_d = calculate_y(b);
380         y15_d = calculate_y(b);
381     end
382     // Output assignments
```

```
383     always_comb begin
384         y1 = y1_d[6:0];
385         y3 = y3_d[6:0];
386         y5 = y5_d[6:0];
387         y7 = y7_d[6:0];
388         y9 = y9_d[6:0];
389         y11 = y11_d[6:0];
390         y13 = y13_d[6:0];
391         y15 = y15_d[6:0];
392     end
393 endmodule
394 
395 module scaling
396 (
397     input wire [1:0] size,
398     input wire signed [SIZE-1:0] x[0:31],
399     output reg signed [SIZE-1:0] y[0:31]
400 );
401     integer i;
402     always @(*) begin
403         for (i = 0; i < 32; i = i + 1) begin
404             case (size)
405                 2'b00: y[i] = {x[i][SIZE-1], x[i][SIZE-1:1]}; // Scale ...
                       down by 2
406                 2'b01: y[i] = {x[i][SIZE-1], x[i][SIZE-1], x[i][SIZE...
                       -1:2]}; // Scale down by 3
407                 2'b10: y[i] = {x[i][SIZE-1], x[i][SIZE-1], x[i][SIZE-1],...
                        x[i][SIZE-1:3]}; // Scale down by 4
408                 2'b11: y[i] = {x[i][SIZE-1], x[i][SIZE-1], x[i][SIZE-1],...
                        x[i][SIZE-1], x[i][SIZE-1:4]}; // Scale down by 5
409                 default: y[i] = x[i]; // Default case (no scaling)
410             endcase
411         end
412     end
413 
414 endmodule
```

Testbench

1.Verilog /System Verilog Code

```
1 `timescale 1ns / 100ps
2 module dct_test();
3 // constants
4 // test vector input registers
5 reg clk;
6 reg [1:0] size;
```

```
 7 reg load;
 8 reg rst;
 9 reg [9:0] x[0];
10 reg [9:0] x[1];
11 reg [9:0] x[2];
12 reg [9:0] x[3];
13 reg [9:0] x[4];
14 reg [9:0] x[5];
15 reg [9:0] x[6];
16 reg [9:0] x[7];
17 reg [9:0] x[8];
18 reg [9:0] x[9];
19 reg [9:0] x[10];
20 reg [9:0] x[11];
21 reg [9:0] x[12];
22 reg [9:0] x[13];
23 reg [9:0] x[14];
24 reg [9:0] x[15];
25 reg [9:0] x[16];
26 reg [9:0] x[17];
27 reg [9:0] x[18];
28 reg [9:0] x[19];
29 reg [9:0] x[20];
30 reg [9:0] x[21];
31 reg [9:0] x[22];
32 reg [9:0] x[23];
33 reg [9:0] x[24];
34 reg [9:0] x[25];
35 reg [9:0] x[26];
36 reg [9:0] x[27];
37 reg [9:0] x[28];
38 reg [9:0] x[29];
39 reg [9:0] x[30];
40 reg [9:0] x[31];
41 // wires
42 wire [15:0]  y0;
43 wire [15:0]  y[11];
44 wire [15:0]  y[2];
45 wire [15:0]  y[3];
46 wire [15:0]  y[4];
47 wire [15:0]  y[5];
48 wire [15:0]  y[6];
49 wire [15:0]  y[7];
50 wire [15:0]  y[8];
51 wire [15:0]  y[9];
52 wire [15:0]  y[10];
53 wire [15:0]  y[11];
54 wire [15:0]  y[12];
```

```
55  wire [15:0]  y[13];
56  wire [15:0]  y[14];
57  wire [15:0]  y[15];
58  wire [15:0]  y[16];
59  wire [15:0]  y[17];
60  wire [15:0]  y[18];
61  wire [15:0]  y[19];
62  wire [15:0]  y[20];
63  wire [15:0]  y[21];
64  wire [15:0]  y[22];
65  wire [15:0]  y[23];
66  wire [15:0]  y[24];
67  wire [15:0]  y[25];
68  wire [15:0]  y[26];
69  wire [15:0]  y[27];
70  wire [15:0]  y[28];
71  wire [15:0]  y[29];
72  wire [15:0]  y[30];
73  wire [15:0]  y[31];
74
75  // assign statements (if any)
76  dct i1 (
77  // port map - connection between master ports and signals/registers
78    .clk(clk),
79    .size(size),
80    .load(load),
81    .rst(rst),
82    .x[0](x[0]),
83    .x[1](x[1]),
84    .x[2](x[2]),
85    .x[3](x[3]),
86    .x[4](x[4]),
87    .x[5](x[5]),
88    .x[6](x[6]),
89    .x[7](x[7]),
90    .x[8](x[8]),
91    .x[9](x[9]),
92    .x[10](x[10]),
93    .x[11](x[11]),
94    .x[12](x[12]),
95    .x[13](x[13]),
96    .x[14](x[14]),
97    .x[15](x[15]),
98    .x[16](x[16]),
99    .x[17](x[17]),
100   .x[18](x[18]),
101   .x[19](x[19]),
102   .x[20](x[20]),
```

```
103     .x[21](x[21]),
104     .x[22](x[22]),
105     .x[23](x[23]),
106     .x[24](x[24]),
107     .x[25](x[25]),
108     .x[26](x[26]),
109     .x[27](x[27]),
110     .x[28](x[28]),
111     .x[29](x[29]),
112     .x[30](x[30]),
113     .x[31](x[31]),
114     .y0(y0),
115     .y[11](y[11]),
116     .y[2](y[2]),
117     .y[3](y[3]),
118     .y[4](y[4]),
119     .y[5](y[5]),
120     .y[6](y[6]),
121     .y[7](y[7]),
122     .y[8](y[8]),
123     .y[9](y[9]),
124     .y[10](y[10]),
125     .y[11](y[11]),
126     .y[12](y[12]),
127     .y[13](y[13]),
128     .y[14](y[14]),
129     .y[15](y[15]),
130     .y[16](y[16]),
131     .y[17](y[17]),
132     .y[18](y[18]),
133     .y[19](y[19]),
134     .y[20](y[20]),
135     .y[21](y[21]),
136     .y[22](y[22]),
137     .y[23](y[23]),
138     .y[24](y[24]),
139     .y[25](y[25]),
140     .y[26](y[26]),
141     .y[27](y[27]),
142     .y[28](y[28]),
143     .y[29](y[29]),
144     .y[30](y[30]),
145     .y[31](y[31])
146 );
147 
148     always #0.625  1.25ns
149       clk = ¬clk;
150 
```

```
151
152   initial
153
154     begin
155     $display($time, " << Starting the Simulation >>");
156     clk = 1'b0;
157     // at time 0
158     rst = 1'b1;
159     // reset is active
160     load = 1'b0;size = 2'b00;
161     // load disabled
162     x[0] = 8'h0; x[1] = 8'h0; x[2] = 8'h0; x[3] = 8'h0; x[4] = 8'h0; x...
           [5] = 8'h0; x[6] = 8'h0; x[7] = 8'h0;
163     x[8] = 8'h0; x[9] = 8'h0; x[10] = 8'h0; x[11] = 8'h0; x[12] = 8'h0; ...
           x[13] = 8'h0; x[14] = 8'h0; x[15] = 8'h0;
164     x[16] = 8'h0; x[17] = 8'h0; x[18] = 8'h0; x[19] = 8'h0; x[20] = 8'h0...
           ; x[21] = 8'h0; x[22] = 8'h0; x[23] = 8'h0;
165     x[24] = 8'h0; x[25] = 8'h0; x[26] = 8'h0; x[27] = 8'h0; x[28] = 8'h0...
           ; x[29] = 8'h0; x[30] = 8'h0; x[31] = 8'h0;
166
167     #4 rst = 1'b0;
168
169     $display($time, " << Coming out of reset >>");
170     //load = 1'b1;x[0]= 8'h1; x[1] = 8'h1; x[2] = 8'h1; x[3] = 8'h1; x...
           [4] = 8'h1; x[5] = 8'h1; x[6] = 8'h1; x[7] = 8'h1;
171     @(negedge clk);
172
173
174
175     //Columna: 0
176     x[0] = 16'd169;x[1] = 16'd158;x[2] = 16'd167;x[3] = 16'd176;x[4] = ...
           16'd149;x[5] = 16'd104;x[6] = 16'd86;x[7] = 16'd91;x[8] = 16'd102...
           ;x[9] = 16'd95;x[10] = 16'd94;x[11] = 16'd102;x[12] = 16'd109;x...
           [13] = 16'd108;x[14] = 16'd103;x[15] = 16'd100;x[16] = 16'd104;x...
           [17] = 16'd89;x[18] = 16'd79;x[19] = 16'd77;x[20] = 16'd78;x[21] ...
           = 16'd70;x[22] = 16'd53;x[23] = 16'd45;x[24] = 16'd61;x[25] = 16'...
           d127;x[26] = 16'd146;x[27] = 16'd103;x[28] = 16'd84;x[29] = 16'...
           d108;x[30] = 16'd114;x[31] = 16'd93;#4;
177     //Columna: 1
178     x[0] = 16'd157;x[1] = 16'd164;x[2] = 16'd165;x[3] = 16'd142;x[4] = ...
           16'd107;x[5] = 16'd92;x[6] = 16'd99;x[7] = 16'd107;x[8] = 16'd105...
           ;x[9] = 16'd110;x[10] = 16'd116;x[11] = 16'd116;x[12] = 16'd109;x...
           [13] = 16'd102;x[14] = 16'd102;x[15] = 16'd105;x[16] = 16'd95;x...
           [17] = 16'd95;x[18] = 16'd73;x[19] = 16'd61;x[20] = 16'd83;x[21] ...
           = 16'd95;x[22] = 16'd91;x[23] = 16'd95;x[24] = 16'd68;x[25] = 16'...
           d86;x[26] = 16'd111;x[27] = 16'd131;x[28] = 16'd138;x[29] = 16'...
           d135;x[30] = 16'd147;x[31] = 16'd171;#4;
179     //Columna: 2
```

```
180     x[0] = 16'd182;x[1] = 16'd178;x[2] = 16'd173;x[3] = 16'd169;x[4] = ...
           16'd169;x[5] = 16'd179;x[6] = 16'd184;x[7] = 16'd177;x[8] = 16'...
           d182;x[9] = 16'd184;x[10] = 16'd185;x[11] = 16'd183;x[12] = 16'...
           d181;x[13] = 16'd181;x[14] = 16'd182;x[15] = 16'd182;x[16] = 16'...
           d176;x[17] = 16'd187;x[18] = 16'd177;x[19] = 16'd170;x[20] = 16'...
           d183;x[21] = 16'd187;x[22] = 16'd186;x[23] = 16'd190;x[24] = 16'...
           d171;x[25] = 16'd156;x[26] = 16'd144;x[27] = 16'd147;x[28] = 16'...
           d148;x[29] = 16'd142;x[30] = 16'd153;x[31] = 16'd179;#4;
181     //Columna: 3
182     x[0] = 16'd148;x[1] = 16'd134;x[2] = 16'd131;x[3] = 16'd140;x[4] = ...
           16'd145;x[5] = 16'd146;x[6] = 16'd142;x[7] = 16'd135;x[8] = 16'...
           d142;x[9] = 16'd147;x[10] = 16'd148;x[11] = 16'd147;x[12] = 16'...
           d149;x[13] = 16'd155;x[14] = 16'd158;x[15] = 16'd156;x[16] = 16'...
           d164;x[17] = 16'd162;x[18] = 16'd161;x[19] = 16'd163;x[20] = 16'...
           d159;x[21] = 16'd160;x[22] = 16'd169;x[23] = 16'd166;x[24] = 16'...
           d181;x[25] = 16'd190;x[26] = 16'd190;x[27] = 16'd187;x[28] = 16'...
           d192;x[29] = 16'd194;x[30] = 16'd182;x[31] = 16'd171;#4;
183     //Columna: 4
184     x[0] = 16'd103;x[1] = 16'd96;x[2] = 16'd92;x[3] = 16'd85;x[4] = 16'...
           d76;x[5] = 16'd75;x[6] = 16'd78;x[7] = 16'd77;x[8] = 16'd75;x[9] ...
           = 16'd85;x[10] = 16'd90;x[11] = 16'd85;x[12] = 16'd78;x[13] = 16'...
           d77;x[14] = 16'd80;x[15] = 16'd81;x[16] = 16'd79;x[17] = 16'd94;x...
           [18] = 16'd96;x[19] = 16'd99;x[20] = 16'd100;x[21] = 16'd103;x...
           [22] = 16'd104;x[23] = 16'd88;x[24] = 16'd84;x[25] = 16'd80;x[26]...
            = 16'd72;x[27] = 16'd69;x[28] = 16'd70;x[29] = 16'd64;x[30] = ...
           16'd57;x[31] = 16'd59;#4;
185     //Columna: 5
186     x[0] = 16'd111;x[1] = 16'd102;x[2] = 16'd97;x[3] = 16'd98;x[4] = 16'...
           d99;x[5] = 16'd103;x[6] = 16'd101;x[7] = 16'd93;x[8] = 16'd102;x...
           [9] = 16'd97;x[10] = 16'd92;x[11] = 16'd90;x[12] = 16'd93;x[13] =...
            16'd99;x[14] = 16'd100;x[15] = 16'd98;x[16] = 16'd81;x[17] = 16'...
           d119;x[18] = 16'd116;x[19] = 16'd102;x[20] = 16'd100;x[21] = 16'...
           d91;x[22] = 16'd79;x[23] = 16'd62;x[24] = 16'd55;x[25] = 16'd42;x...
           [26] = 16'd32;x[27] = 16'd36;x[28] = 16'd40;x[29] = 16'd35;x[30] ...
           = 16'd43;x[31] = 16'd65;#4;
187     //Columna: 6
188     x[0] = 16'd116;x[1] = 16'd106;x[2] = 16'd106;x[3] = 16'd111;x[4] = ...
           16'd110;x[5] = 16'd107;x[6] = 16'd104;x[7] = 16'd98;x[8] = 16'd94...
           ;x[9] = 16'd87;x[10] = 16'd81;x[11] = 16'd83;x[12] = 16'd91;x[13]...
            = 16'd98;x[14] = 16'd102;x[15] = 16'd104;x[16] = 16'd124;x[17] =...
            16'd118;x[18] = 16'd92;x[19] = 16'd81;x[20] = 16'd66;x[21] = 16'...
           d54;x[22] = 16'd65;x[23] = 16'd63;x[24] = 16'd78;x[25] = 16'd74;x...
           [26] = 16'd68;x[27] = 16'd66;x[28] = 16'd64;x[29] = 16'd57;x[30] ...
           = 16'd59;x[31] = 16'd73;#4;
189     //Columna: 7
190     x[0] = 16'd129;x[1] = 16'd126;x[2] = 16'd125;x[3] = 16'd119;x[4] = ...
           16'd111;x[5] = 16'd115;x[6] = 16'd128;x[7] = 16'd134;x[8] = 16'...
           d140;x[9] = 16'd153;x[10] = 16'd163;x[11] = 16'd155;x[12] = 16'...
```

```
        d130;x[13] = 16'd112;x[14] = 16'd118;x[15] = 16'd133;x[16] = 16'...
        d165;x[17] = 16'd94;x[18] = 16'd72;x[19] = 16'd107;x[20] = 16'd93...
        ;x[21] = 16'd80;x[22] = 16'd108;x[23] = 16'd103;x[24] = 16'd90;x...
        [25] = 16'd74;x[26] = 16'd55;x[27] = 16'd46;x[28] = 16'd45;x[29] ...
        = 16'd49;x[30] = 16'd68;x[31] = 16'd95;#4;
191     //Columna: 8
192     x[0] = 16'd138;x[1] = 16'd131;x[2] = 16'd128;x[3] = 16'd129;x[4] = ...
        16'd122;x[5] = 16'd126;x[6] = 16'd114;x[7] = 16'd124;x[8] = 16'...
        d119;x[9] = 16'd149;x[10] = 16'd170;x[11] = 16'd184;x[12] = 16'...
        d191;x[13] = 16'd170;x[14] = 16'd139;x[15] = 16'd120;x[16] = 16'...
        d71;x[17] = 16'd65;x[18] = 16'd62;x[19] = 16'd66;x[20] = 16'd106;...
        x[21] = 16'd92;x[22] = 16'd88;x[23] = 16'd122;x[24] = 16'd118;x...
        [25] = 16'd91;x[26] = 16'd63;x[27] = 16'd58;x[28] = 16'd58;x[29] ...
        = 16'd60;x[30] = 16'd56;x[31] = 16'd87;#4;
193     //Columna: 9
194     x[0] = 16'd139;x[1] = 16'd129;x[2] = 16'd125;x[3] = 16'd129;x[4] = ...
        16'd121;x[5] = 16'd119;x[6] = 16'd104;x[7] = 16'd109;x[8] = 16'...
        d104;x[9] = 16'd86;x[10] = 16'd103;x[11] = 16'd127;x[12] = 16'...
        d120;x[13] = 16'd130;x[14] = 16'd135;x[15] = 16'd96;x[16] = 16'...
        d69;x[17] = 16'd89;x[18] = 16'd60;x[19] = 16'd39;x[20] = 16'd42;x...
        [21] = 16'd119;x[22] = 16'd162;x[23] = 16'd127;x[24] = 16'd113;x...
        [25] = 16'd129;x[26] = 16'd122;x[27] = 16'd104;x[28] = 16'd97;x...
        [29] = 16'd108;x[30] = 16'd92;x[31] = 16'd92;#4;
195     //Columna: 10
196     x[0] = 16'd139;x[1] = 16'd134;x[2] = 16'd129;x[3] = 16'd128;x[4] = ...
        16'd117;x[5] = 16'd120;x[6] = 16'd114;x[7] = 16'd121;x[8] = 16'...
        d119;x[9] = 16'd125;x[10] = 16'd130;x[11] = 16'd116;x[12] = 16'...
        d130;x[13] = 16'd134;x[14] = 16'd84;x[15] = 16'd51;x[16] = 16'd80...
        ;x[17] = 16'd93;x[18] = 16'd102;x[19] = 16'd74;x[20] = 16'd78;x...
        [21] = 16'd168;x[22] = 16'd144;x[23] = 16'd49;x[24] = 16'd73;x...
        [25] = 16'd113;x[26] = 16'd136;x[27] = 16'd131;x[28] = 16'd101;x...
        [29] = 16'd79;x[30] = 16'd61;x[31] = 16'd75;#4;
197     //Columna: 11
198     x[0] = 16'd139;x[1] = 16'd135;x[2] = 16'd129;x[3] = 16'd131;x[4] = ...
        16'd120;x[5] = 16'd128;x[6] = 16'd129;x[7] = 16'd133;x[8] = 16'...
        d126;x[9] = 16'd143;x[10] = 16'd138;x[11] = 16'd135;x[12] = 16'...
        d133;x[13] = 16'd83;x[14] = 16'd40;x[15] = 16'd59;x[16] = 16'd49;...
        x[17] = 16'd28;x[18] = 16'd42;x[19] = 16'd104;x[20] = 16'd143;x...
        [21] = 16'd71;x[22] = 16'd31;x[23] = 16'd45;x[24] = 16'd47;x[25] ...
        = 16'd62;x[26] = 16'd74;x[27] = 16'd81;x[28] = 16'd53;x[29] = 16'...
        d25;x[30] = 16'd30;x[31] = 16'd76;#4;
199     //Columna: 12
200     x[0] = 16'd139;x[1] = 16'd127;x[2] = 16'd125;x[3] = 16'd143;x[4] = ...
        16'd135;x[5] = 16'd132;x[6] = 16'd129;x[7] = 16'd134;x[8] = 16'...
        d148;x[9] = 16'd147;x[10] = 16'd141;x[11] = 16'd151;x[12] = 16'...
        d102;x[13] = 16'd50;x[14] = 16'd62;x[15] = 16'd51;x[16] = 16'd31;...
        x[17] = 16'd54;x[18] = 16'd129;x[19] = 16'd142;x[20] = 16'd73;x...
        [21] = 16'd23;x[22] = 16'd51;x[23] = 16'd49;x[24] = 16'd47;x[25] ...
```

```
        = 16'd46;x[26] = 16'd36;x[27] = 16'd49;x[28] = 16'd54;x[29] = 16'...
        d61;x[30] = 16'd81;x[31] = 16'd120;#4;
201     //Columna: 13
202     x[0] = 16'd139;x[1] = 16'd122;x[2] = 16'd127;x[3] = 16'd164;x[4] = ...
        16'd153;x[5] = 16'd135;x[6] = 16'd142;x[7] = 16'd163;x[8] = 16'...
        d171;x[9] = 16'd168;x[10] = 16'd155;x[11] = 16'd107;x[12] = 16'...
        d68;x[13] = 16'd60;x[14] = 16'd50;x[15] = 16'd44;x[16] = 16'd113;...
        x[17] = 16'd180;x[18] = 16'd140;x[19] = 16'd48;x[20] = 16'd44;x...
        [21] = 16'd86;x[22] = 16'd55;x[23] = 16'd59;x[24] = 16'd52;x[25] ...
        = 16'd60;x[26] = 16'd60;x[27] = 16'd84;x[28] = 16'd92;x[29] = 16'...
        d94;x[30] = 16'd107;x[31] = 16'd138;#4;
203     //Columna: 14
204     x[0] = 16'd140;x[1] = 16'd122;x[2] = 16'd131;x[3] = 16'd179;x[4] = ...
        16'd175;x[5] = 16'd160;x[6] = 16'd172;x[7] = 16'd195;x[8] = 16'...
        d184;x[9] = 16'd176;x[10] = 16'd150;x[11] = 16'd79;x[12] = 16'd51...
        ;x[13] = 16'd56;x[14] = 16'd72;x[15] = 16'd151;x[16] = 16'd223;x...
        [17] = 16'd152;x[18] = 16'd90;x[19] = 16'd121;x[20] = 16'd130;x...
        [21] = 16'd129;x[22] = 16'd105;x[23] = 16'd57;x[24] = 16'd51;x...
        [25] = 16'd71;x[26] = 16'd79;x[27] = 16'd106;x[28] = 16'd112;x...
        [29] = 16'd110;x[30] = 16'd121;x[31] = 16'd147;#4;
205     //Columna: 15
206     x[0] = 16'd142;x[1] = 16'd122;x[2] = 16'd127;x[3] = 16'd183;x[4] = ...
        16'd197;x[5] = 16'd193;x[6] = 16'd192;x[7] = 16'd190;x[8] = 16'...
        d179;x[9] = 16'd176;x[10] = 16'd151;x[11] = 16'd110;x[12] = 16'...
        d44;x[13] = 16'd52;x[14] = 16'd158;x[15] = 16'd213;x[16] = 16'd91...
        ;x[17] = 16'd123;x[18] = 16'd176;x[19] = 16'd156;x[20] = 16'd135;...
        x[21] = 16'd144;x[22] = 16'd144;x[23] = 16'd95;x[24] = 16'd94;x...
        [25] = 16'd118;x[26] = 16'd109;x[27] = 16'd117;x[28] = 16'd132;x...
        [29] = 16'd150;x[30] = 16'd146;x[31] = 16'd136;#4;
207     //Columna: 16
208     x[0] = 16'd140;x[1] = 16'd129;x[2] = 16'd115;x[3] = 16'd160;x[4] = ...
        16'd206;x[5] = 16'd201;x[6] = 16'd198;x[7] = 16'd199;x[8] = 16'...
        d184;x[9] = 16'd177;x[10] = 16'd156;x[11] = 16'd61;x[12] = 16'd68...
        ;x[13] = 16'd162;x[14] = 16'd210;x[15] = 16'd126;x[16] = 16'd61;x...
        [17] = 16'd154;x[18] = 16'd186;x[19] = 16'd171;x[20] = 16'd160;x...
        [21] = 16'd141;x[22] = 16'd132;x[23] = 16'd138;x[24] = 16'd132;x...
        [25] = 16'd147;x[26] = 16'd139;x[27] = 16'd146;x[28] = 16'd148;x...
        [29] = 16'd137;x[30] = 16'd146;x[31] = 16'd143;#4;
209     //Columna: 17
210     x[0] = 16'd127;x[1] = 16'd131;x[2] = 16'd124;x[3] = 16'd116;x[4] = ...
        16'd187;x[5] = 16'd239;x[6] = 16'd204;x[7] = 16'd200;x[8] = 16'...
        d197;x[9] = 16'd186;x[10] = 16'd133;x[11] = 16'd64;x[12] = 16'...
        d169;x[13] = 16'd204;x[14] = 16'd166;x[15] = 16'd92;x[16] = 16'...
        d94;x[17] = 16'd162;x[18] = 16'd177;x[19] = 16'd152;x[20] = 16'...
        d149;x[21] = 16'd129;x[22] = 16'd119;x[23] = 16'd165;x[24] = 16'...
        d151;x[25] = 16'd131;x[26] = 16'd144;x[27] = 16'd140;x[28] = 16'...
        d129;x[29] = 16'd141;x[30] = 16'd146;x[31] = 16'd153;#4;
211     //Columna: 18
```

```
212     x[0] = 16'd135;x[1] = 16'd144;x[2] = 16'd127;x[3] = 16'd104;x[4] = ...
           16'd130;x[5] = 16'd207;x[6] = 16'd240;x[7] = 16'd202;x[8] = 16'...
           d188;x[9] = 16'd188;x[10] = 16'd150;x[11] = 16'd173;x[12] = 16'...
           d199;x[13] = 16'd169;x[14] = 16'd170;x[15] = 16'd164;x[16] = 16'...
           d122;x[17] = 16'd135;x[18] = 16'd123;x[19] = 16'd98;x[20] = 16'...
           d132;x[21] = 16'd159;x[22] = 16'd147;x[23] = 16'd174;x[24] = 16'...
           d174;x[25] = 16'd162;x[26] = 16'd162;x[27] = 16'd158;x[28] = 16'...
           d150;x[29] = 16'd141;x[30] = 16'd136;x[31] = 16'd155;#4;
213     //Columna: 19
214     x[0] = 16'd121;x[1] = 16'd111;x[2] = 16'd115;x[3] = 16'd125;x[4] = ...
           16'd80;x[5] = 16'd85;x[6] = 16'd186;x[7] = 16'd239;x[8] = 16'd201...
           ;x[9] = 16'd175;x[10] = 16'd174;x[11] = 16'd201;x[12] = 16'd180;x...
           [13] = 16'd200;x[14] = 16'd200;x[15] = 16'd203;x[16] = 16'd185;x...
           [17] = 16'd182;x[18] = 16'd188;x[19] = 16'd176;x[20] = 16'd183;x...
           [21] = 16'd187;x[22] = 16'd130;x[23] = 16'd86;x[24] = 16'd160;x...
           [25] = 16'd218;x[26] = 16'd173;x[27] = 16'd150;x[28] = 16'd164;x...
           [29] = 16'd141;x[30] = 16'd144;x[31] = 16'd152;#4;
215     //Columna: 20
216     x[0] = 16'd134;x[1] = 16'd110;x[2] = 16'd117;x[3] = 16'd116;x[4] = ...
           16'd106;x[5] = 16'd85;x[6] = 16'd93;x[7] = 16'd169;x[8] = 16'd207...
           ;x[9] = 16'd180;x[10] = 16'd189;x[11] = 16'd207;x[12] = 16'd178;x...
           [13] = 16'd198;x[14] = 16'd242;x[15] = 16'd198;x[16] = 16'd102;x...
           [17] = 16'd134;x[18] = 16'd183;x[19] = 16'd183;x[20] = 16'd144;x...
           [21] = 16'd99;x[22] = 16'd46;x[23] = 16'd19;x[24] = 16'd85;x[25] ...
           = 16'd217;x[26] = 16'd222;x[27] = 16'd190;x[28] = 16'd188;x[29] =...
            16'd171;x[30] = 16'd167;x[31] = 16'd151;#4;
217     //Columna: 21
218     x[0] = 16'd167;x[1] = 16'd163;x[2] = 16'd146;x[3] = 16'd135;x[4] = ...
           16'd158;x[5] = 16'd174;x[6] = 16'd151;x[7] = 16'd132;x[8] = 16'...
           d161;x[9] = 16'd196;x[10] = 16'd208;x[11] = 16'd204;x[12] = 16'...
           d117;x[13] = 16'd82;x[14] = 16'd139;x[15] = 16'd97;x[16] = 16'd41...
           ;x[17] = 16'd73;x[18] = 16'd90;x[19] = 16'd80;x[20] = 16'd58;x...
           [21] = 16'd37;x[22] = 16'd43;x[23] = 16'd73;x[24] = 16'd37;x[25] ...
           = 16'd139;x[26] = 16'd234;x[27] = 16'd239;x[28] = 16'd217;x[29] =...
            16'd200;x[30] = 16'd185;x[31] = 16'd183;#4;
219     //Columna: 22
220     x[0] = 16'd160;x[1] = 16'd168;x[2] = 16'd156;x[3] = 16'd161;x[4] = ...
           16'd139;x[5] = 16'd119;x[6] = 16'd138;x[7] = 16'd124;x[8] = 16'...
           d149;x[9] = 16'd210;x[10] = 16'd222;x[11] = 16'd164;x[12] = 16'...
           d127;x[13] = 16'd110;x[14] = 16'd56;x[15] = 16'd71;x[16] = 16'd71...
           ;x[17] = 16'd80;x[18] = 16'd72;x[19] = 16'd59;x[20] = 16'd59;x...
           [21] = 16'd55;x[22] = 16'd55;x[23] = 16'd70;x[24] = 16'd68;x[25] ...
           = 16'd43;x[26] = 16'd88;x[27] = 16'd156;x[28] = 16'd209;x[29] = ...
           16'd234;x[30] = 16'd227;x[31] = 16'd220;#4;
221     //Columna: 23
222     x[0] = 16'd161;x[1] = 16'd158;x[2] = 16'd159;x[3] = 16'd156;x[4] = ...
           16'd146;x[5] = 16'd89;x[6] = 16'd30;x[7] = 16'd47;x[8] = 16'd195;...
           x[9] = 16'd228;x[10] = 16'd191;x[11] = 16'd147;x[12] = 16'd144;x...
```

```
        [13] = 16'd69;x[14] = 16'd78;x[15] = 16'd89;x[16] = 16'd98;x[17] ...
        = 16'd84;x[18] = 16'd84;x[19] = 16'd84;x[20] = 16'd86;x[21] = 16'...
        d99;x[22] = 16'd107;x[23] = 16'd110;x[24] = 16'd117;x[25] = 16'...
        d110;x[26] = 16'd90;x[27] = 16'd106;x[28] = 16'd119;x[29] = 16'...
        d124;x[30] = 16'd163;x[31] = 16'd192;#4;
223     //Columna: 24
224     x[0] = 16'd150;x[1] = 16'd141;x[2] = 16'd163;x[3] = 16'd143;x[4] = ...
        16'd146;x[5] = 16'd151;x[6] = 16'd106;x[7] = 16'd174;x[8] = 16'...
        d239;x[9] = 16'd209;x[10] = 16'd177;x[11] = 16'd129;x[12] = 16'...
        d65;x[13] = 16'd31;x[14] = 16'd33;x[15] = 16'd36;x[16] = 16'd26;x...
        [17] = 16'd41;x[18] = 16'd62;x[19] = 16'd96;x[20] = 16'd140;x[21]...
         = 16'd168;x[22] = 16'd171;x[23] = 16'd169;x[24] = 16'd138;x[25] ...
        = 16'd151;x[26] = 16'd170;x[27] = 16'd170;x[28] = 16'd116;x[29] =...
         16'd64;x[30] = 16'd76;x[31] = 16'd98;#4;
225     //Columna: 25
226     x[0] = 16'd210;x[1] = 16'd164;x[2] = 16'd130;x[3] = 16'd137;x[4] = ...
        16'd138;x[5] = 16'd148;x[6] = 16'd176;x[7] = 16'd228;x[8] = 16'...
        d235;x[9] = 16'd179;x[10] = 16'd112;x[11] = 16'd57;x[12] = 16'd27...
        ;x[13] = 16'd34;x[14] = 16'd59;x[15] = 16'd72;x[16] = 16'd118;x...
        [17] = 16'd139;x[18] = 16'd162;x[19] = 16'd173;x[20] = 16'd168;x...
        [21] = 16'd153;x[22] = 16'd150;x[23] = 16'd161;x[24] = 16'd122;x...
        [25] = 16'd130;x[26] = 16'd136;x[27] = 16'd135;x[28] = 16'd136;x...
        [29] = 16'd61;x[30] = 16'd77;x[31] = 16'd130;#4;
227     //Columna: 26
228     x[0] = 16'd175;x[1] = 16'd221;x[2] = 16'd202;x[3] = 16'd134;x[4] = ...
        16'd151;x[5] = 16'd163;x[6] = 16'd75;x[7] = 16'd63;x[8] = 16'd78;...
        x[9] = 16'd45;x[10] = 16'd23;x[11] = 16'd29;x[12] = 16'd61;x[13] ...
        = 16'd99;x[14] = 16'd140;x[15] = 16'd170;x[16] = 16'd167;x[17] = ...
        16'd165;x[18] = 16'd158;x[19] = 16'd153;x[20] = 16'd150;x[21] = ...
        16'd147;x[22] = 16'd150;x[23] = 16'd158;x[24] = 16'd120;x[25] = ...
        16'd95;x[26] = 16'd124;x[27] = 16'd135;x[28] = 16'd142;x[29] = ...
        16'd177;x[30] = 16'd145;x[31] = 16'd123;#4;
229     //Columna: 27
230     x[0] = 16'd102;x[1] = 16'd132;x[2] = 16'd195;x[3] = 16'd232;x[4] = ...
        16'd147;x[5] = 16'd43;x[6] = 16'd32;x[7] = 16'd38;x[8] = 16'd25;x...
        [9] = 16'd39;x[10] = 16'd74;x[11] = 16'd126;x[12] = 16'd168;x[13]...
         = 16'd176;x[14] = 16'd167;x[15] = 16'd166;x[16] = 16'd155;x[17] ...
        = 16'd154;x[18] = 16'd145;x[19] = 16'd135;x[20] = 16'd133;x[21] =...
         16'd133;x[22] = 16'd127;x[23] = 16'd121;x[24] = 16'd119;x[25] = ...
        16'd100;x[26] = 16'd198;x[27] = 16'd223;x[28] = 16'd216;x[29] = ...
        16'd179;x[30] = 16'd115;x[31] = 16'd96;#4;
231     //Columna: 28
232     x[0] = 16'd133;x[1] = 16'd108;x[2] = 16'd111;x[3] = 16'd134;x[4] = ...
        16'd61;x[5] = 16'd26;x[6] = 16'd46;x[7] = 16'd28;x[8] = 16'd59;x...
        [9] = 16'd120;x[10] = 16'd172;x[11] = 16'd185;x[12] = 16'd177;x...
        [13] = 16'd163;x[14] = 16'd157;x[15] = 16'd162;x[16] = 16'd161;x...
        [17] = 16'd143;x[18] = 16'd133;x[19] = 16'd141;x[20] = 16'd157;x...
        [21] = 16'd171;x[22] = 16'd182;x[23] = 16'd189;x[24] = 16'd195;x...
```

```
        [25] = 16'd221;x[26] = 16'd216;x[27] = 16'd183;x[28] = 16'd159;x...
        [29] = 16'd82;x[30] = 16'd55;x[31] = 16'd83;#4;
233     //Columna: 29
234     x[0] = 16'd135;x[1] = 16'd143;x[2] = 16'd90;x[3] = 16'd27;x[4] = 16'...
        d35;x[5] = 16'd50;x[6] = 16'd37;x[7] = 16'd101;x[8] = 16'd165;x...
        [9] = 16'd173;x[10] = 16'd167;x[11] = 16'd156;x[12] = 16'd160;x...
        [13] = 16'd164;x[14] = 16'd157;x[15] = 16'd149;x[16] = 16'd150;x...
        [17] = 16'd137;x[18] = 16'd151;x[19] = 16'd190;x[20] = 16'd213;x...
        [21] = 16'd216;x[22] = 16'd222;x[23] = 16'd231;x[24] = 16'd236;x...
        [25] = 16'd165;x[26] = 16'd103;x[27] = 16'd91;x[28] = 16'd71;x...
        [29] = 16'd73;x[30] = 16'd92;x[31] = 16'd119;#4;
235     //Columna: 30
236     x[0] = 16'd150;x[1] = 16'd91;x[2] = 16'd35;x[3] = 16'd43;x[4] = 16'...
        d42;x[5] = 16'd58;x[6] = 16'd116;x[7] = 16'd176;x[8] = 16'd173;x...
        [9] = 16'd164;x[10] = 16'd151;x[11] = 16'd149;x[12] = 16'd158;x...
        [13] = 16'd162;x[14] = 16'd154;x[15] = 16'd146;x[16] = 16'd142;x...
        [17] = 16'd164;x[18] = 16'd198;x[19] = 16'd216;x[20] = 16'd210;x...
        [21] = 16'd205;x[22] = 16'd215;x[23] = 16'd225;x[24] = 16'd134;x...
        [25] = 16'd43;x[26] = 16'd88;x[27] = 16'd101;x[28] = 16'd99;x[29]...
         = 16'd103;x[30] = 16'd112;x[31] = 16'd88;#4;
237     //Columna: 31
238     x[0] = 16'd113;x[1] = 16'd33;x[2] = 16'd53;x[3] = 16'd62;x[4] = 16'...
        d59;x[5] = 16'd148;x[6] = 16'd179;x[7] = 16'd170;x[8] = 16'd156;x...
        [9] = 16'd161;x[10] = 16'd163;x[11] = 16'd162;x[12] = 16'd162;x...
        [13] = 16'd163;x[14] = 16'd157;x[15] = 16'd148;x[16] = 16'd140;x...
        [17] = 16'd181;x[18] = 16'd213;x[19] = 16'd210;x[20] = 16'd208;x...
        [21] = 16'd225;x[22] = 16'd221;x[23] = 16'd192;x[24] = 16'd62;x...
        [25] = 16'd89;x[26] = 16'd115;x[27] = 16'd103;x[28] = 16'd102;x...
        [29] = 16'd88;x[30] = 16'd62;x[31] = 16'd83;#200;
239
240
241     $display($time, " << Simulation Complete >>");
242     $finish;
243     // stop the simulation
244   end
```

1.7 Results and Comparison

Two types of test benches were developed to validate the functionality of the design. In the first test bench, only a single column of the input matrix was provided, and the corresponding results are attached. This initial step helped verify the partial functionality of the design. In the second test bench, a complete 32x32 matrix was used as input to thoroughly assess the system's performance and correctness. The results of this full matrix test are also attached for reference (please rotate and zoom the figures for better visualization).

These test benches ensure that the design is functioning as expected with both partial and complete inputs, setting the stage for further validation and optimization in the next stages of development.

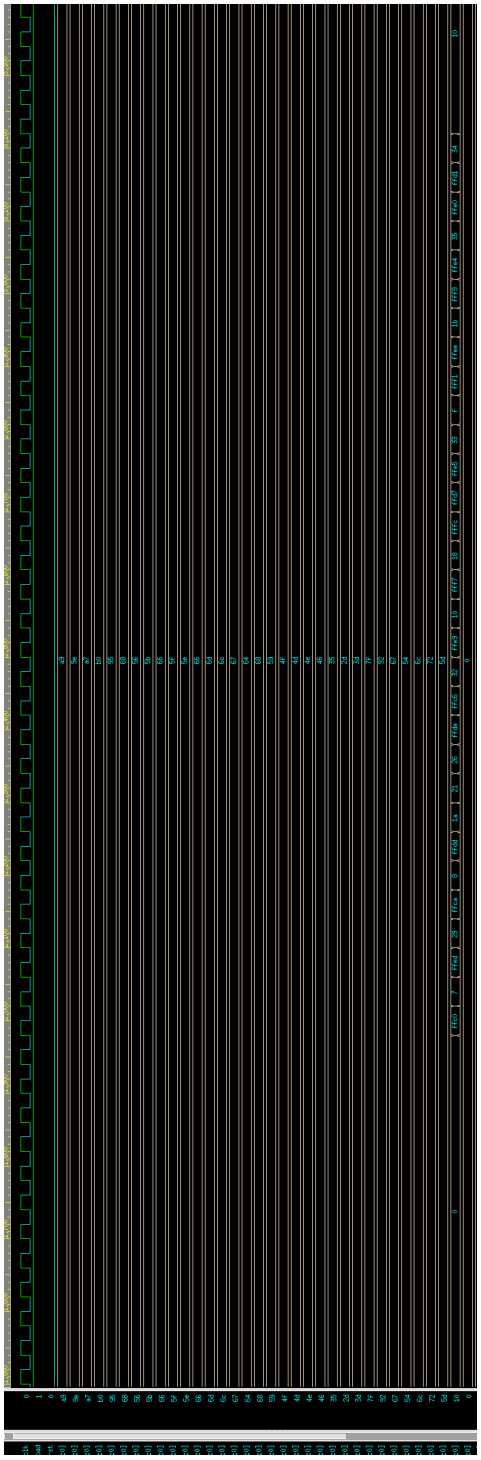


Figure 4: 1D DCT input(ROTATE AND ZOOM FOR VERIFICATION)

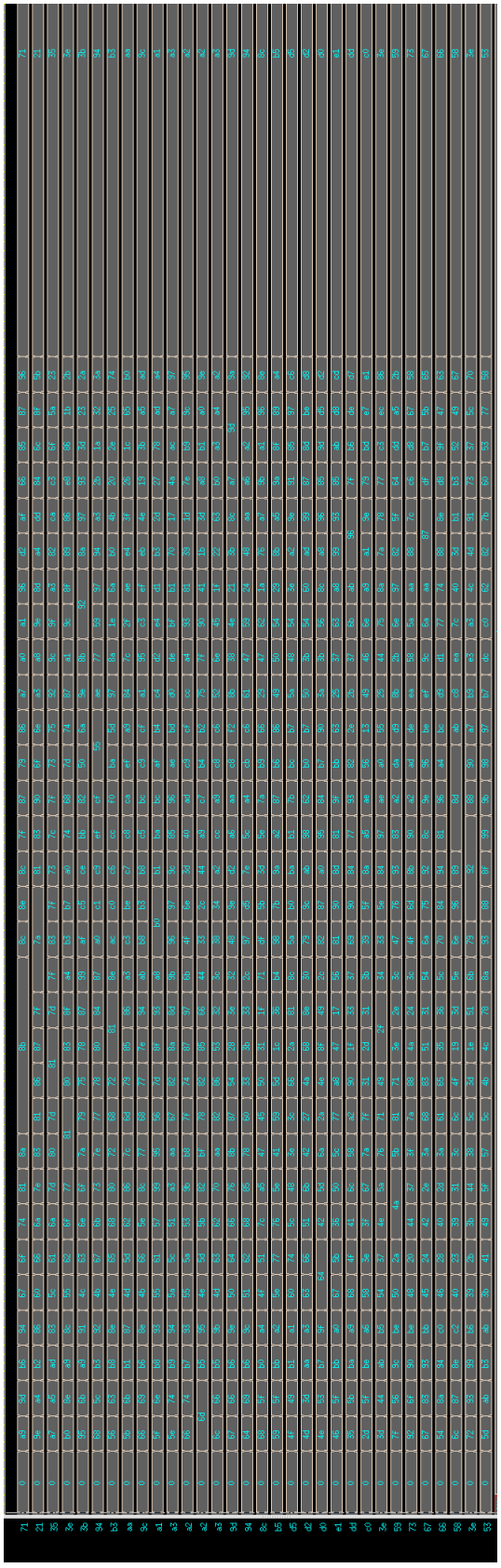
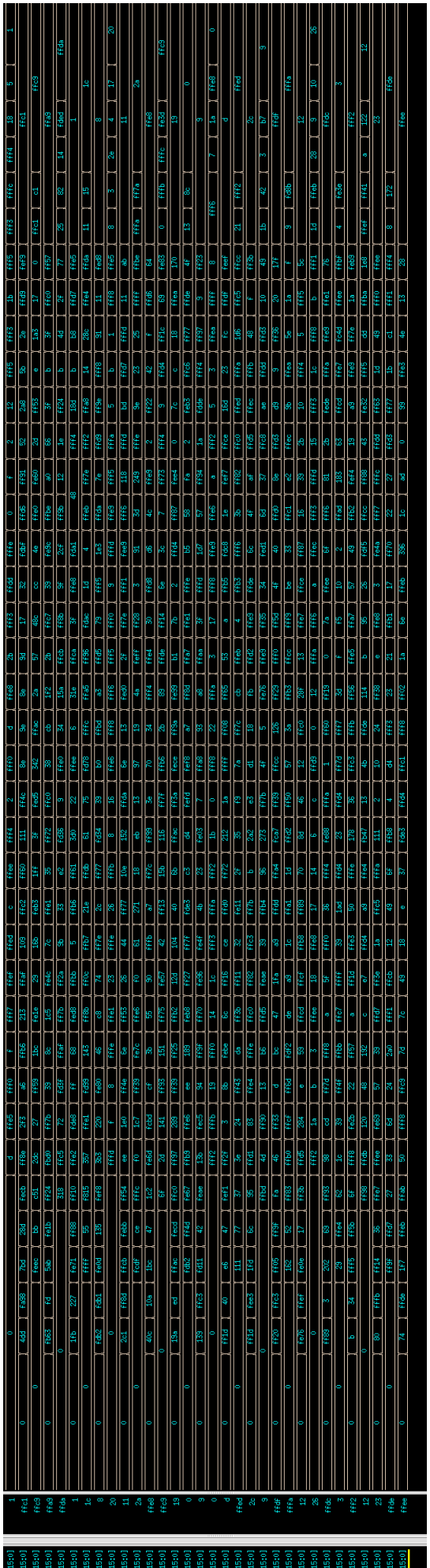


Figure 5: 32*32 input()



Area.rpt

```
1
2  ****************************************
3  Report : area
4  Design : dct
5  Version: U-2022.12-SP5
6  Date   : Sun Oct 27 21:43:26 2024
7  ****************************************
8
9  Information: Updating design information... (UID-85)
10 Warning: Design 'dct' contains 3 high-fanout nets. A fanout number of ...
       1000 will be used for delay calculations involving these nets. (TIM-134)
11 Library(s) Used:
12
13     N16ADFP_StdCelltt0p8v25c_ccs (File: ...
           /home/vlsilab2/TSMCHOME/Executable_Package/Collaterals/IP/stdcell/N16ADFP_S
14
15 Number of ports:                         6244
16 Number of nets:                        100899
17 Number of cells:                        88110
18 Number of combinational cells:          64801
19 Number of sequential cells:             22950
20 Number of macros/black boxes:               0
21 Number of buf/inv:                       4822
22 Number of references:                      15
23
24 Combinational area:              24664.902626
25 Buf/Inv area:                      788.175392
26 Noncombinational area:           26172.047032
27 Macro/Black Box area:                0.000000
28 Net Interconnect area:      undefined  (Wire load has zero net area)
29
30 Total cell area:                 50836.949657
31 Total area:                 undefined
32
33 Information: This design contains black box (unknown) components. (RPT-8)
34 1
35 LOGNAME = chint078
```

2.2 qor.rpt

```
1
2
3  ****************************************
4  Report : qor
5  Design : dct
```

```
Version: U-2022.12-SP5
Date   : Sun Oct 27 21:43:33 2024
****************************************


  Timing Path Group 'comb'
  -----------------------------------
  Levels of Logic:               6.00
  Critical Path Length:          0.68
  Critical Path Slack:           0.44
  Critical Path Clk Period:      1.20
  Total Negative Slack:          0.00
  No. of Violating Paths:        0.00
  Worst Hold Violation:          0.00
  Total Hold Violation:          0.00
  No. of Hold Violations:        0.00
  -----------------------------------

  Timing Path Group 'inputs'
  -----------------------------------
  Levels of Logic:               9.00
  Critical Path Length:          1.14
  Critical Path Slack:           0.02
  Critical Path Clk Period:      1.20
  Total Negative Slack:          0.00
  No. of Violating Paths:        0.00
  Worst Hold Violation:          0.00
  Total Hold Violation:          0.00
  No. of Hold Violations:        0.00
  -----------------------------------

  Timing Path Group 'output'
  -----------------------------------
  Levels of Logic:               0.00
  Critical Path Length:          0.06
  Critical Path Slack:           1.08
  Critical Path Clk Period:      1.20
  Total Negative Slack:          0.00
  No. of Violating Paths:        0.00
  Worst Hold Violation:          0.00
  Total Hold Violation:          0.00
  No. of Hold Violations:        0.00
  -----------------------------------

  Timing Path Group 'reg2reg'
  -----------------------------------
  Levels of Logic:              30.00
  Critical Path Length:          1.17
```

```
Critical Path Slack:           0.00
Critical Path Clk Period:      1.20
Total Negative Slack:          0.00
No. of Violating Paths:        0.00
Worst Hold Violation:          0.00
Total Hold Violation:          0.00
No. of Hold Violations:        0.00
----------------------------------


Cell Count
----------------------------------
Hierarchical Cell Count:          5
Hierarchical Port Count:       4892
Leaf Cell Count:              87751
Buf/Inv Cell Count:            4822
Buf Cell Count:                 230
Inv Cell Count:                4592
CT Buf/Inv Cell Count:            0
Combinational Cell Count:     64804
Sequential Cell Count:        22947
Macro Count:                      0
----------------------------------


Area
----------------------------------
Combinational Area:    24664.902626
Noncombinational Area: 26172.047032
Buf/Inv Area:            788.175392
Total Buffer Area:            72.32
Total Inverter Area:         715.86
Macro/Black Box Area:      0.000000
Net Area:                  0.000000
----------------------------------
Cell Area:             50836.949657
Design Area:           50836.949657


Design Rules
----------------------------------
Total Number of Nets:         96407
Nets With Violations:             1
Max Trans Violations:             1
Max Cap Violations:               1
----------------------------------


```

```
  Hostname: ece-martigny

  Compile CPU Statistics
  ----------------------------------------
  Resource Sharing:                   0.06
  Logic Optimization:                10.73
  Mapping Optimization:             467.81
  ----------------------------------------
  Overall Compile Time:             768.15
  Overall Compile Wall Clock Time:  773.28

  --------------------------------------------------------------------

  Design  WNS: 0.00  TNS: 0.00  Number of Violating Paths: 0


  Design (Hold)  WNS: 0.00  TNS: 0.00  Number of Violating Paths: 0

  --------------------------------------------------------------------
```

2.3 Report .rpt

```
****************************************
Report : design
Design : dct
Version: U-2022.12-SP5
Date   : Mon Oct 21 16:53:43 2024
****************************************

Design allows ideal nets on clock nets.

Library(s) Used:

    N16ADFP_StdCelltt0p8v25c_ccs (File: ...
       /home/vlsilab2/TSMCHOME/Executable_Package
    /Collaterals/IP/stdcell/N16ADFP_StdCell/CCS
    /N16ADFP_StdCelltt0p8v25c_ccs.db)

Local Link Library:

    {N16ADFP_StdCelltt0p8v25c_ccs.db}

Flip-Flop Types:

    No flip-flop types specified.

Latch Types:
```

```
25
26      No latch types specified.
27
28  Operating Conditions:
29
30
31      Operating Condition Name : tt0p8v25c
32      Library : N16ADFP_StdCelltt0p8v25c_ccs
33      Process :   1.00
34      Temperature :  25.00
35      Voltage :   0.80
36      Interconnect Model : balanced_tree
37
38  Wire Loading Model:
39
40      Selected automatically from the total cell area.
41
42  Name           :   ZeroWireload
43  Location       :   N16ADFP_StdCelltt0p8v25c_ccs
44  Resistance     :   1e-05
45  Capacitance    :   1
46  Area           :   0
47  Slope          :   0
48  Fanout   Length   Points Average Cap Std Deviation
49  ------------------------------------------------------------------
50       1      0.00
51       2      0.00
52       3      0.00
53       4      0.00
54       5      0.00
55       6      0.00
56       7      0.00
57       8      0.00
58       9      0.00
59      10      0.00
60      11      0.00
61      12      0.00
62      13      0.00
63      14      0.00
64      15      0.00
65      16      0.00
66      17      0.00
67      18      0.00
68      19      0.00
69      20      0.00
70
71
72
```

```
Wire Loading Model Mode: segmented.

Timing Ranges:

    No timing ranges specified.

Pin Input Delays:

    None specified.

Pin Output Delays:

    None specified.

Disabled Timing Arcs:

    No arcs disabled.

Required Licenses:

    DesignWare

Design Parameters:

    None specified.
1

Design        Wire Load Model            Library
------------------------------------------------
dct                    ZeroWireload      N16ADFP_StdCelltt0p8v25c_ccs
fsm_test_1             ZeroWireload      N16ADFP_StdCelltt0p8v25c_ccs
dct32_WIDTH_X16_WIDTH_Y21_test_1 ZeroWireload N16ADFP_StdCelltt0p8v25c_ccs
transpuesta_WIDTH21_test_1 ZeroWireload  N16ADFP_StdCelltt0p8v25c_ccs
scaling2_WIDTH26       ZeroWireload      N16ADFP_StdCelltt0p8v25c_ccs
shift_add32_WIDTH21_test_1 ZeroWireload  N16ADFP_StdCelltt0p8v25c_ccs


Global Operating Voltage = 0.8
Power-specific unit information :
    Voltage Units = 1V
    Capacitance Units = 1.000000pf
    Time Units = 1ns
    Dynamic Power Units = 1mW    (derived from V,C,T units)
    Leakage Power Units = 1nW


Attributes
----------
```

```
i - Including register clock pin internal power


  Cell Internal Power  =   6.6033  W   (94%)
  Net Switching Power  = 405.7604 mW    (6%)
                         ---------
Total Dynamic Power    =   7.0091  W  (100%)

Cell Leakage Power     = 112.4765 uW

           Intern   Switching     Leakage    Total           Cell
PowerGroup  Power     Power        Power      Power   ( % )  Attrs  Count
-----------------------------------------------------------------------------
io_pad    0.0000     0.0000       0.0000   0.0000  (   0.00%)          0
memory     0.0000    0.0000       0.0000   0.0000  (   0.00%)          0
black_box  0.0000    0.0000       0.0000   0.0000  (   0.00%)          0
clock_network  50.4006 401.0078 1.8097e+04 451.3890(   6.44%)      47551 i
register   6.5471e+03 0.4384 8.2574e+04  6.5476e+03  (  93.42%)    22947
sequential 0.0000   0.0000     0.0000    0.0000  (   0.00%)            0
combinational  5.6555    4.3464  1.1806e+04  10.0138  (   0.14%)  17251
-----------------------------------------------------------------------------
Total 6.6031e+03 mW   405.7926 mW  1.1248e+05 nW  7.0090e+03 mW
1
LOGNAME = chint078
```

Table 1: Comparison of Design Metrics

Metrics	**Unit**	**Reference Design**	**Your Design**
Gate Count	Cell Count	88.6K	88.1k
Clock Frequency	MHz	256	833
Area	μm^2	–	107190

The verified results are attached, and the values have been cross-checked. While there is a slight accuracy mismatch between the general computation of the DCT and this implementation, it arises from the use of shift-adders to approximate multiplication in higher-order DCTs. However, the loss is minimal, and the DCT coefficients computed using butterfly techniques are sufficiently accurate to reconstruct the image without noticeable loss or distortion.

The comparison between the reference design and my design highlights several key performance metrics. As shown in Table 1, the gate count for the reference design stands at 88.6K, while my design achieves a slightly reduced gate count of 88.1K. In terms of clock frequency, my design demonstrates a significant improvement, operating at 833 MHz compared to the reference design's 256 MHz, indicating enhanced processing speed. Additionally, my design has an area footprint of 107,190 μm^2, which reflects the initial layout from the first synthesis pass.

One notable aspect of my design is its current power consumption of 7W. While this figure is relatively high(refernce design has power consumption in milli Watt) it represents a preliminary result from the first synthesis, with no power optimization techniques applied yet. There is substantial scope for further power reduction through subsequent optimizations. Overall, these initial synthesis results show promising performance gains, particularly in speed, with room for further refinement in power and area efficiency.

1.8 Conclusion

In this first phase of the project, the design was successfully synthesized and the initial performance metrics were evaluated. The results show that the gate count of the design is comparable to the reference design, with a minor variation of 88.1K cells. The synthesized design achieved a clock frequency of 833 MHz, demonstrating improved speed over the reference design's 256 MHz. However, the initial area usage of 107190 μm^2 suggests further optimization opportunities. Additionally, the power consumption stands at 7W, indicating the need for power optimization techniques to meet efficiency targets. These synthesis results serve as a baseline for further improvements in subsequent stages. Future work will focus on layout generation, timing closure, power optimization, and thorough validation to ensure the design meets all performance, power, and area constraints. Overall, this report establishes a solid foundation, with the design being well-positioned for the next steps in the RTL-to-GDSII flow.Verified Results are Attached .

1.9 Reference

S. Chatterjee and K. Sarawadekar, "An Optimized Architecture of HEVC Core Transform Using Real-Valued DCT Coefficients," in IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 65, no. 12, pp. 2052-2056, Dec. 2018, doi: 10.1109/TCSII.2018.2815532. keywords: Discrete cosine transforms;Hardware;Computer architecture;Finite wordlength effects;Laplace equations;Circuits and systems;DCT;integer DCT;HEVC;FPGA;ASIC,