|  |
| --- |
| Microelectronic Systems WS18/19 |
|  |
| Lab 2: Designing Hardware IP |
| Report submitted by :  **Bharath Ramachandraiah**  Matrikelnr : 762596  **Bharadwaj Krishnan**  Matrikelnr : 762559 |
|  |
| **Dec/09/2018** |

**Content**

[1 Introduction 3](#_Toc9934889)

[1.1 The objectives of the lab are as follows 3](#_Toc9934890)

[1.2 Data-files for this lab 3](#_Toc9934891)

[1.3 Manuals for this lab 3](#_Toc9934892)

[2 Implementation 4](#_Toc9934893)

[2.1 Verification of DCT-IP 4](#_Toc9934894)

[2.2 Integration of DCT-IP into JPEG encoder project 6](#_Toc9934895)

[3 Comparison and Conclusion 9](#_Toc9934896)

# Introduction

The purpose of this lab is to reuse a third-party DCT-IP, analyze the DCT-IP core and synthesize it into an FPGA using Vivado IP integrator. DCT stands for discrete cosine transform which has its application in signal and image processing, especially for lossy compression of images such as JPEG encoding. In this lab, we will use the hardware implementation of DCT to perform fast JPEG conversion using a DCT-IP provided by Unicore Systems, which is available for free via **opencores.org**. For this purpose, the ZedBoard is used which comes with a Zynq device, a SoC from Xilinx.

## The objectives of the lab are as follows

1. Create an IP on FPGA Programmable logic
2. DCT IP reuse from Unicore Systems
3. Create a block design using the DCT-IP
4. Interface the block design with a C++ project running on ARM Processor
5. Build a Hardware/Software co-design

## Data-files for this lab

*DCT\_IP\_VHDL\_Testbench.zip*

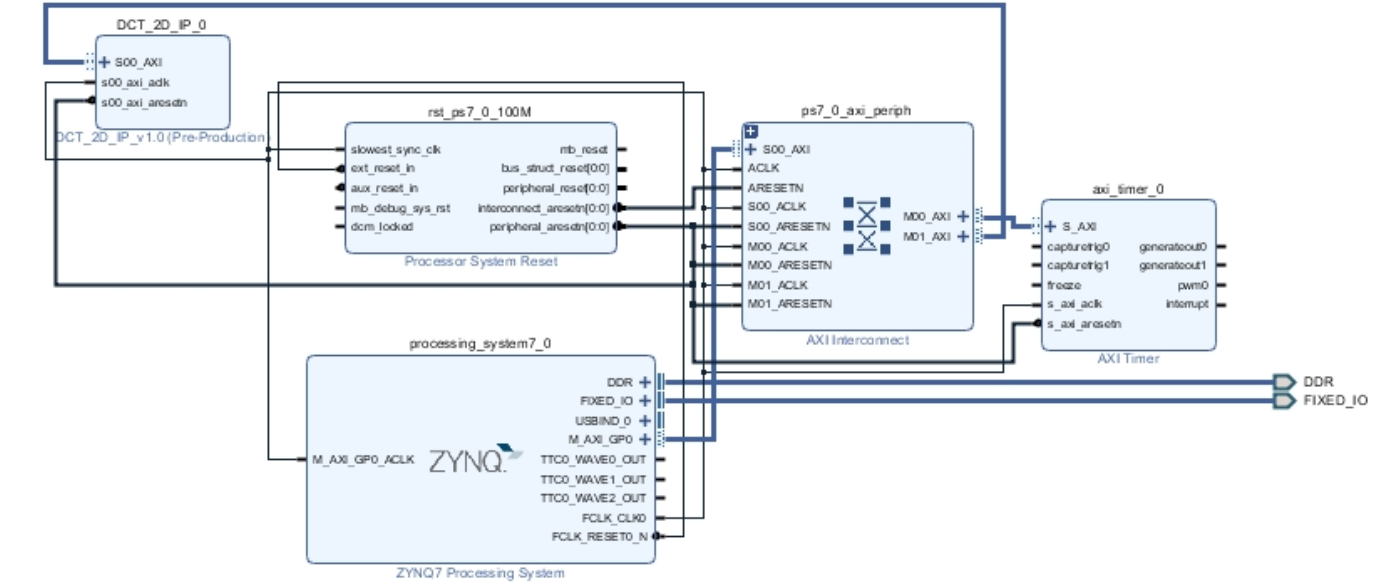
*jpeg\_encoder\_lab\_files.zip*

## Manuals for this lab

*Unicore\_DCT\_IP\_user\_manual.pdf*

*ZedBoard\_HW\_UG\_v2\_2.pdf*

Creating the hardware block design using Processing System (PS) – ZYNQ7 Microprocessor, AXI Interconnect, DCT-IP provided by Unicore Systems to build hardware/software co-design as shown in figure 1.



**Figure 1:**

# Implementation

## Verification of DCT-IP

Task-Description:

Verify the DCT-IP using the VHDL files provided by Unicore Systems

Elaboration:

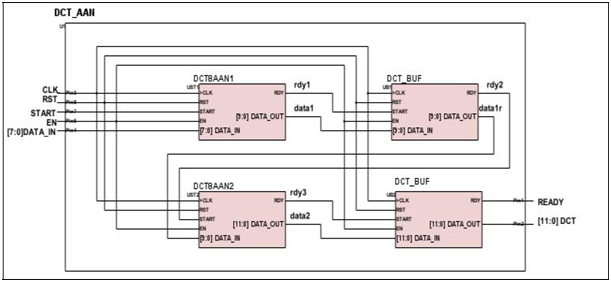
Using the seven VHDL files in the *DCT\_IP\_VHDL\_Testbench.zip,* create a new RTL project and perform the simulation - Behavioural Simulation.

Analysis:

1. ***Analysis of the Custom IP***
2. ***Draw a block diagram of the VHDL module ,,DCT\_AAN”! Briefly explain with the help of IP documentation how to calculate the 2D DCT of a 8x8 matrix.***

***Hint: You may have to analyze its subcomponents***

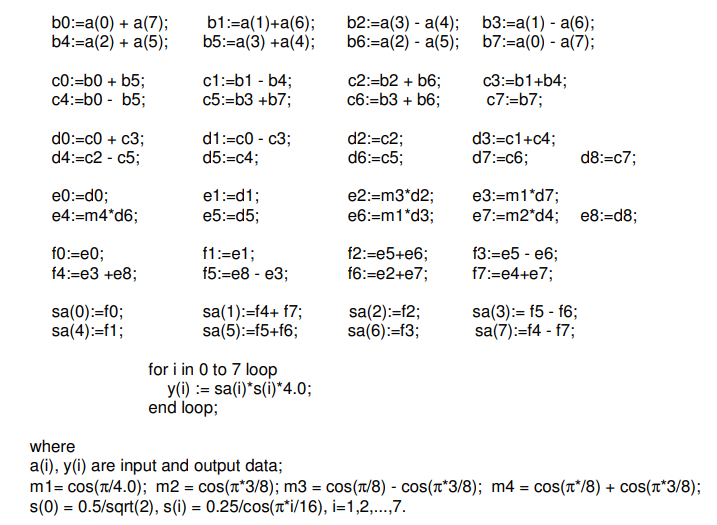
The block diagram of the DCT\_AAN is depicted in the figure 2:



**Figure 2:**

The characteristics of the DCT make it extremely suited for image compression algorithms such as JPEG because DCT removes spatial data redundancies in two-dimensional data. The 2D DCT transforms a 8x8 block data array into a 8x8 block result array. First the 8-point DCT transforms the columns, then it transforms the rows. The steps to perform the calculation is as follows:

1. Subtract the mean value 128 from input data to minimize the redundancy in the input data block. If the core can compute input data in range -128 to 127, then the mean value subtraction can be skipped.
2. The DCT is calculated using Arai, Agui, and Nakajama 8-point DCT algorithm to minimize the amount of calculations. The algorithm consists in the following calculations



This algorithm has only 13 multiplies. Its 8 rest multiplies can be discarded when the outputs are considered to be scaled ones. In this situation the proper scaling is usually performed in the stage of DCT coefficient dividing to the scale factors before the Huffman encoding.

1. ***A test bench for the DCT-IP core is also available. Explain the functions of the three components „TEST\_DCT“, „DCT\_BEH“ and „BMP\_Generator“! What function does the process „ERROR\_CALC“ have?***

The functions of the three components „TEST\_DCT“, „DCT\_BEH“ and „BMP\_Generator“ are:

**TEST\_DCT**

TEST\_DCT is the test bench of DCT IP wherein we instantiate the BMP\_Generator, DCT\_AAN, DCT\_BEH modules to check the DCT behaviour. The data pattern is generated using the module BMP\_Generator. Discrete Cosine Transform is calculated in two methods in this module. The first method is calculated using the DCT\_AAN module and the other method is calculated using DCT\_BEH module.

**DCT\_BEH**

It is the behavioral model of the DCT processor. It computes 2D DCT using the floating point calculations. Its results are rounded to 12 bits. Therefore, it serves as the standard model.

**BMP\_Generator**

BMP\_Generator generates the dataflow of testing arrays. In one mode it generates predefined arrays, in another mode does randomized ones.

***What function does the process „ERROR\_CALC“ have?***

The ERROR\_CALC calculates the differences in the output data of DCT\_AAN and DCT\_BEH. The resulting signals are ERROR which is result of signal subtraction.

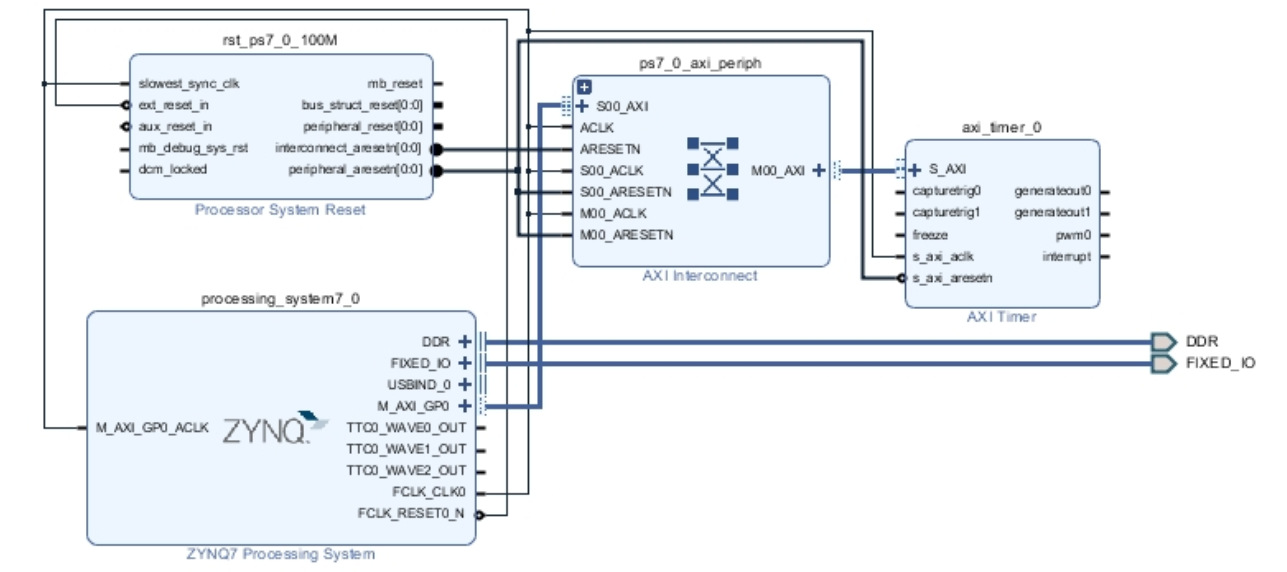
## Integration of DCT-IP into JPEG encoder project

Task-Description:

Integrate the hardware custom IP verified earlier into an existing JPEG encoder application project.

Elaboration:

1. Create a Block Design by adding IPs - **Zynq Processing System** and **AXI Timer.**
2. Create a new AXI4 peripheral to allow a connection between the ARM and FPGA via the **AXI4Lite** Bus with 50 registers each having 32 bits as shown in figure 3.



**Figure 3:**

1. Create a new DCT-IP using the VHDL files provided in zip package to the IP peripheral.
2. Replace the content of DCT\_2D\_IP\_v1\_0\_S00\_AXI to the new interfaces to make it compatible with the jpeg encoder application.
3. Instantiate the DCT\_AAN core with the following configuration for DCT\_AAN

The Code snippet:

U2: DCT\_AAN

generic map( d\_SIGNED => 0, scale\_out => 1)

port map(

CLK => S\_AXI\_ACLK,

RST => not(S\_AXI\_ARESETN),

START => start\_pulse2,

EN => slv\_reg49(1),

DATA\_IN => dct\_data,

DATA\_OUT => dct\_core\_out,

RDY => dct\_core\_rdy

);

-- User logic ends

1. Add the created DCT-IP to the above block design, create HDL Wrapper and Generate the bitstream for the FPGA.
2. Export the hardware to the SDK, and launch the SDK; define hardware platform and the board support package. Get the Jpeg encoder C++ application to the SDK by creating a new project.
3. Flash the bitstream to the FPGA and the JPEG encoder application
4. Now the application should read the BMP images containing in the SD Card and converts them to compressed JPEG images. The conversion process and results are confirmed by the UART log messages.

Analysis:

1. ***Verification of the Custom IP***
2. ***Explain the meaning of the following output signals: DCT[11:0], DCT\_STD[11:0], ERROR, QUADMEAN!***

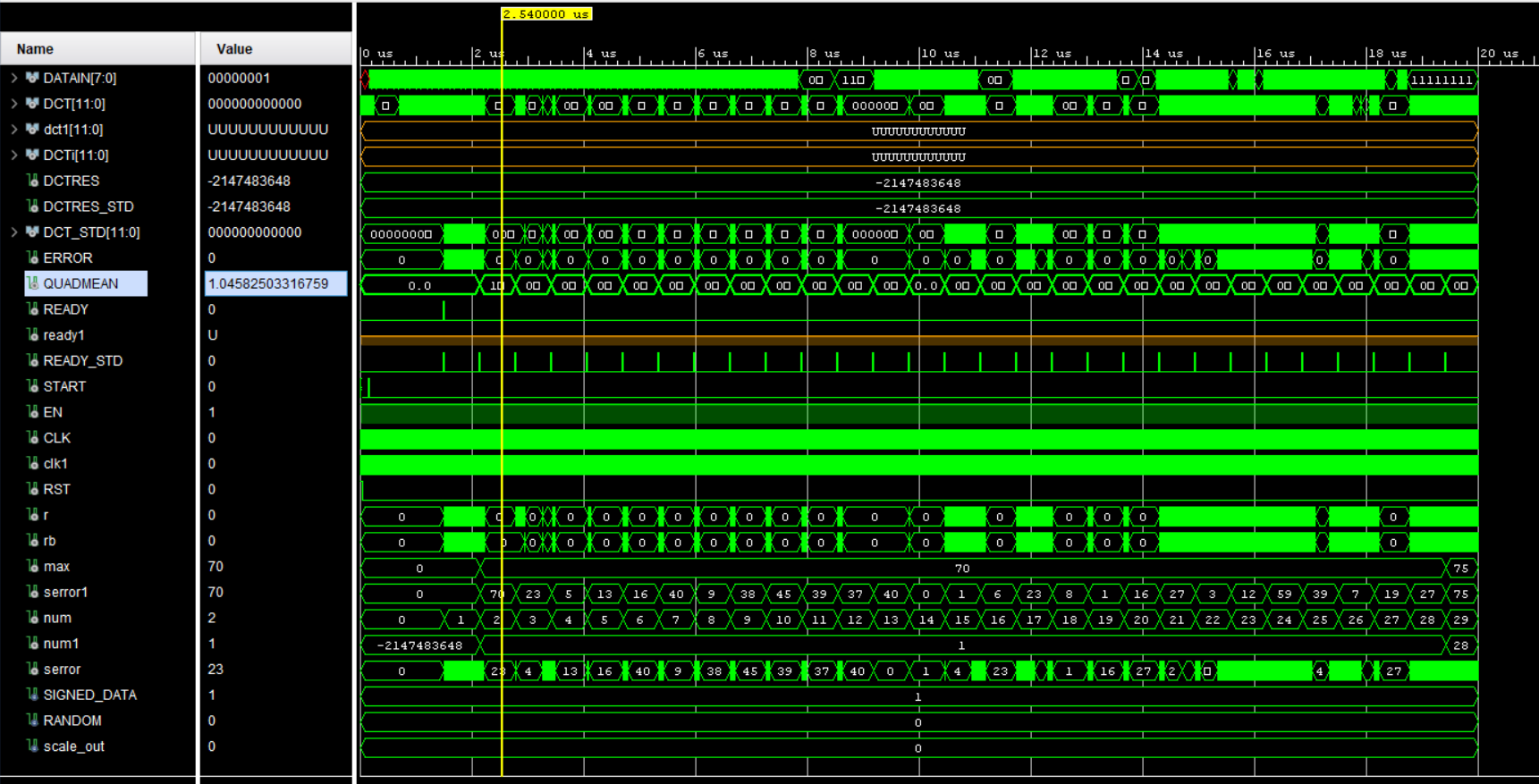
The **DCT[11:0]** - is the DCT output signal from DCT\_AAN (DCT instance), which is the Discrete Cosine Transform signal from the input signal.

The **DCT\_STD[11:0]** - is the DCT output signal from DCT\_BEH (Behavioral model of the DCT processor), which is the Discrete Cosine Transform signal computed by 2D DCT using the floating point calculations from the input signal.

**ERROR** are the differences in the output data of DCT\_AAN and DCT\_BEH.

**QUADMEAN** is the mean square error value for the current data array.

1. ***Verify the functionality of the custom IP using the provided test bench. Simulate at least for 20µs (step 4).***

**

**Figure 4:**

1. ***What is the delay (number of clock cycles) between first data input and first data output?***

132 Cycles

1. ***How many clock cycles does it take to read in one input matrix?***

64 Cycles to read 8x8 input matrix

1. ***Give the min. and max. value of mean square error for this simulation! Attach simulation snapshots to verify your findings!***

Minimum Value : 0

Maximum Value : 1.04582503316759

1. ***VHDL-code for instantiation of the DCT core!***

U2: DCT\_AAN

generic map( d\_SIGNED => 0, scale\_out => 1)

port map(

CLK => S\_AXI\_ACLK,

RST => not(S\_AXI\_ARESETN),

START => start\_pulse2,

EN => slv\_reg49(1),

DATA\_IN => dct\_data,

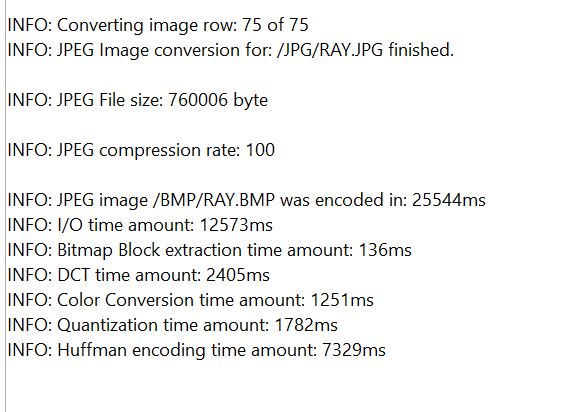
DATA\_OUT => dct\_core\_out,

RDY => dct\_core\_rdy

);

-- User logic ends

1. ***Screenshot of statistics in Terminal 1 (step14)! Compare BMP vs JPEG regarding size and quality!***



# Comparison and Conclusion

***Comparison 1:***

Original Image:

Name: BLU.BMP; Size: 84.4KB

******

Compressed Image:

Name: BLU.JPG; Size: 3.8KB

**

***Conclusion:***

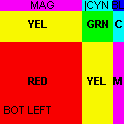
size reduction: 95.49%

Quality : Same quality is maintained

***Comparison 2:***

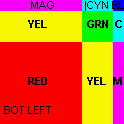
Original Image:

Name: FLAG\_B24.BMP; Size: 45KB

******

Compressed Image:

Name: FLAG\_B24.JPG; Size: 18KB

**

***Conclusion:***

size reduction: 60%

Quality : Same quality is maintained

***Comparison 3:***

Original Image:

Name: RAY.BMP; Size: 1.37MB

******

Compressed Image:

Name: RAY.JPG; Size: 741KB

**

***Conclusion:***

size reduction: 47.18%

Quality : The quality is a bit deteriorate, which is evident near the Alphabets D and e. But still it is under acceptable range.