Indiana University Purdue University Indianapolis
Multimedia Applications
ECE 53700 (Fall 2016)

Final Project Report On

Optimization of a DCT_hier Implementation using Catapult C

Submitted By

MD SAIFUL ISLAM
UID: 2000101450

Submitted To

Dr. Mohamed El-Sharkawy
Professor of Electrical and Computer Engineering, IUPUI

Date: 21st November, 2016

Goal

The purpose of this project is to gain experience optimizing a given implementation using the Catapult C analysis tools.

The C++ implementation of a DCT_hier algorithm provided by Mentor Graphics is used to design a chip architecture meeting the following design specifications:

Altera StratixII EP2S60F484C with speed grade 4
(Other technologies are acceptable)
Clock speed = 100MHz
Latency <=6us
Total Area Score <=3000
I/O mapped to Single port RAMs

Implementation

It is mentioned that in order to achieve the above requirements, the source code and the specifications should not be modified. This can be achieved through unrolling, memory mapping, pipelining, and increasing word width that was studied in previous labs.

DCT heir Algorithm

The DCT is often used for JPEG image compression. Typically, 8x8 blocks of pixels are processed with the DCT in order to reduce the number of bits to represent the same 8x8 block. The source code below is a DCT_hier algorithm. It differs from the basic DCT algorithm only for fact that the main loop calls two functions "matmult1" and "matmult2" which is "mult1" loop and "mult2" loop respectively. This algorithm multiplies one 8x8 input block by one 8x8 orthogonal block to produce the DCT. This is done in both the horizontal and vertical dimensions. The "mult1" loop multiplies the input by the coefficients and stores the results in a 21 bit "temp" array. The "mult 2" loop multiplies the "temp" array by the coefficients to yield the 31 bit DCT value. That value is then shifted right by 20 bits.

DCT heir Source Code:

```
#include "global_dct.h"
const ac_int<10> coeff[XYSIZE][XYSIZE] = {

      362,    362,    362,    362,    362,    362,    362,    362,
      502,    425,    284,     99,    -99,   -284,   -425,   -502,
      473,    195,   -195,   -473,   -473,   -195,    195,    473,
      425,    -99,   -502,   -284,    284,    502,     99,   -425,
      362,   -362,   -362,    362,    362,   -362,   -362,    362,
      284,   -502,     99,    425,   -425,    -99,    502,   -284,
      195,   -473,    473,   -195,   -195,    473,   -473,    195,
       99,   -284,    425,   -502,    502,   -425,    284,    -99
   };

#pragma hls_design
void matmult1 ( ac_int<9> input[][XYSIZE], ac_int<21> temp[][XYSIZE] )
{
     ac_int<21> tmp; ac_int<31>
     dct_value;

     mult1:for (int i=0; i < XYSIZE; ++i ) middle1:for (int j=0; j
        < XYSIZE; ++j ) {
           tmp = 0;
           inner1:for (int k=0; k < XYSIZE; ++k )
                 tmp = tmp + input[i][k] * coeff[j][k]; temp[j][i] =
           tmp;
        }
}

#pragma hls_design
void matmult2 ( ac_int<21> temp[][XYSIZE], ac_int<11> dct[][XYSIZE] ) {  ac_int<21>
   tmp;
   ac_int<31> dct_value;

   mult2:for (int ii=0 ; ii < XYSIZE; ++ii ) middle2:for (int
       j=0; j < XYSIZE; ++j ) {
          dct_value = 0;
          inner2:for (int k=0 ; k < XYSIZE ; ++k )
                dct_value = dct_value + coeff[ii][k] * temp[j][k]; dct[ii][j] = dct_value >>
             20;
        }
}

#pragma design top
void hier_dct_proc(ac_int<9> input[XYSIZE][XYSIZE], ac_int<11>
dct[XYSIZE][XYSIZE]) {

   ac_int<21>temp[XYSIZE][XYSIZE];

   matmult1( input, temp);
   matmult2(temp, dct);
}
```

Procedure

Part 1:

Add Input Files hier_dct.cpp and global_dct.h files to your project. Set up your design, then generate RTL and Gantt chart.
Record your results in Tables 1.1., 1.2 and 1.3.

Table 1.1: Results of Part 1

Total Area Score	766.53
Latency Cycles	2182
Latency Time (ns)	21820

Does this meet the desired specifications? No

Table 1.2: Analysis for mult1/inner1 of Part 1.

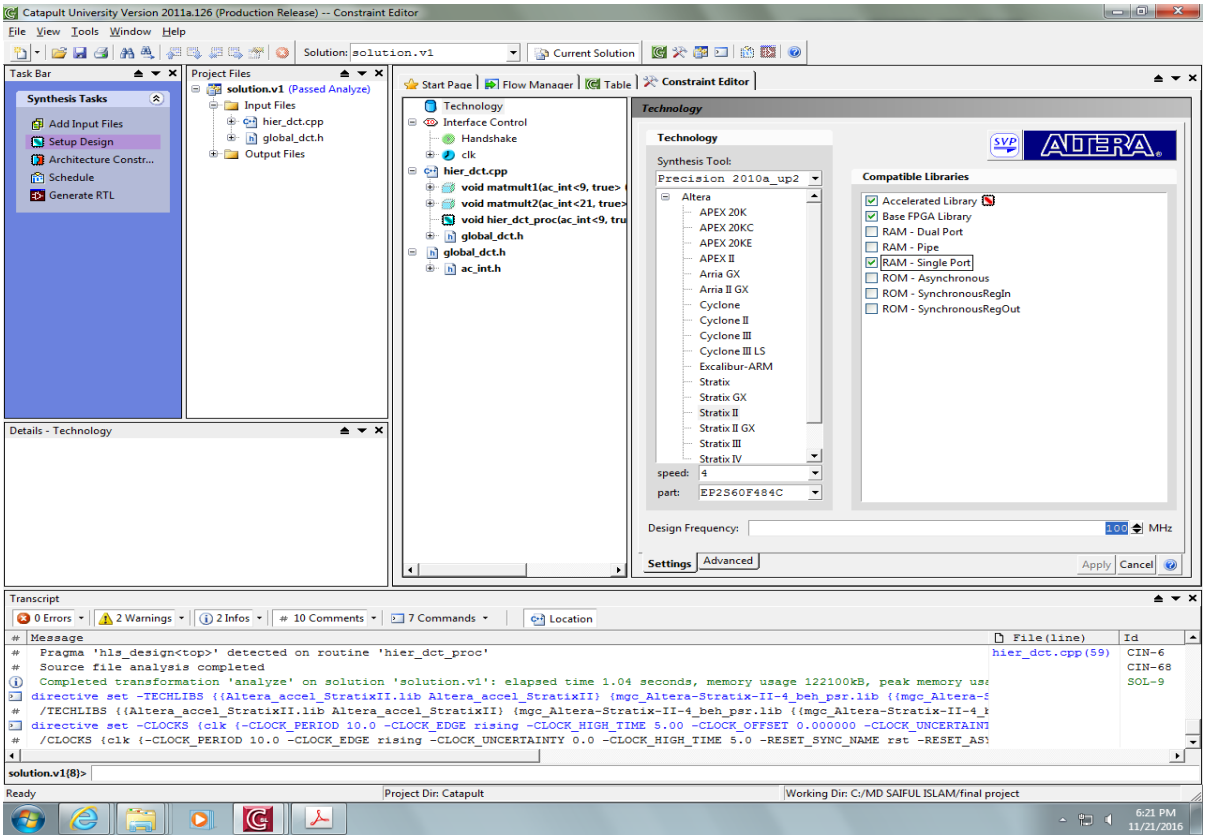
How many c-steps (cycles) are required to execute one iteration of the inner1_for loop?	2
How many multipliers are needed for one loop iteration of the inner1_for loop?	1
How many times does the inner1_for loop iterate?	8
What is the total latency (cycles) required to fully execute the inner1_for loop?	1024
What is the total latency (cycles) required to fully execute the mult1 loop?	1096

Table 1.3: Analysis for mult2/inner2 of Part 1.

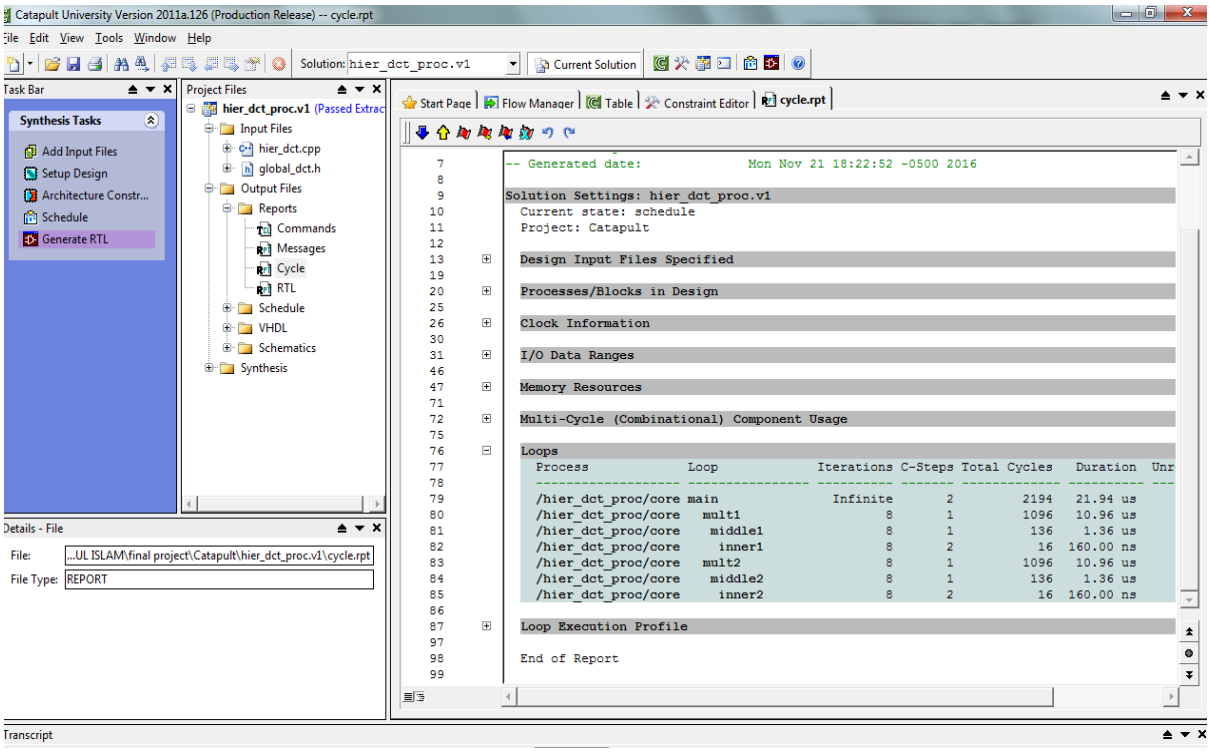
How many c-steps (cycles) are required to execute one iteration of the inner2_for loop?	2
How many multipliers are needed for one loop iteration of the inner2_for loop?	1
How many times does the inner2_for loop iterate?	8
What is the total latency (cycles) required to fully execute the inner2_for loop?	1024
What is the total latency (cycles) required to fully execute the mult2 loop?	1096

The following steps need to be performed to achieve our goal:

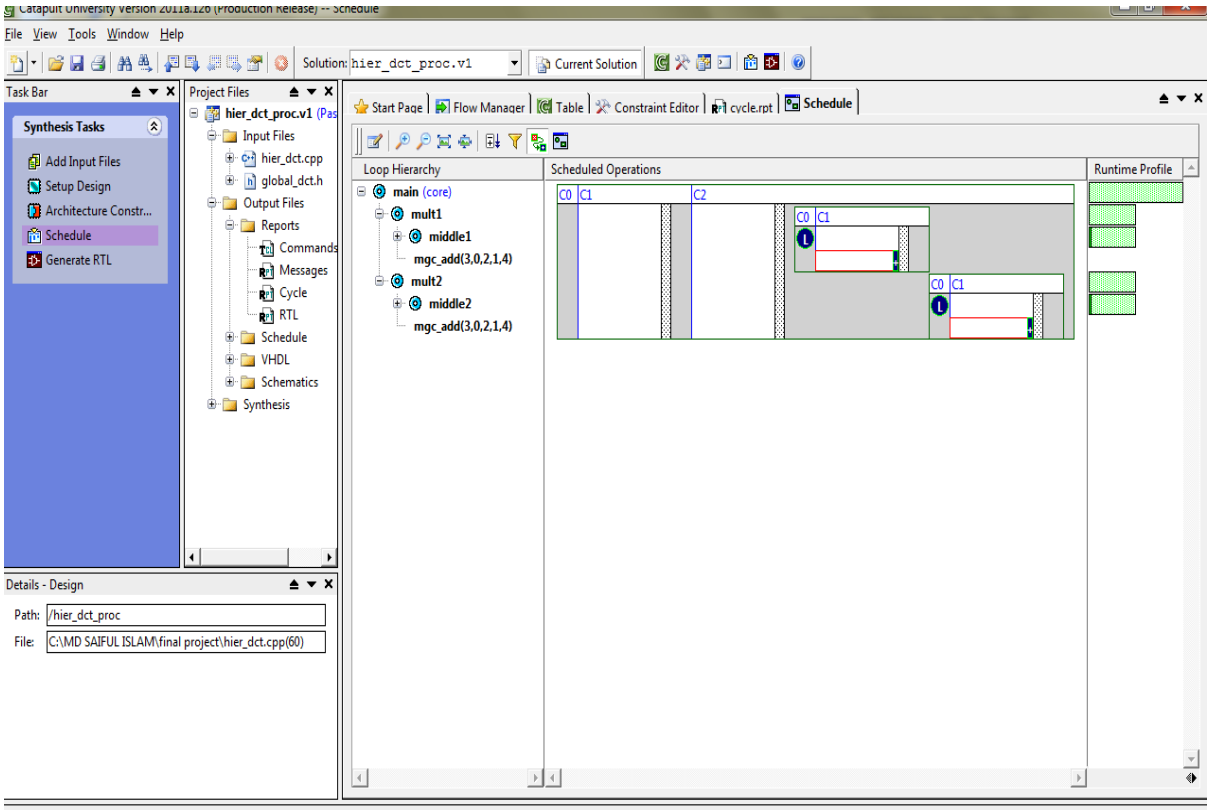
1. Setting up design.



2. Cycle Report:



3. Gantt chart:



Part 2:

Unroll the inner1_for and inner2_for loops. Generate RTL and Gantt chart for such option.
Record the results in Tables 2.1, 2.2 and 2.3.
Table 2.1: Results of Part 2.

Total Area Score	1922.18
Latency Cycles	1413
Latency Time (ns)	14130

Does this meet the desired specifications? No

Table 2.2: Analysis of Part 2.

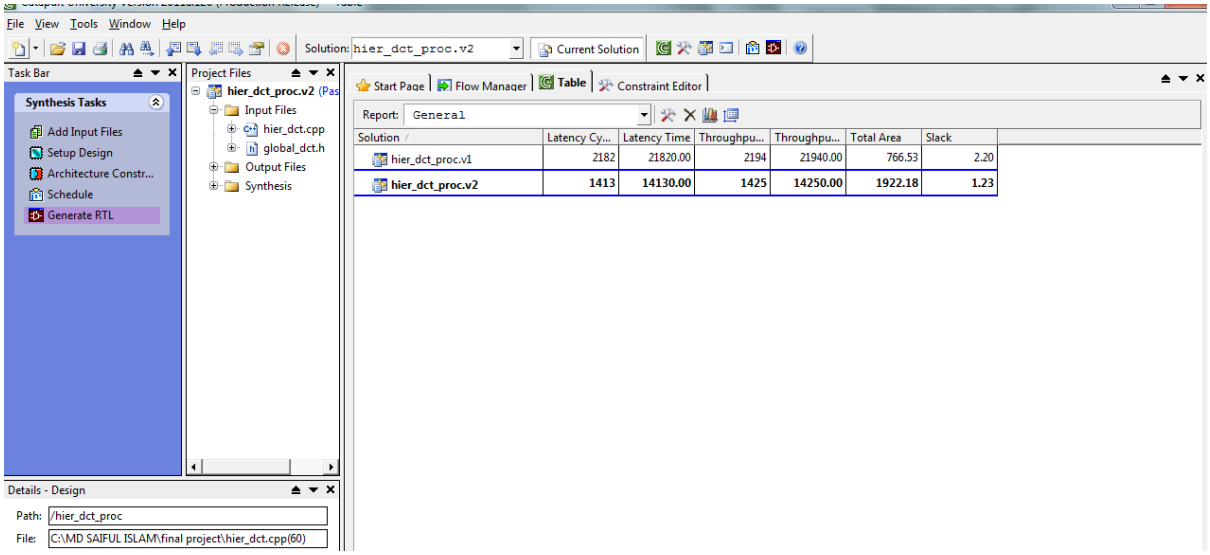
How many cycles are required to execute the unrolled inner2_for loop?	Inner2 not exit due to unrolling
How many cycles were required to execute the rolled inner2_for loop?	Inner2 not exit due to unrolling
How many cycles are required to execute the mult2 loop with inner2_for unrolled?	704
How many cycles were required to execute the mult2 loop with inner2_for rolled?	1024

Table 2.3: Analysis using Component Utilization view

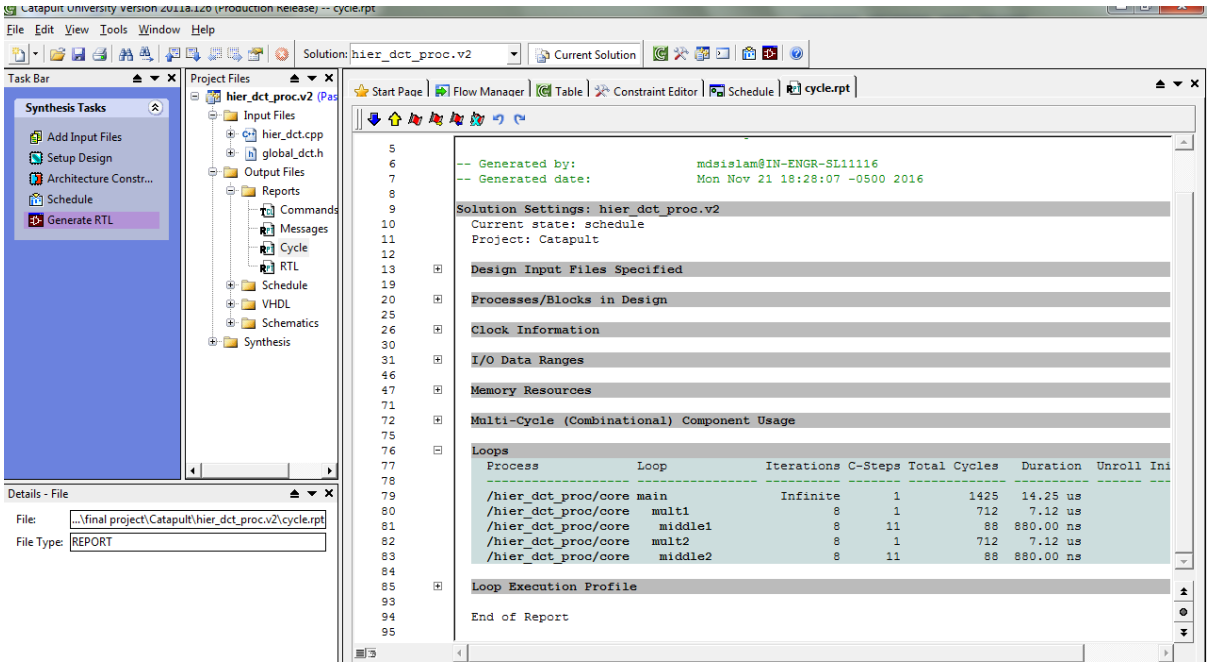
How many multipliers are required to execute the unrolled inner2_for loop in Part	8
How many multipliers are required to execute the rolled inner2_for loop in Part 1?	1

The following steps need to be performed to achieve our goal:

1. Table



2. Cycle Report



Part 3:

To optimize inner2 loop further, map the Temp array to a dual port memory in order to read up to two samples per cycle. Remember that the IO specification is Singleport RAMs. Generate RTL and Gantt chart for such option.

Record the results in Table 3.1 and Table 3.2.

Table 3.1: Results of Part 3

Total Area Score	1948.74
Latency Cycles	1413
Latency Time (ns)	14130

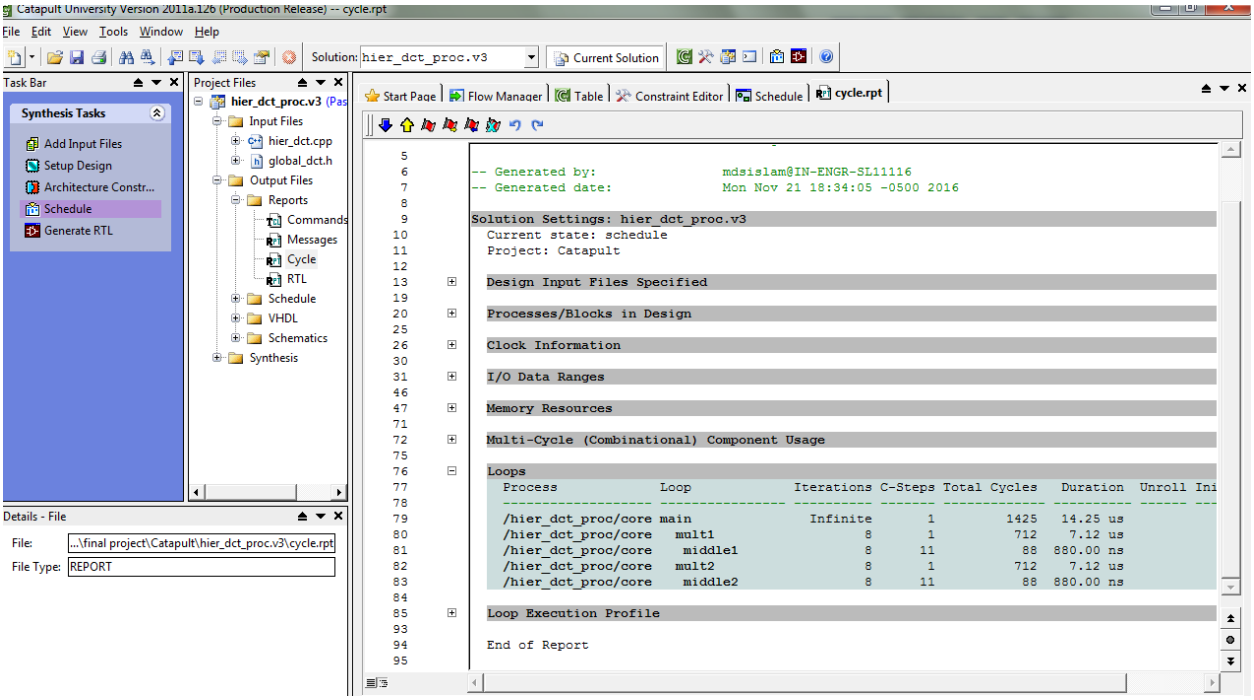
Does this meet the desired specifications? No

Table 3.2: Analysis of Part 3

How many cycles are required to execute an iteration of middle2 in Part 3?	11
How does this compare with Part 2?	11
How many multipliers are required to implement middle2 in Part 3?	8
How does this compare with Part 2?	4
How many C-steps require accesses to the dual port RAM in Part 3?	4
How does this compare with Part 2 considering single port RAM?	8

The following steps need to be performed to achieve our goal:

1. **Cycle report:**



Part 4:

Pipelining is a way to improve the design performance by forcing to start a loop iteration before the previous iteration finishes. Generate RTL and Gantt chart for such option.

Record the results in Table 4.1. Table

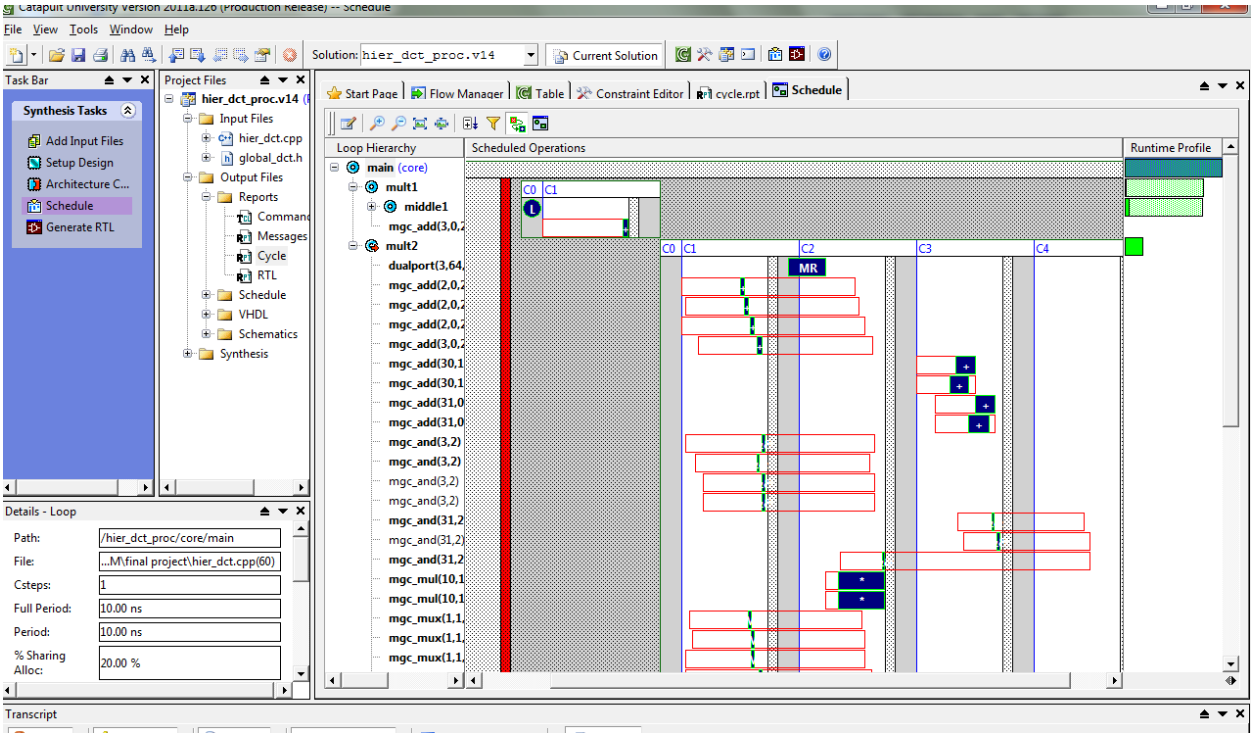
4.1: Results for Part 4

Total Area Score	2464.93
Latency Cycles	1352
Latency Time (ns)	13520

Does this meet the desired specifications? No

The following steps need to be performed to achieve our goal:

1. **Gantt Chart:**



Part 5:

By default, the generated memory will have a word width equivalent to the bit width of the stored data type. In this case (ac_int<9>), the memory is 9 bit wide, and can store up to 64 (8x8) elements. Increase the word width to 18 bits and record the results in Table 5.1.

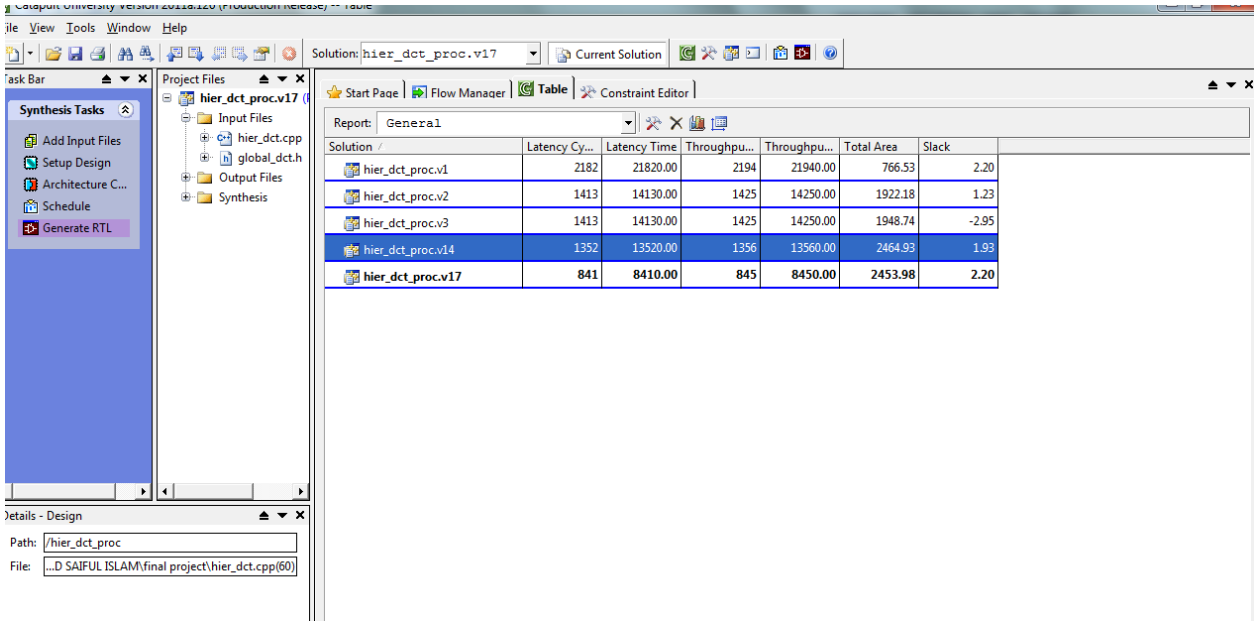
Table 5.1: Results for Part 5

Total Area Score	2453.98
Latency Cycles	841
Latency Time (ns)	8410

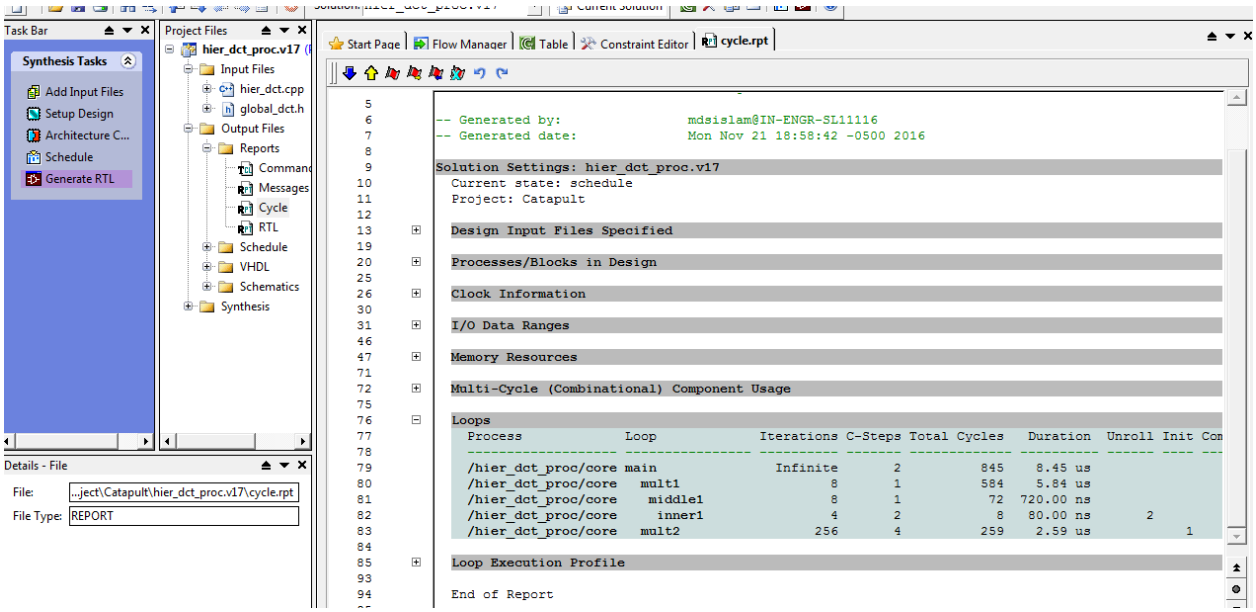
Does this meet the desired specifications? No, but it is very close to it

The following steps need to be performed to achieve our goal:

1. **Table:**



2. Cycle Report:



If not, recommend other options to meet your specifications.

To get desired output we make the following changes:

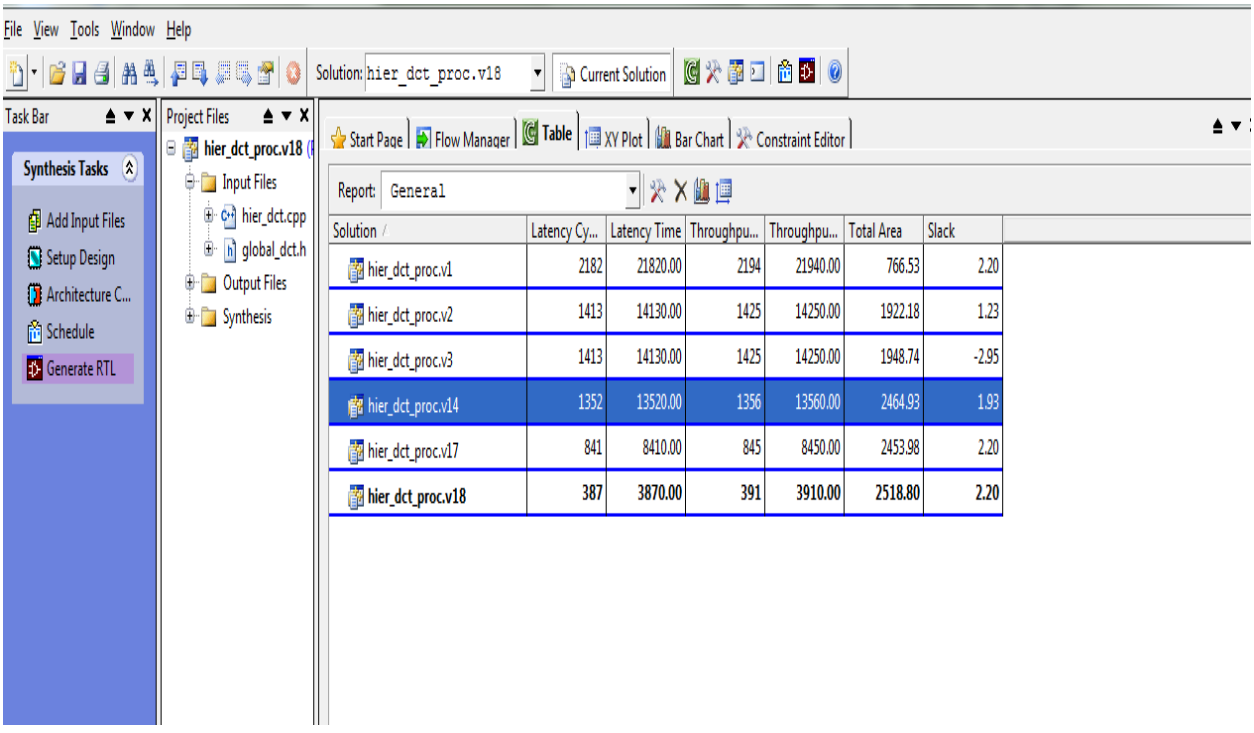
Input word width=18
dct word width=22
temp word width=42
Pipelined both mult1 and mult2

Desired Output table:

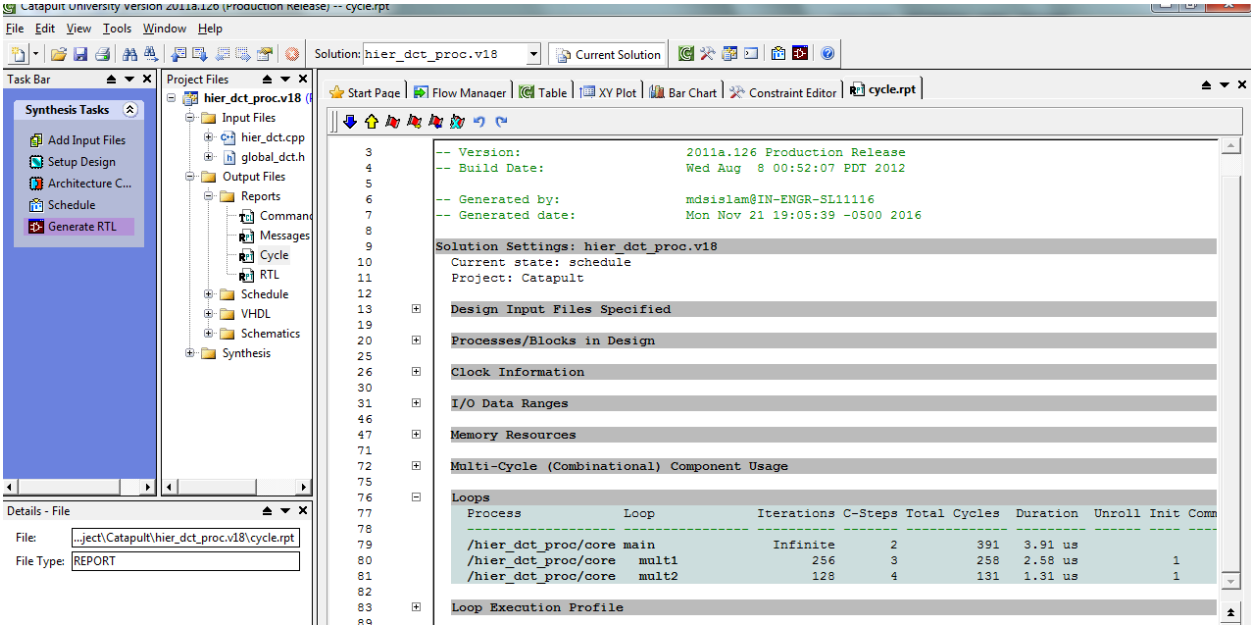
Total Area Score	2518.80 (less than 3000)
Latency Cycles	387
Latency Time (ns)	3870 (less than 6 micro second)

The following steps need to be performed to achieve our goal:

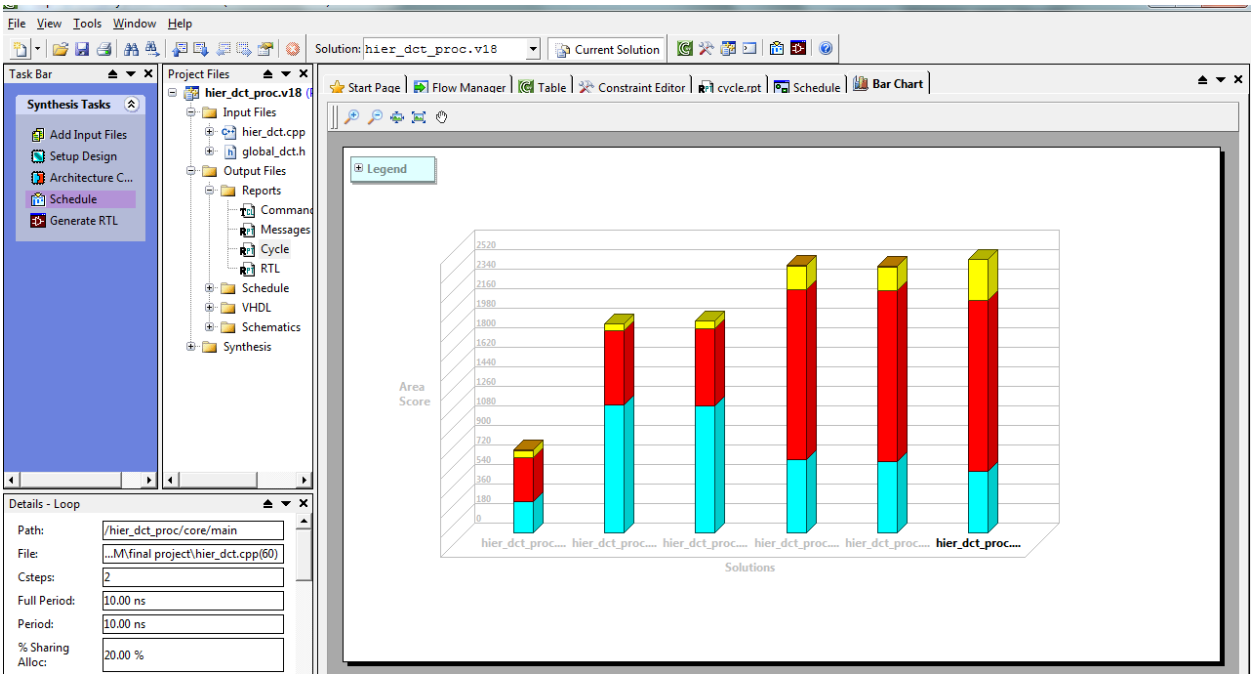
1. **Table**



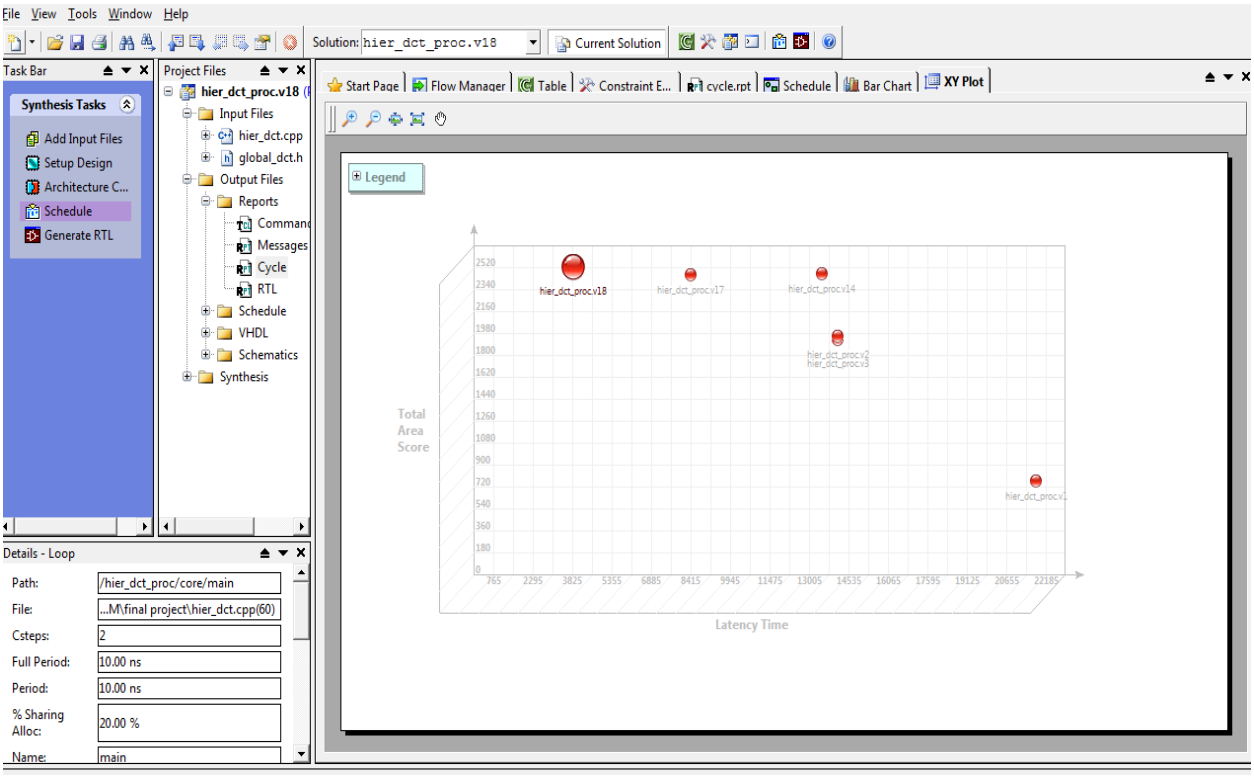
2. Cycle Report



3. Bar chart:



4. X-Y Plot:



Results

The latency time that I obtained was 3.87 µs (less than 6 micro second) and the total area was 2518.80 (less than 3000).