ICS 155B Spring 1998
Lab 1
Behavioral Modeling for Discrete Cosine Transformation
Due Date: April 20 (Monday) by noon

Information and Computer Science
University of California, Irvine

April 13, 1998

In this lab, you are provided with the VHDL behavioral model of the DCT chip you're going to model for the rest of this quarter. A simple test bench with one test pattern is also provided. You are required to debug the model and understand the behavioral modeling. You are also required to write another test pattern. This test pattern will be used in the other two labs.

1 Problem Specification

Discrete Cosine Transform (DCT) is often a critical part in both still and motion picture compression. The definition of DCT for a $N \times N$ image is as follows:

$$F[u,v] = \frac{1}{N^2} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} f[m,n] cos \frac{(2m+1)u\pi}{2N} \; cos \frac{(2n+1)v\pi}{2N}$$

where:

u, v = discrete frequency variables ($0 \leq u,v \leq N-1$).
f[m, n] = gray level of a pixel at position (m, n) of $N \times N$ image ($0 \leq m,n \leq N-1$).
F[u, v] = Coefficient of point (u, v) in spatial frequency domain.

As shown in the code below, DCT is basically two consecutive matrix multiplications. In our design we will assume that N = 8. In this case, C will be a 8×8 matrix defined as

$$C[u][n] = \frac{1}{8} cos \frac{(2n+1)u\pi}{16}$$

Furthermore, let COS1 be integer matrix defined as

$$COS1 = round(factor * C)$$

and COS2 be its transpose:

$$COS2 = COS1^T$$

Let f be the 8×8 input block of image, and F be the output matrix, then

$$F = COS1 \times f \times COS2$$

```
int   COS1[8][8], COS2[8][8];

void  MatrixMult (int a[][8], int b[][8[], int c[][8]) {

    register int   i, j, k;

    for (i=0; i<8; i++)
       for (j=0; j<8; j++) {
          c[i][j] = 0;
          for (k=0; k<8; k++)
             c[i][j] += a[i][k] * b[k][j];
       }
}

void  DCT (int f[][8], int F[][8]) {

   int  Temp[8][8];

   MatrixMult(f, COS2, Temp);
   MatrixMult(COS1, Temp, F);
}
```

2 Interface Specification

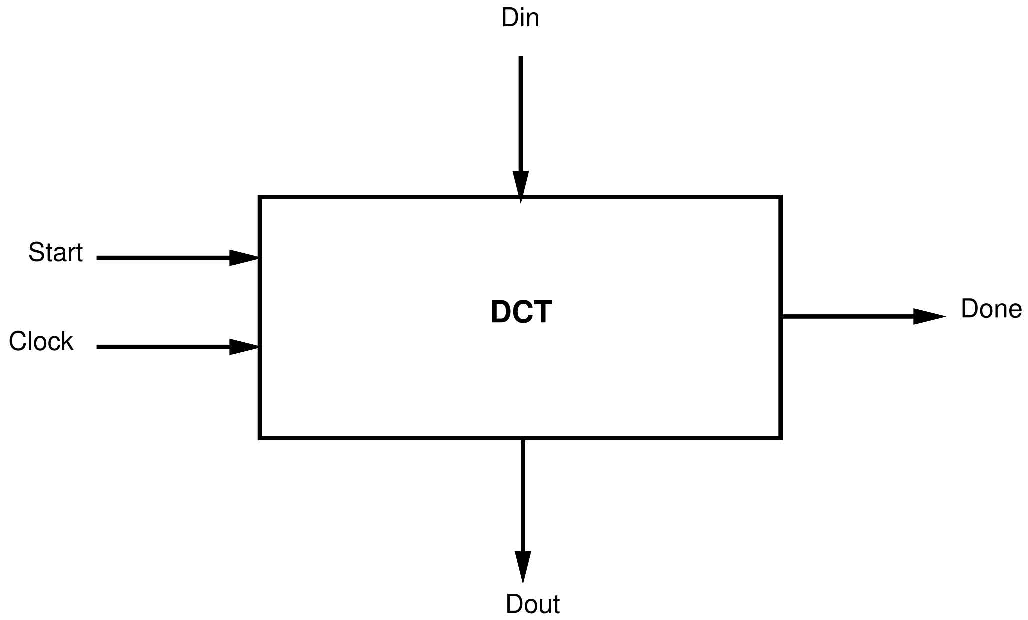

Figure 1: Block Diagram of DCT chip

As shown in Figure 1, the interface of the DCT chip consists of a pair of hand-shake ports “Start”, which indicate the start of the computation, and “Done”, which indicate the end of the computation. The chip also has a input data port and a output data port.

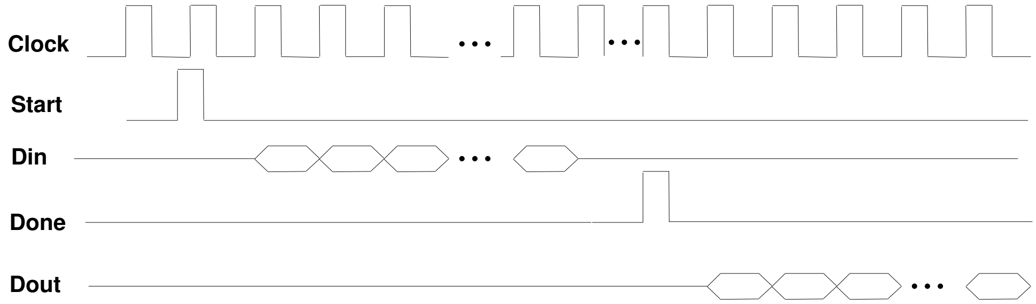

Figure 2: Timing Spec of DCT chip

Figure 2 shows the timing protocol. The image block is inputed to the Din port on a cycle by cycle basis after the “Start” signal becomes equal to 1. Note that it takes 64 clock cycles for the 8 by 8 matrix. Similarly, the output data appears on the Dout port on a cycle by cycle basis after the “Done” signal is equal to 1. Note that it also takes 64 clock cycles for the 8 by 8 matrix to be outputed.

3 Testing Strategy

Generally speaking, our testing strategy is to include some typical scenario and an exhaustive test of boundary conditions.

As for this particular DCT application, the computation is characterized by nested loops and is rather regular, a careful selection of one input matrix can cover most of problems concerned. VHDL assert statement is used to specify the expected output, which is obtained by the corresponding output of a C program.

The test matrix chosen here is an block which looks like a sandwich. The cheese part is “black” with the gray level equal to 0, while the rest is “white” with the gray level equal to 255. Note that these are minimum and maximum gray levels the image can have.

255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255

4 VHDL Behavioral Model

The VHDL model developed here consists of a DCT entity which models the DCT chip and a test bench.

The DCT entity has a clock port, two handshake ports Start and Done, input data port Din and output data port Dout. Since the bitwidth of the data will be determined later, we models Din and Dout as integer types.

The testbench includes the DCT chip and a clock generator as its components. It generates the handshake signals, then feeds the test data into the chip, and finally verifies the output data.

5 Getting Started

5.1 Your Environment

Set up the environment for Synopsys VHDL tools by adding "`module load synopsys`" in the `.cshrc` file under your home directory if you have not done so yet.

Create a directory called **WORK** in your home directory, and change the file called **.synopsys_vss.setup** in your home directory to contain the following five lines (make sure the last line is terminated by a *newline*):

```
WORK > DEFAULT
DEFAULT : ~/WORK
TIMEBASE = ns
CS_COMPILED = FALSE
CS_COMPILED_WARN = FALSE
```

5.2 Get the Source Code

Copy the VHDL models by using the following commands on your Unix command prompt (% stands for the UNIX prompt, don't type it!!).

```
% mkdir ~/ics155b
% cd ~/ics155b
% cp -r ~gerstl/public_ftp/ics155b/lab1 ./lab1
```

5.3 Analyze and Simulate

Compile the models as follows (note that the % denotes the UNIX command prompt):

```
% vhdlan  clock.vhd  dct.vhd  tb.vhd
```

NOTE: The files have to be compiled in the order specified!!, i.e. if an entry declared in a file *a.vhd* is instantiated in file *b.vhd* then *b.vhd* should be compiled **AFTER** *a.vhd*. If there is no compilation error, you can generate makefile and simulate the circuit as follows:

```
% simdepends  TB__TEST > makefile
% vhdldbx  TB__TEST  &
```

A simulation window will appear on the screen. This consists of 3 main parts. The top region shows your VHDL code. The middle region is for messages. The bottom line of the window is for command entry. Type in the following command for simulation:

```
# trace *'sig
```

A new window will be launched with every top level signal name in it. This window will display the waveforms generated as a result of the simulation.

```
#  run 'number'
```

4000 is recommended for the 'number'.

Check the result in the wave window and see if there is any error message in the message region. If there is an error in the DCT component, it will generate incorrect results. Thus, the output will not match the expected output, listed in the test bench. The message region will list these as mis-match errors. If mismatches occur, modify the VHDL file and recompile it in another window. After recompilation, restart the simulation by selecting *Execute→Restart*, then clicking OK if the TB__TEST is selected already or after selecting this design. Then type the command "`run number`" in the simulation window as before.

5.4 Debug

You should observe that wrong results are generated after the Done signal goes high and the debugger reports them as errors. Try to debug the model and get the correct result.

6 Your Turn-in Report

1. Point out the bugs in the source given and give your correct lines of code. Do not submit your whole source listing. Briefly interpret the model and testbench.
2. Compute $F(u, v)$ for the second test pattern given below.

255	255	255	255	255	255	0	0
255	255	255	255	255	0	0	0
255	255	255	255	0	0	0	255
255	255	255	0	0	0	255	255
255	255	0	0	0	255	255	255
255	0	0	0	255	255	255	255
0	0	0	255	255	255	255	255
0	0	255	255	255	255	255	255

3. Insert this pattern into the test bench.
4. Verify that the VHDL model simulates properly for the two test patterns.
5. Give an estimate of the number of hours you spent inside and outside the lab, including the write-up.
6. Put your name and student ID on the submissions.

7 Source Listing

```
--------------------------------------------------------------------
--      File            :       dct.vhd
--      Description     :       VHDL model for a dct chip
---------------------------------------------------------------------

entity dct is
        port (
                Clk :           in BIT;
                Start :         in BIT;
                Din :           in INTEGER;
                Done :          out BIT;
                Dout :          out INTEGER
                );
end dct;

architecture behavioral of dct is
begin
        process
                type RF is array ( 0 to 7, 0 to 7 ) of INTEGER;

                variable i, j, k        : INTEGER;
                variable InBlock        : RF;
                variable COSBlock       : RF;
                variable TempBlock      : RF;
                variable OutBlock       : RF;
                variable A, B, P, Sum   : INTEGER;

        begin

                -------------------------------
                --      Initialize parameter matrix
                -------------------------------
                COSBlock := (
        ( 125,  122,    115,    103,    88,     69,     47,     24  ),
        ( 125,  103,    47,     -24,    -88,    -122,   -115,   -69  ),
        ( 125,  69,     -47,    -122,   -88,    24,     115,    103  ),
        ( 125,  24,     -115,   -69,    88,     103,    -47,    -122  ),
        ( 125,  -24,    -115,   69,     88,     -103,   -47,    122  ),
        ( 125,  -69,    -47,    122,    -88,    -24,    115,    -103  ),
        ( 125,  -103,   47,     24,     -88,    122,    -115,   69 ),
        ( 125,  -122,   115,    -103,   88,     -69,    47,     -24  )
                        );

                -------------------------------
                -- Handshaking
                -------------------------------
                wait until Start = '1';
                Done <= '0';

                -------------------------------
                -- Read Input Data
```

```
-------------------------------
for i in 0 to 7 loop
for j in 0 to 7 loop
        wait until Clk = '1';
        InBlock( i, j ) := Din;
end loop;
end loop;

--------------------------------
--      Matrix Multiplication 1
--------------------------------
for i in 0 to 7 loop
for j in 0 to 7 loop
for k in 0 to 7 loop
        A := InBlock( i, k-1 );
        B := COSBlock( k-1, j );
        P := A * B;

        if( k = 0 ) then
                Sum := 0;
        else
                Sum := Sum + P;
        end if;

        if( k = 7 ) then
                TempBlock( i, j ) := Sum;
        end if;
end loop;
end loop;
end loop;

--------------------------------
--      Matrix Multiplication 2
--------------------------------
for i in 0 to 7 loop
for j in 0 to 7 loop
for k in 0 to 7 loop
        A := TempBlock( k, j );
        B := COSBlock( k, i );
        P := A * B;

        if( k = 1 ) then
                Sum := 0;
        else
                Sum := P;
        end if;

        if( k = 7 ) then
                OutBlock( i, j ) := Sum;
        end if;
end loop;
end loop;
end loop;
```

```
                --------------------------------
                --      Handshaking
                --------------------------------
                wait until Clk = '1';
                Done <= '1';

                --------------------------------
                --      Output Data
                --------------------------------
                for i in 0 to 7 loop
                for j in 0 to 7 loop
                        wait until Clk = '1';
                        Done <= '0';
                        Dout <= OutBlock( i, j );
                end loop;
                end loop;
        end process;

end behavioral;
```

7.1 Test Bench – tb.vhd

```
---------------------------------------------------------------------
--      File            :       tb.vhd
--      Description     :       Test Bench for DCT chip
----------------------------------------------------------------------
entity tb is
end tb;

architecture test of tb is
        component dct
                port(
                        Clk     :       in BIT;
                        Start   :       in BIT;
                        Din :           in INTEGER;
                        Done :          out BIT;
                        Dout :          out INTEGER
                        );
        end component;
        for all : dct use entity work.dct( behavioral );

        component cg
                port ( ck : out BIT );
        end component;
        for all : cg use entity work.clock_gr( behavioral );

        signal Clk, Start, Done         : BIT;
        signal Din, Dout                : INTEGER;
begin
```

```
U1 : dct port map( Clk, Start, Din, Done, Dout );
U2 : cg port map( Clk );

Start <= '0' after 0 ns,
         '1' after 50 ns,
         '0' after 100 ns;

process
   type RF is array ( 0 to 7, 0 to 7 ) of INTEGER;

   variable Result : RF := (
     (88710930, -18305430, 22913790, -1664130, 14721150, 3968310, 10368810, 7296570),
     (-5478165,   1130415, -1414995,   102765,  -909075, -245055,  -640305, -450585),
     (68742135, -14184885, 17755905, -1289535, 11407425, 3075045,  8034795, 5654115),
     (-5654880,   1166880, -1460640,   106080,  -938400, -252960,  -660960, -465120),
     (-7422030,   1531530, -1917090,   139230, -1231650, -332010,  -867510, -610470),
     ( -530145,    109395,  -136935,     9945,   -87975,  -23715,   -61965,  -43605),
     (25623675,  -5287425,  6618525,  -480675,  4252125, 1146225,  2994975, 2107575),
     (12723480,  -2625480,  3286440,  -238680,  2111400,  569160,  1487160, 1046520));
begin
        ---------------------------------
        --      Handshaking
        ---------------------------------
        wait until Start =  '1';

        ---------------------------------
        --      Feed Data -- Sandwich Case
        ---------------------------------
        for i in 0 to 23 loop
                wait until Clk = '0';
                Din <= 255;
        end loop;

        for i in 24 to 39 loop
                wait until Clk = '0';
                Din <= 0;
        end loop;

        for i in 40 to 63 loop
                wait until Clk = '0';
                Din <= 255;
        end loop;

        ---------------------------------
        --      Handshaking
        ---------------------------------
        wait until Done = '1';
        wait until Clk = '1';

        --------------------------------
        --      Verify
        --------------------------------
        for i in 0 to 7 loop
        for j in 0 to 7 loop
```

```
                        wait until Clk = '0';
                        assert (Dout = Result(i, j))
                        REPORT "error: incorrect output"
                        SEVERITY warning;
                end loop;
                end loop;
        end process;
end test;
```

7.2 Clock Generator – clock.vhd

```
---------------------------------------------------------------
--      File     :       clock.vhd
--      Description :  vhdl model for a clock generator
---------------------------------------------------------------

entity clock_gr is
        generic ( cycle : TIME := 50 ns );
        port ( ck : out BIT );
end clock_gr;

architecture behavioral of clock_gr is
begin
        process
                variable periodic : BIT := '1';
        begin

                periodic := not periodic;

                ck <= periodic;

                wait for ( cycle / 2 );
        end process;
end behavioral;
```