ICS 155B Spring 1998
Lab Assignment 4 and 5: RT Architecture

Assignment 4 due on June 3, 1998
Assignment 5 due on June 1,, 1998

Handed out: May 18, 1998

1 Introduction

In this assignment, you will be developing a structural model for the DCT chip in VHDL. You are provided with a primitive Register Transfer (RT) component library. The VHDL model and the delay for each component is given. You will be instantiating components from the library and developing the chip model. You will learn the hierarchial design methodology in structural VHDL by developing and simulating models for the DCT chip.

This assignment has 2 parts. The first part is due 1 week from now (on June 3rd) and the second part is due 4 weeks from now (on June 12th).

2 Structural Model

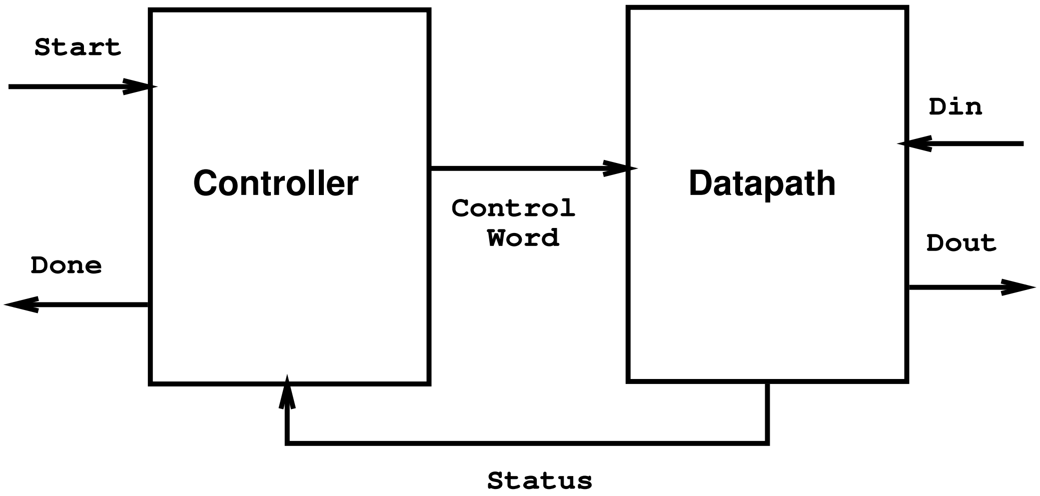

Figure 1: The DCT Chip

As shown in Figure 1, the structural view of the sequential design is decomposed into two parts, namely controller and datapath. The controller generates a set of control signals grouped as *control word* which controls the datapath. The datapath send a *status word* to the controller. The status signals helps in determining the next state and control signals generated. In the DCT example, the status word consists of the nine bits of the count register.

3 Datapath Routing

The top-level view of the DCT chip is shown in Figure 2. Note that the entire design has been partitioned into a controller and a datapath. The controller is in-turn modelled as shown in Figure 3. The controller is a finite-state machine with a State Register (described in the `str.vhd` file), Next State and Output Logic (described in the `logic.vhd` file).

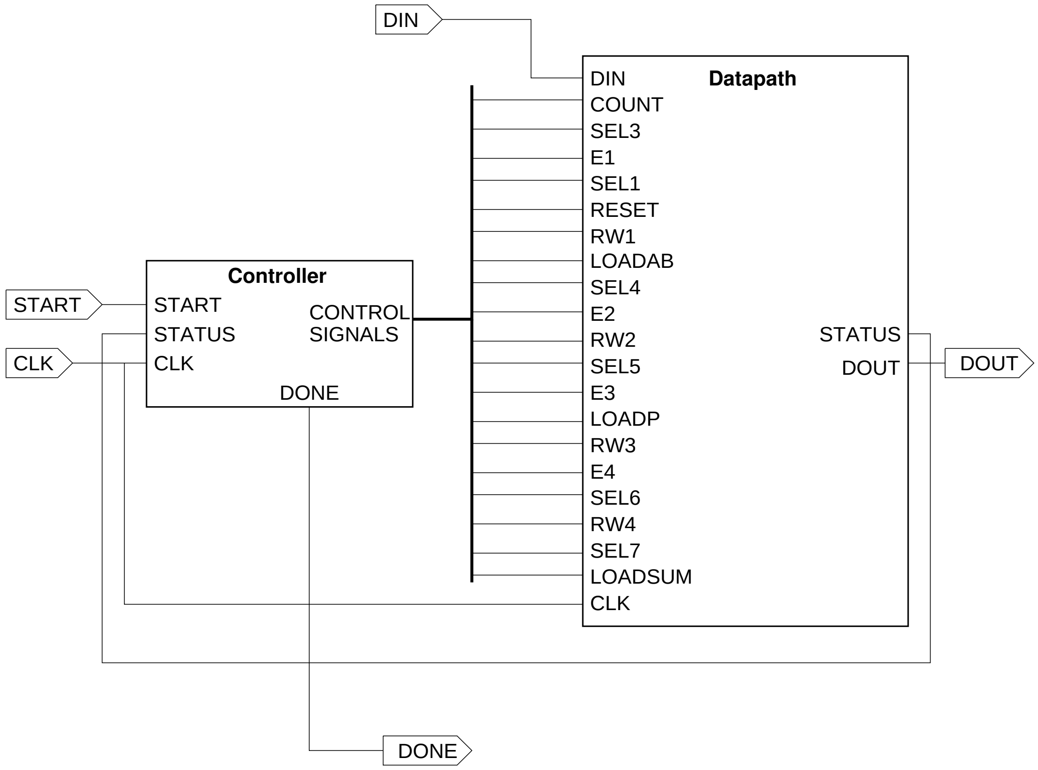

Figure 2: Hierarchical view of the DCT chip

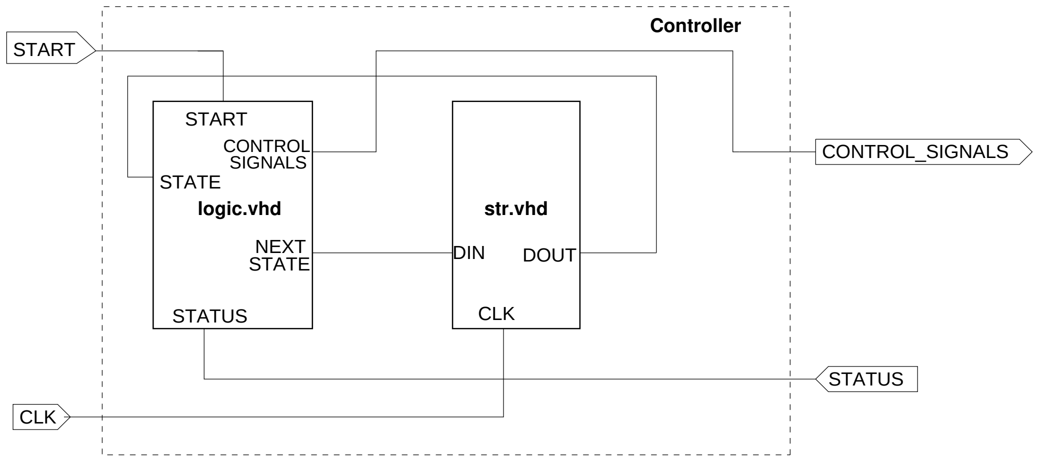

Figure 3: The controller for the DCT chip

In the first part of this lab assignment, you are required to figure out the datapath interconnections given the controller model and the unconnected datapath shown in Figure 4 executing the behavior given in Figure 5. You are required to finish an incomplete datapath schematic drawn in Synopsys Schematic Editor. For this part you just need to complete the schematic. All other parts of the DCT chip are provided and are correct. Finish the wiring and generate the VHDL code for the datapath. The code for the datapath can generated easily using the Synopsys tools if the datapath has been wired correctly (as done in the first tutorial assignment). Compile the code for the datapath, the controller with the test bench together and observe the simulation results.

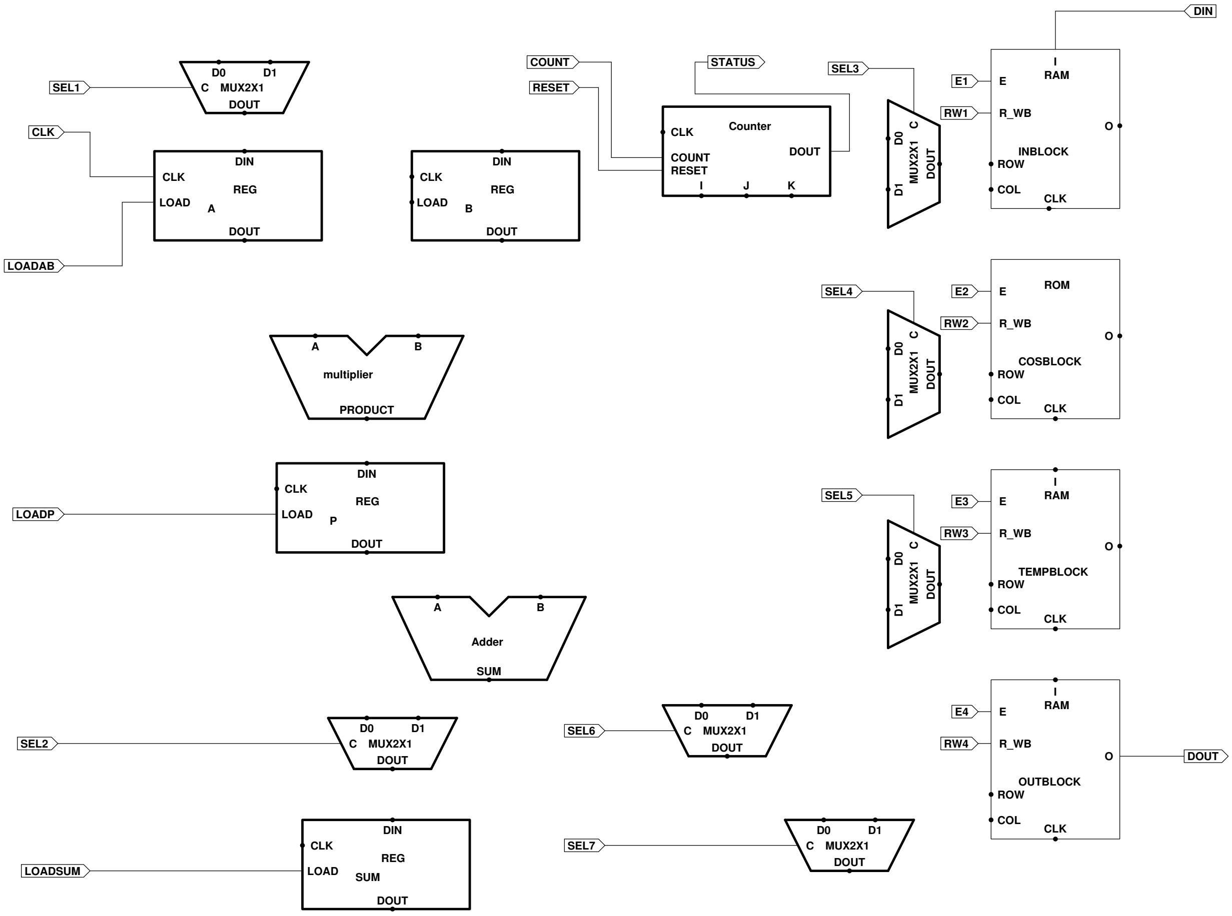

Figure 4: Unfinished Datapath schematic drawn in Synopsys Schematic Editor

4 Performance Optimization

In Lab assignment 2, you used Chaining, Multicycling and Loop Unrolling to improve the performance. In Lab assignment 3, you used Pipelining to further improve the performance of the design. In this

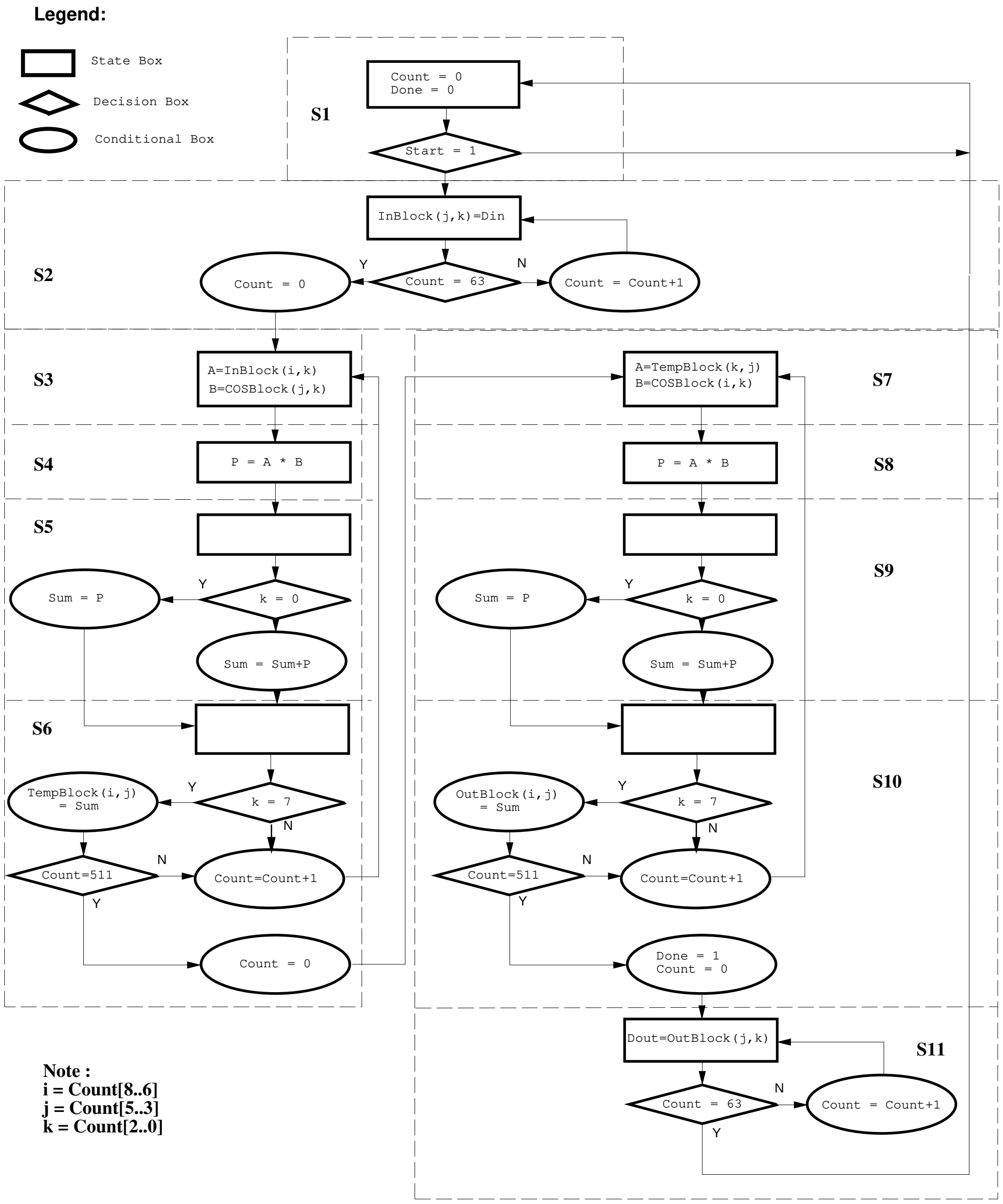

Figure 5: ASM chart for the sequential behavior of the DCT chip

part, you will design and simulate a structural model for the controller and the datapath for the best optimized DCT design that you developed in assignment 3. You may need to change the datapath schematic that you completed for part 1 of this assigment. The controller model will then need to be changed inorder to provide the correct control signals for the optimized design.

5 Assignment for Lab 4

1. Familiarize yourself with the design (controller and datapath) given in the Appendix.
2. **Study the VHDL model for each component given** in the Appendix and the files (`ram.vhd, rom.vhd, reg.vhd, str.vhd` etc. Pay attention to the delay of each component. Study the higher level models as well as their symbols. Understand the hierarchy of the structural model of the DCT chip.
3. Read the `logic.vhd` file carefully. This file has the Next State and Output Logic for the controller. Go through these files and **figure out the wiring of the datapath schematic according to the given controller model**.
4. Write the structural model of the controller and the datapath using components given and the diagrams provided. Use the Synopsys Schematic Editor in the Synopsys Graphical Editor to wire the datapath and generate the VHDL source code for the datapath.
5. **Generate the VHDL code** for the wired datapath using the appropriate commands in the Synopsys Graphical Environment, **compile and simulate the design**. In order to find all the critical paths, determine the maximal delay and the minimal clock period to execute your design.

6 Assignment for Lab 5

1. Repeat Lab 4 for your best optimized design, from the previous assignments, which is obtained via techniques like Loop Unrolling, Chaining, Multicycling and Pipeling.
2. Modify the datapath schematic for the new design with the optimizations that you come up with.
3. Modifiy the controller to generate the correct control signals for the new design. Add/remove components from the datapath of Lab 4 and change the component models if required.

7 Report for Lab 4

1. Printout of the the complete wired datapath schematic (for the sequential design without optimizations) along with 2 pages of waveforsms that show the performance and correctness of the result.
2. Do not submit any code that was not changed by you in this lab assignment.

3. Mail an electronic copy of your VHDL code (datapath schematic) to the TA.

4. Give an estimate of the number of hours you spent inside and outside the lab, including the write-up.

5. Put your name and student ID on the submissions.

8 Report for Lab 5

1. List all the optmization techniques that you have use for improving performance.

2. Sumbit the new datapath schematic and the VHDL code for the new controller. If you modified the ports, delays, component symbols, functionality of the component models given or any other files given to you, document the modifications and submit the new compoment model/VHDL code or whatever was changed.

3. Do not submit any code that was not changed by you in this lab assignment.

4. Mail an electronic copy of your VHDL code (datapath schematic and the controller) to the TA.

5. Write a short discussion on the optimizations done and the performance of the chip versus hardware costs etc.

6. Give an estimate of the number of hours you spent inside and outside the lab, including the write-up.

7. Put your name and student ID on the submissions.

A Source Listing for Component Library

A.1 comp.vhd

```
--------------------------------------------------------------------
--      File            :       comp.vhd
--      Description     :       Package for declaration of some RT components
---------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_MISC.all;
use IEEE.STD_LOGIC_ARITH.all;

package COMPONENTS is

subtype INT is INTEGER;

-------------------------------------------------------
--        S1 -- Idle State
--        S2 -- Load Data
```

```
--        S3-S7 -- Loop of First Matrix Multiplication
--        S8-S11 -- Loop of Second Matrix Multiplication
--        S12 -- Output Data
-------------------------------------------------------
type STATE_VALUE is ( S1, S2, S3, S4, S5, S6,
        S7, S8, S9, S10, S11 );

        function int_to_uvec(number, length : integer) return unsigned;
end COMPONENTS;

package body COMPONENTS is

        function int_to_uvec(number, length : integer) return unsigned is
                variable temp : unsigned(length -1 downto 0);
                variable num, quotient : integer := 0;
        begin
                quotient := number;

                for i in 0 to length -1 loop
                        num := 0;
                        while quotient > 1 loop
                                quotient := quotient - 2;
                                num := num + 1;
                        end loop;

                        case quotient is
                                when 1 => temp(i) := '1';
                                when 0 => temp(i) := '0';
                                when others => null;
                        end case;

                        quotient := num;
                end loop;

                return temp;
        end int_to_uvec;

end COMPONENTS;
```

A.2 adder.vhd

```
library IEEE;
   use IEEE.std_logic_1164.all;
   use IEEE.std_logic_misc.all;
   use IEEE.std_logic_arith.all;
   use IEEE.std_logic_components.all;
   use work.components.all;


entity ADDER is
      generic ( Delay:   Time := 7 ns );
```

```
      Port (       A : In    INT;
                   B : In    INT;
                 SUM : Out   INT );
end ADDER;

architecture BEHAVIORAL of ADDER is

   begin
        P: process ( A, B )
                variable Result : INT := 0;
        begin
                if( A /= INT'left and B /= INT'left ) then
                        Result := A + B;
                end if;
                SUM <= Result after Delay;
        end process P;

end BEHAVIORAL;

configuration CFG_ADDER_BEHAVIORAL of ADDER is
   for BEHAVIORAL

   end for;

end CFG_ADDER_BEHAVIORAL;
```

A.3 multiplier.vhd

```
library IEEE;
   use IEEE.std_logic_1164.all;
   use IEEE.std_logic_misc.all;
   use IEEE.std_logic_arith.all;
   use IEEE.std_logic_components.all;
   use work.components.all;


entity MULTIPLIER is
      generic ( Delay:   Time := 40 ns );
      Port (       A : In    INT;
                   B : In    INT;
             PRODUCT : Out   INT );
end MULTIPLIER;

architecture BEHAVIORAL of MULTIPLIER is

   begin
        P: process ( A, B )
                variable Result : INT := 0;
        begin
                if( A /= INT'left and B /= INT'left ) then
                        Result := A * B;
```

```
                end if;
                PRODUCT <= Result after Delay;
        end process P;
end BEHAVIORAL;

configuration CFG_MULTIPLIER_BEHAVIORAL of MULTIPLIER is
   for BEHAVIORAL

   end for;

end CFG_MULTIPLIER_BEHAVIORAL;
```

A.4 mux2x1.vhd

```
library IEEE;
   use IEEE.std_logic_1164.all;
   use IEEE.std_logic_misc.all;
   use IEEE.std_logic_arith.all;
   use IEEE.std_logic_components.all;
   use work.components.all;

entity MUX2X1 is
      generic ( Delay:   Time := 5 ns );
      Port (       C : In    std_logic;
                  D0 : In    INT;
                  D1 : In    INT;
                DOUT : Out   INT );
end MUX2X1;

architecture BEHAVIORAL of MUX2X1 is

   begin
        P: process ( D0, D1, C )
                variable data_out : INT;
        begin
                if( C = '0' ) then
                        data_out := D0;
                else
                        data_out := D1;
                end if;
                Dout <= data_out after Delay;
        end process P;
end BEHAVIORAL;

configuration CFG_MUX2X1_BEHAVIORAL of MUX2X1 is
   for BEHAVIORAL

   end for;

end CFG_MUX2X1_BEHAVIORAL;
```

A.5 reg.vhd

```
library IEEE;
   use IEEE.std_logic_1164.all;
   use IEEE.std_logic_misc.all;
   use IEEE.std_logic_arith.all;
   use IEEE.std_logic_components.all;
   use work.components.all;

entity REG is
        generic (
                Delay   : Time := 5 ns;
                Setup   : Time := 0 ns;
                Hold    : Time := 0 ns
                );
      Port (     CLK : In    std_logic;
                 DIN : In    INT;
                LOAD : In    std_logic;
                DOUT : Out   INT );
end REG;

architecture BEHAVIORAL of REG is

   begin
        P: process
                variable Value : INT := 0;
        begin
                wait on Clk;
                if( Clk'event and Clk = '1' ) then
                        assert ( Din'Stable(Setup) )
                                REPORT "Setup Time  Constraint Violated!"
                                SEVERITY warning;
                        if( Load = '1' ) then
                                wait for Hold;
                                assert ( Din'Stable(Hold) )
                                        REPORT "Hold Time Constraint Violated!"
                                        SEVERITY warning;
                                Value := Din;
                        end if;
                end if;

                Dout <= Value after (Delay-Hold);
        end process P;
end BEHAVIORAL;

configuration CFG_REG_BEHAVIORAL of REG is
   for BEHAVIORAL

   end for;
```

```
end CFG_REG_BEHAVIORAL;
```

A.6 counter.vhd

```
library IEEE;
   use IEEE.std_logic_1164.all;
   use IEEE.std_logic_misc.all;
   use IEEE.std_logic_arith.all;
   use IEEE.std_logic_components.all;
   use work.components.all;

entity COUNTER is
        generic ( Delay:   Time := 15 ns );
      Port (     CLK : In    std_logic;
               COUNT : In    std_logic;
               RESET : In    std_logic;
                DOUT : Out   INT;
                   I : Out   INT;
                   J : Out   INT;
                   K : Out   INT );
end COUNTER;

architecture BEHAVIORAL of COUNTER is

   begin
        P: process ( Clk )
                variable Value : UNSIGNED( 8 downto 0 ) := "000000000";
        begin
                if( Clk'event and Clk = '1' ) then
                        if( Reset = '1' ) then
                                Value := "000000000";
                        elsif( Count = '1' ) then
                                Value := Value + 1;
                        end if;
                end if;
                Dout <= CONV_INTEGER( Value ) after Delay;
                I <= CONV_INTEGER( Value( 8 downto 6 ) ) after Delay;
                J <= CONV_INTEGER( Value( 5 downto 3 ) ) after Delay;
                K <= CONV_INTEGER( Value( 2 downto 0 ) ) after Delay;
        end process P;

end BEHAVIORAL;

configuration CFG_COUNTER_BEHAVIORAL of COUNTER is
   for BEHAVIORAL

   end for;

end CFG_COUNTER_BEHAVIORAL;
```

A.7 rom.vhd

```
library IEEE;
   use IEEE.std_logic_1164.all;
   use IEEE.std_logic_misc.all;
   use IEEE.std_logic_arith.all;
   use IEEE.std_logic_components.all;
   use work.components.all;

entity ROM is
        generic ( Delay:   Time := 15 ns );
      Port (     CLK : In    std_logic;
                 COL : In    INT;
                   E : In    std_logic;
                R_WB : In    std_logic;
                 ROW : In    INT;
                   O : Out   INT );
end ROM;

architecture BEHAVIORAL of ROM is

begin
        P: process( Clk )
                type RF is array ( 0 to 7, 0 to 7 ) of INT;
                variable Storage : RF := (
                        ( 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 )
                        );
                variable Index_r, Index_c : INT;
        begin

        if( Clk'event and Clk = '0' ) then
                if( E = '1' ) then
                        Index_r := CONV_INTEGER( ROW );
                        Index_c := CONV_INTEGER( COL );
                        if( R_WB = '1' ) then
                        -- read --
                                O <= Storage( Index_r, Index_c ) after Delay;
                        end if;
                else
                        O <= INT'left;
                end if;
        end if;
        end process P;

end BEHAVIORAL;
```

```
configuration CFG_ROM_BEHAVIORAL of ROM is
   for BEHAVIORAL

   end for;

end CFG_ROM_BEHAVIORAL;
```

A.8 ram.vhd

```
library IEEE;
   use IEEE.std_logic_1164.all;
   use IEEE.std_logic_misc.all;
   use IEEE.std_logic_arith.all;
   use IEEE.std_logic_components.all;
   use work.components.all;

entity RAM is
      generic ( Delay:   Time := 15 ns );
      Port (     CLK : In    std_logic;
                 COL : In    INT;
                   E : In    std_logic;
                   I : In    INT;
                R_WB : In    std_logic;
                 ROW : In    INT;
                   O : Out   INT );
end RAM;

architecture BEHAVIORAL of RAM is

begin
        P: process( Clk )
                type RF is array ( 0 to 7, 0 to 7 ) of INT;
                variable Storage : RF := (
                        ( 0, 0, 0, 0, 0, 0, 0, 0 ),
                        ( 0, 0, 0, 0, 0, 0, 0, 0 ),
                        ( 0, 0, 0, 0, 0, 0, 0, 0 ),
                        ( 0, 0, 0, 0, 0, 0, 0, 0 ),
                        ( 0, 0, 0, 0, 0, 0, 0, 0 ),
                        ( 0, 0, 0, 0, 0, 0, 0, 0 ),
                        ( 0, 0, 0, 0, 0, 0, 0, 0 ),
                        ( 0, 0, 0, 0, 0, 0, 0, 0 )
                        );
                variable Index_r, Index_c : INT;
        begin

        if( Clk'event and Clk = '0' ) then
                if( E = '1' ) then
                        Index_r := CONV_INTEGER( ROW );
                        Index_c := CONV_INTEGER( COL );
                        if( R_WB = '1' ) then
```

```
                        -- read --
                                O <= Storage( Index_r, Index_c ) after Delay;
                        else
                        -- write --
                                Storage( Index_r, Index_c ) := I;
                        end if;
                else
                        O <= INT'left;
                end if;
        end if;
        end process P;

end BEHAVIORAL;

configuration CFG_RAM_BEHAVIORAL of RAM is
   for BEHAVIORAL

   end for;

end CFG_RAM_BEHAVIORAL;
```

A.9 controller.vhd

```
library IEEE;
   use IEEE.std_logic_1164.all;
   use IEEE.std_logic_misc.all;
   use IEEE.std_logic_arith.all;
   use IEEE.std_logic_components.all;
   use work.components.all;

entity CONTROLLER is
      Port (     CLK : In    std_logic;
               START : In    std_logic;
              STATUS : In    INTEGER;
               COUNT : Out   std_logic;
                DONE : Out   std_logic;
                  E1 : Out   std_logic;
                  E2 : Out   std_logic;
                  E3 : Out   std_logic;
                  E4 : Out   std_logic;
              LOADAB : Out   std_logic;
               LOADP : Out   std_logic;
             LOADSUM : Out   std_logic;
               RESET : Out   std_logic;
                 RW1 : Out   std_logic;
                 RW2 : Out   std_logic;
                 RW3 : Out   std_logic;
                 RW4 : Out   std_logic;
                SEL1 : Out   std_logic;
                SEL2 : Out   std_logic;
                SEL3 : Out   std_logic;
                SEL4 : Out   std_logic;
```

```
                SEL5 : Out   std_logic;
                SEL6 : Out   std_logic;
                SEL7 : Out   std_logic );
end CONTROLLER;

architecture SCHEMATIC of CONTROLLER is

   signal      N_1 : STATE_VALUE;
   signal      N_2 : STATE_VALUE;

   component STR
      Port (     CLK : In    std_logic;
                 DIN : In    STATE_VALUE;
                DOUT : Out   STATE_VALUE );
   end component;

   component LOGIC
      Port (   START : In    std_logic;
               STATE : In    STATE_VALUE;
              STATUS : In    INTEGER;
               COUNT : Out   std_logic;
                DONE : Out   std_logic;
                  E1 : Out   std_logic;
                  E2 : Out   std_logic;
                  E3 : Out   std_logic;
                  E4 : Out   std_logic;
              LOADAB : Out   std_logic;
               LOADP : Out   std_logic;
             LOADSUM : Out   std_logic;
             NEXTSTATE : Out   STATE_VALUE;
               RESET : Out   std_logic;
                 RW1 : Out   std_logic;
                 RW2 : Out   std_logic;
                 RW3 : Out   std_logic;
                 RW4 : Out   std_logic;
                SEL1 : Out   std_logic;
                SEL2 : Out   std_logic;
                SEL3 : Out   std_logic;
                SEL4 : Out   std_logic;
                SEL5 : Out   std_logic;
                SEL6 : Out   std_logic;
                SEL7 : Out   std_logic );
   end component;

begin

   I_2 : STR
      Port Map ( CLK=>CLK, DIN=>N_2, DOUT=>N_1 );
   I_3 : LOGIC
      Port Map ( START=>START, STATE=>N_1, STATUS=>STATUS, COUNT=>COUNT,
                 DONE=>DONE, E1=>E1, E2=>E2, E3=>E3, E4=>E4,
                 LOADAB=>LOADAB, LOADP=>LOADP, LOADSUM=>LOADSUM,
                 NEXTSTATE=>N_2, RESET=>RESET, RW1=>RW1, RW2=>RW2,
                 RW3=>RW3, RW4=>RW4, SEL1=>SEL1, SEL2=>SEL2, SEL3=>SEL3,
```

```
                 SEL4=>SEL4, SEL5=>SEL5, SEL6=>SEL6, SEL7=>SEL7 );

end SCHEMATIC;

configuration CFG_CONTROLLER_SCHEMATIC of CONTROLLER is

   for SCHEMATIC
      for I_2: STR
         use configuration WORK.CFG_STR_BEHAVIORAL;
      end for;
      for I_3: LOGIC
         use configuration WORK.CFG_LOGIC_BEHAVIORAL;
      end for;
   end for;

end CFG_CONTROLLER_SCHEMATIC;
```

A.10 str.vhd

This file has the source code for the State Register used inside the controller for storing the current state of the controller.

```
library IEEE;
   use IEEE.std_logic_1164.all;
   use IEEE.std_logic_misc.all;
   use IEEE.std_logic_arith.all;
   use IEEE.std_logic_components.all;
   use work.components.all;

entity STR is
        generic (
                Delay   : Time := 5 ns;
                Setup   : Time := 0 ns;
                Hold    : Time := 0 ns
        );
      Port (     CLK : In    std_logic;
                 DIN : In    STATE_VALUE;
                DOUT : Out   STATE_VALUE );
end STR;

architecture BEHAVIORAL of STR is
   begin
        P: process
                variable Value : STATE_VALUE := S1;
        begin
                wait on Clk;
                if( Clk'event and Clk = '1' ) then
                        assert ( Din'Stable(Setup) )
                                REPORT "Setup Time Constraint Violated!"
                                SEVERITY warning;
```

```
                        wait for Hold;
                        assert ( Din'Stable(Hold) )
                                REPORT "Hold Time Constraint Violated!"
                                SEVERITY warning;
                        Value := Din;
                end if;

                Dout <= Value after (Delay-Hold);
        end process P;
end BEHAVIORAL;

configuration CFG_STR_BEHAVIORAL of STR is
   for BEHAVIORAL
   end for;

end CFG_STR_BEHAVIORAL;
```

A.11 logic.vhd

This file has the VHDL source code for the Next State and Output Logic used inside the controller for determining the next state and output of the controller.

```
library IEEE;
   use IEEE.std_logic_1164.all;
   use IEEE.std_logic_misc.all;
   use IEEE.std_logic_arith.all;
   use IEEE.std_logic_components.all;
   use work.components.all;


entity LOGIC is
   generic ( Delay   : Time := 5 ns);
      Port (   START : In    std_logic;
               STATE : In    STATE_VALUE;
              STATUS : In    INT;
           NEXTSTATE : Out   STATE_VALUE;
               COUNT : Out   std_logic := '0';
                  E1 : Out   std_logic := '0';
                  E2 : Out   std_logic := '0';
                  E3 : Out   std_logic := '0';
                  E4 : Out   std_logic := '0';
              LOADAB : Out   std_logic := '0';
               LOADP : Out   std_logic := '0';
             LOADSUM : Out   std_logic := '0';
               RESET : Out   std_logic := '0';
                 RW1 : Out   std_logic := '0';
                 RW2 : Out   std_logic := '0';
                 RW3 : Out   std_logic := '0';
                 RW4 : Out   std_logic := '0';
                SEL1 : Out   std_logic := '0';
                SEL2 : Out   std_logic := '0';
```

```
                SEL3 : Out   std_logic := '0';
                SEL4 : Out   std_logic := '0';
                SEL5 : Out   std_logic := '0';
                SEL6 : Out   std_logic := '0';
                SEL7 : Out   std_logic := '0';
                DONE : Out   std_logic );

end LOGIC;

architecture BEHAVIORAL of LOGIC is

begin
        process( State, Start, Status )
                variable  NextStateValue : STATE_VALUE := S1;
                variable       Cnt : unsigned(8 downto 0);
                variable   VarDone : STD_LOGIC;
                variable   i, j, k : INT;

        begin
            Cnt := int_to_uvec(STATUS, 9);
            if( Cnt(0) /= 'U' and Start /= 'U' ) then
                i := CONV_INTEGER( Cnt( 8 downto 6 ) );
                j := CONV_INTEGER( Cnt( 5 downto 3 ) );
                k := CONV_INTEGER( Cnt( 2 downto 0 ) );

                     -- Default Values :
                     -- Control Signals for Muxes
                        Sel1 <= '0';
                        Sel2 <= '0';
                        Sel3 <= '0';
                        Sel4 <= '0';
                        Sel5 <= '0';
                        Sel6 <= '0';
                        Sel7 <= '0';

                        -- Control Signals for Registers
                        LoadAB <= '0';
                        LoadP <= '0';
                        LoadSum <= '0';

                        -- Control Signal for Counter
                        Count <= '0';
                        Reset <= '0';

                        -- Control Signals for Memories
                        E1 <= '0';
                        RW1 <= '0';
                        E2 <= '0';
                        RW2 <= '0';
                        E3 <= '0';
                        RW3 <= '0';
                        E4 <= '0';
                        RW4 <= '0';
```

```
case State is
        --------------------------------------------
        --       State S1
        --------------------------------------------
        when S1 =>
                if( Start = '1' ) then
                        NextStateValue := S2;
                else
                        NextStateValue := S1;
                end if;

                -- Cnt := "000000000";
                Reset <= '1' after Delay;
                Count <= '0' after Delay;
                VarDone := '0';


        --------------------------------------------
        --       State S2
        --------------------------------------------
        when S2 =>
                if( Cnt = 63 ) then
                        -- Cnt := "000000000";
                        Reset <= '1' after Delay;
                        Count <= '0' after Delay;

                        NextStateValue := S3;
                else
                        -- Cnt := Cnt + 1;
                        Reset <= '0' after Delay;
                        Count <= '1' after Delay;

                        NextStateValue := S2;
                end if;

                -- InBlock( j, k ) :=  Din;
                Sel3 <= '1' after Delay;
                E1 <= '1' after Delay;
                RW1 <= '0' after Delay;


        --------------------------------------------
        --       State S3
        --------------------------------------------
        when S3 =>
                NextStateValue := S4;

                -- A := InBlock(i, k);
                LoadAB <= '1' after Delay;
                Sel1 <= '1' after Delay;

                Sel3 <= '0' after Delay;
                E1 <= '1' after Delay;
                RW1 <= '1' after Delay;
```

```
        -- B := COSBlock(j, k);
        Sel4 <= '0' after Delay;
        E2 <= '1' after Delay;
        RW2 <= '1' after Delay;



--------------------------------------------
--       State S4
--------------------------------------------
when S4 =>
        NextStateValue := S5;

        -- P := A * B;
        LoadP <= '1' after Delay;


--------------------------------------------
--       State S5
--------------------------------------------
when S5 =>
        NextStateValue :=  S6;

        if( k = 0 ) then
                -- Sum := P;
                LoadSum <= '1' after Delay;
                Sel2 <= '0' after Delay;
        else
                -- Sum := P + Sum;
                LoadSum <= '1' after Delay;
                Sel2 <= '1' after Delay;
        end if;


--------------------------------------------
--       State S6
--------------------------------------------
when S6 =>
        if( k = 7 ) then
                -- TempBlock(i, j) := Sum;
                Sel5 <= '0' after Delay;
                E3 <= '1' after Delay;
                RW3 <= '0' after Delay;
        end if;

        if( Cnt = 511 ) then
                -- Cnt := "000000000";
                Reset <= '1' after Delay;
                Count <= '0' after Delay;

                NextStateValue := S7;
        else
                -- Cnt := Cnt + 1;
```

```
                Reset <= '0' after Delay;
                Count <= '1' after Delay;

                NextStateValue := S3;
        end if;


--------------------------------------------
--       State S7
--------------------------------------------
when S7 =>
        NextStateValue := S8;

        -- A := TempBlock(k, j);
        LoadAB <= '1' after Delay;
        Sel1 <= '0' after Delay;
        E3 <= '1' after Delay;
        RW3 <= '1' after Delay;
        Sel5 <= '1' after Delay;

        -- B := COSBlock(i, k);
        Sel4 <= '1' after Delay;
        E2 <= '1' after Delay;
        RW2 <= '1' after Delay;



--------------------------------------------
--       State S8
--------------------------------------------
when S8 =>
        NextStateValue := S9;

        -- P := A * B;
        LoadP <= '1' after Delay;


--------------------------------------------
--       State S9
--------------------------------------------
when S9 =>
        NextStateValue :=  S10;

        if( k = 0 ) then
                -- Sum := P;
                LoadSum <= '1' after Delay;
                Sel2 <= '0' after Delay;
        else
                -- Sum <= P + Sum;
                LoadSum <= '1' after Delay;
                Sel2 <= '1' after Delay;
        end if;
```

```
-------------------------------------------
--       State S10
-------------------------------------------
when S10 =>

        if( k = 7 ) then
                -- OutBlock(i, j) := Sum;
                Sel6 <= '0' after Delay;
                Sel7 <= '0' after Delay;
                E4 <= '1' after Delay;
                RW4 <= '0' after Delay;
        end if;
        if( Cnt = 511 ) then
                -- Cnt := "000000000";
                Reset <= '1' after Delay;
                Count <= '0' after Delay;
                VarDone := '1';

                NextStateValue := S11;
        else
                -- Cnt := Cnt + 1;
                Reset <= '0' after Delay;
                Count <= '1' after Delay;

                NextStateValue := S7;

        end if;


-------------------------------------------
--       State S11
-------------------------------------------
when S11 =>

        VarDone := '0';
        if( Cnt = 63 ) then
                -- Cnt := "000000000";
                Reset <= '1' after Delay;
                Count <= '0' after Delay;

                NextStateValue := S1;
        else
                -- Cnt := Cnt + 1;
                Reset <= '0' after Delay;
                Count <= '1' after Delay;

                NextStateValue := S11;
        end if;

        -- Dout :=  OutBlock( j, k );
        Sel6 <= '1' after Delay;
        Sel7 <= '1' after Delay;
        E4 <= '1' after Delay;
        RW4 <= '1' after Delay;
```

```
                end case;

                NextState <= NextStateValue after Delay;
                Done <= VarDone after Delay;
            end if;

        end process;


end BEHAVIORAL;

configuration CFG_LOGIC_BEHAVIORAL of LOGIC is
   for BEHAVIORAL

   end for;

end CFG_LOGIC_BEHAVIORAL;
```

A.12 clock_gr.vhd

```
library IEEE;
   use IEEE.std_logic_1164.all;
   use IEEE.std_logic_misc.all;
   use IEEE.std_logic_arith.all;
   use IEEE.std_logic_components.all;

entity CLOCK_GR is
      generic ( cycle : TIME := 50 ns );
      Port (      CK : Out   std_logic );
end CLOCK_GR;

architecture BEHAVIORAL of CLOCK_GR is

begin
        process
                variable periodic : STD_LOGIC := '1';
        begin

                periodic := not periodic;

                ck <= periodic;

                wait for ( cycle / 2 );
        end process;

end BEHAVIORAL;

configuration CFG_CLOCK_GR_BEHAVIORAL of CLOCK_GR is
   for BEHAVIORAL
```

```
   end for;

end CFG_CLOCK_GR_BEHAVIORAL;
```