Appendix G

The Design of Multipath Searcher by Using VHDL Code

Program Description

Input:

- Reset signal.
- Frame synchronization signal.
- Clock signal.
- In-phase digital signal (6 bits).
- Quadrature digital signal (6 bits).
- Scrambling code number (24 bits).

Output:

- Frame synchronization output signal for the 2^{nd} finger (Time delay for the 2^{nd} finger).
- Frame synchronization output signal for the 3^{rd} finger (Time delay for the 3^{rd} finger).

```
LIBRARY ieee; USE ieee.std_logic_1164.all; USE
ieee.std_logic_arith.all; USE ieee.std_logic_signed.all;

ENTITY Block_ULmultipathsearcher IS
    Port (  reset : IN std_logic;
        FrameSync : IN std_logic;              -- Finger1's FrameSync
        clk : IN std_logic;                -- clk 15.36 MHz
        I_channel : IN std_logic_vector(5 downto 0);        -- Input I_channel
        Q_channel : IN std_logic_vector(5 downto 0);        -- Input Q_channel
        ScCode_number : IN std_logic_vector(23 downto 0);   -- Scrambling Code Number
        FrameSync_Finger2 : OUT std_logic;          -- Finger2's FrameSync (output)
        FrameSync_Finger3 : OUT std_logic           -- Finger3's FrameSync (output)
        );
END Block_ULmultipathsearcher;

ARCHITECTURE behavior OF Block_ULmultipathsearcher IS
    COMPONENT ULPS_Decimator
    PORT (  reset : IN std_logic;
        clk : IN std_logic;          -- input clk : 15.36 MHz
        FrmSync : IN std_logic;
        clkout : OUT std_logic;
        clkCodeOut : OUT std_logic;
        FrmSyncOut : INOUT std_logic
        );
    END COMPONENT;

    COMPONENT ULPS_ScLongCodeGen
    PORT (  clk : IN std_logic;
        reset : IN std_logic;
        FrameSync : IN std_logic;
        ScCode_number : IN std_logic_vector(23 downto 0);   -- 24-bit UL Scrambling Code Number.

        Code_I : OUT std_logic_vector(0 to 1);
        Code_Q : OUT std_logic_vector(0 to 1)
        );
    END COMPONENT;

    COMPONENT ULPS_PathSearcher
    Port (  reset : IN std_logic;
        FrameSync : IN std_logic;
        clk : IN std_logic;
        I_channel : IN std_logic_vector(5 downto 0);
        Q_channel : IN std_logic_vector(5 downto 0);
        clk_ChipCodeGen : IN std_logic;
        ScCode_I : IN std_logic_vector(0 to 1);
        ScCode_Q : IN std_logic_vector(0 to 1);

        SL0_FrameSync : OUT std_logic;
        SL1_FrameSync : OUT std_logic
        );
    END COMPONENT;

    -- (Inputs which are outputs of ULPS_Decimator)---
    signal clk_ChipCodeGen : std_logic := '0';
    signal clk_chip : std_logic := '0';     -- not be used
    signal FrameSync_chip : std_logic := '0';   -- not be used
    -- (Inputs which are outputs of ULPS_ScLongCodeGen) ---
    signal ScCode_I : std_logic_vector(0 to 1);
    signal ScCode_Q : std_logic_vector(0 to 1);

    FOR Ps_DECIMATOR : ULPS_Decimator USE ENTITY work.ULPS_Decimator(behavior);
    FOR Ps_CODEGEN : ULPS_ScLongCodeGen USE ENTITY work.ULPS_ScLongCodeGen(behavior);
    FOR Ps_UL_PATHSEARCH : ULPS_PathSearcher USE ENTITY work.ULPS_PathSearcher(behavior);
```

```
BEGIN
    Ps_DECIMATOR :
    ULPS_Decimator PORT MAP (reset,clk,FrameSync,clk_chip,clk_ChipCodeGen,FrameSync_chip);
    Ps_CODEGEN :
    ULPS_ScLongCodeGen PORT MAP (clk_ChipCodeGen,reset,FrameSync,ScCode_number,ScCode_I,ScCode_Q);
    Ps_UL_PATHSEARCH :
    ULPS_PathSearcher PORT MAP
    (reset,FrameSync,clk,I_channel,Q_channel,clk_ChipCodeGen,ScCode_I,ScCode_Q,FrameSync_Finger2,
    FrameSync_Finger3);
END behavior;

%%%%%

LIBRARY ieee; USE ieee.std_logic_1164.all;

ENTITY ULPS_Decimator IS
    PORT (  reset : IN std_logic;
        clk : IN std_logic;          -- input clk : 15.36 MHz
        FrmSync : IN std_logic;
        clkout : OUT std_logic;
        clkCodeOut : OUT std_logic;
        FrmSyncOut : INOUT std_logic
        );
END ULPS_Decimator ;

ARCHITECTURE behavior OF ULPS_Decimator IS
    signal count : integer range 0 to 4:=4;  -- max = 3, default value = 4
    signal ChipCount : integer range 0 to 38399:= 0;  --38399
    signal TmpClk1,TmpClk2 : std_logic:= '0';
    signal TmpF1,TmpF2 : std_logic:= '0';
    signal ChClk1,ChClk2 : std_logic:= '0';
    signal ChF1,ChF2 : std_logic:= '0';
    signal tick_en : integer range 0 to 2 := 0;
BEGIN
    FrmSyncOut <= ChF1;

    clkout <= ChF2 when ChF1='1' else
          (ChClk1 AND ChClk2);

    clkCodeOut <=  NOT(TmpF2) when TmpF1='1' else
             (TmpClk1 AND NOT(TmpClk2));

    PROCESS (reset,clk)
    BEGIN
        IF (reset = '1') THEN
            tick_en <= 0;
        ELSIF (clk'event AND clk='1') THEN
            if (FrmSync='1' AND tick_en<2 ) then
                tick_en <= tick_en + 1;
            end if;
        END IF;
    END PROCESS;


    PROCESS (reset,clk)
    BEGIN
        IF (reset = '1') THEN
            count <= 4;
            ChipCount <= 0;
            TmpClk1 <= '0';
            ChClk1 <= '0';
        ELSIF (clk'event AND clk='1') THEN
            IF (FrmSync='1') THEN
                count <= 0;
```

```
                ChipCount <= 0;
                TmpClk1 <= '0';
                ChClk1 <= '0';
            ELSIF ((count=3)AND(ChipCount<38399)) THEN
                count <= 0;
                ChipCount <= ChipCount+1;
                TmpClk1 <= '1';
                if (tick_en = 2) then
                    ChClk1 <= '1';
                end if;
            ELSE    --[ChipCount>=38399] && [count=[0,1,2,4,(3 & chipCount>=38399)]]
                IF (count<3) THEN
                    count <= count+1;
                END IF;
                TmpClk1 <= '0';
                ChClk1 <= '0';
            END IF;
        END IF;
    END PROCESS;


    PROCESS (clk)
    BEGIN
        if(clk'event AND clk='1') then
            TmpF1 <= FrmSync;
            if (tick_en > 0) then
                ChF1 <= FrmSync;
            end if;
        end if;
    END PROCESS;


    PROCESS (clk)
    BEGIN
        if(clk'event AND clk='0') then
            TmpClk2 <= TmpClk1;
            ChClk2 <= ChClk1;
            TmpF2 <= TmpF1;
            ChF2 <= ChF1;
        end if;
    END PROCESS;
END behavior;

%%%%

LIBRARY ieee; USE ieee.std_logic_1164.all; USE
ieee.std_logic_unsigned.all;

ENTITY ULPS_PathSearcher IS
    Port (  reset : IN std_logic;
        FrameSync : IN std_logic;
        clk : IN std_logic;
        I_channel : IN std_logic_vector(5 downto 0);
        Q_channel : IN std_logic_vector(5 downto 0);
        clk_ChipCodeGen : IN std_logic;
        ScCode_I : IN std_logic_vector(0 to 1);
        ScCode_Q : IN std_logic_vector(0 to 1);

        SL0_FrameSync : OUT std_logic;
        SL1_FrameSync : OUT std_logic
        );
END ULPS_PathSearcher ;

ARCHITECTURE behavior OF ULPS_PathSearcher IS
```

```
component Acc_ram
port (
    clk : IN std_logic;
    we  : IN std_logic;
    wraddress : IN std_logic_vector(1 downto 0);
    rdaddress : IN std_logic_vector(1 downto 0);
    di  : IN std_logic_vector(31 downto 0);
    do  : OUT std_logic_vector(31 downto 0)
);
end component;

component Ins_ram
PORT (
    clk : IN std_logic;
    we  : IN std_logic;
    wraddress : IN std_logic_vector(7 downto 0);
    rdaddress : IN std_logic_vector(7 downto 0);
    di  : IN std_logic_vector(34 downto 0);
    do  : OUT std_logic_vector(34 downto 0)
);
end component;

--//hide_st --//*************** Function Declaration ***************//
function MulRef (a:std_logic_vector; b:std_logic_vector(1 downto 0)) RETURN std_logic_vector IS
    variable tmp : std_logic_vector(6 downto 0);
begin
    tmp := a(5) & a(5 downto 0);
    case b is
        when "01" => return (tmp & '0');
        when "11" => return (("0000000"-tmp) & '0');
        when others => return "00000000";
    end case;
end MulRef;

function CorrExt (a:std_logic_vector) RETURN std_logic_vector IS
    variable tmp : std_logic_vector(15 downto 0);
begin
    tmp(7 downto 0):=a;
    tmp(15 downto 8):=(others=>tmp(7));
    return tmp;
end CorrExt;

function InsPowerExt (a:std_logic_vector) RETURN std_logic_vector IS
    variable tmp : std_logic_vector(7 downto 0);
begin
    tmp(4 downto 0):=a;
    tmp(7 downto 5):=(others=>'0');
    return tmp;
end InsPowerExt;

function sumMeanExt (a:std_logic_vector) RETURN std_logic_vector IS
    variable tmp : std_logic_vector(14 downto 0);
begin
    tmp(7 downto 0):=a;
    tmp(14 downto 8):=(others=>'0');
    return tmp;
end sumMeanExt;

function CorrRound (a:std_logic_vector) RETURN std_logic_vector IS
    constant carry : std_logic_vector(16 downto 0):= "00000000100000000";
    variable tmp : std_logic_vector(16 downto 0);
    variable tmp_abs : std_logic_vector(7 downto 0);
begin
    tmp := (a(15)&a) + carry;
```

```
    if (tmp(16)='1') then
        tmp_abs := "00000000"-tmp(16 downto 9);
    else
        tmp_abs := tmp(16 downto 9);
    end if;
    return tmp_abs(4 downto 0);
end CorrRound;

function Correlate_I (sig_I, sig_Q : std_logic_vector;
             Refsig_I, Refsig_Q : std_logic_vector;
             Acc_I : std_logic_vector) RETURN std_logic_vector IS
    variable tmpout_I : std_logic_vector(7 downto 0);
begin
    tmpout_I := MulRef(sig_I,Refsig_I)-MulRef(sig_Q,Refsig_Q);
    return (Acc_I + CorrExt(tmpout_I));
end Correlate_I;

function Correlate_Q (sig_I, sig_Q : std_logic_vector;
             Refsig_I, Refsig_Q : std_logic_vector;
             Acc_Q : std_logic_vector) RETURN std_logic_vector IS
    variable tmpout_Q : std_logic_vector(7 downto 0);
begin
    tmpout_Q := MulRef(sig_Q,Refsig_I)+MulRef(sig_I,Refsig_Q);
    return (Acc_Q + CorrExt(tmpout_Q));
end Correlate_Q;

function Min2 (a, b : std_logic_vector) RETURN std_logic IS
begin
    if (a<=b) then
        return '0';
    else
        return '1';
    end if;
end Min2;

function Cal_TH (a : std_logic_vector) RETURN std_logic_vector IS
    variable tmp : std_logic_vector(9 downto 0);
    variable round_a : std_logic_vector(14 downto 0);
begin
    round_a := a + "1000000";
    tmp := round_a(14 downto 7) & "00";
    return tmp(7 downto 0);
end Cal_TH;
--//hide_sp --//*****************************************************//

--//hide_st --//*************** Signal Declaration ***************//
--//----- Constant (Design parameter of Multipath Searcher) -----
constant tapsize : integer := 1536;        -- Correlator's tapsize
constant MaxDe_sample : integer := 159;    -- Maximum-DelaySpread(samples)
constant MaxDe_chip : integer := 39;       -- Maximum-DelaySpread(chips)
constant NumFinger : integer := 2;     -- The number of Rake's finger.

--//----- RAM's signal -----
signal InsRAM_clk : std_logic := '0';
signal InsRAM_we : std_logic := '0';                    -- assigned in 'P3 process'
signal InsRAM_wra : std_logic_vector(7 downto 0) :=(others=>'0');   -- assigned in 'P3 process'
     --InsRAM_rda is equivalent to "D_index" signal in 'P3 process'
signal InsRAM_di : std_logic_vector(34 downto 0) :=(others=>'0');   -- assigned in 'P3 process'
signal InsRAM_do : std_logic_Vector(34 downto 0) :=(others=>'0');   -- assigned in 'P3 process'

signal share_we : std_logic := '0';                -- assigned in concurrent
signal share_wra : std_logic_vector(1 downto 0) :=(others=>'0');    -- assigned in 'P2 process'
signal share_rda : std_logic_vector(1 downto 0) :=(others=>'0');    -- assigned in 'P2 process'
--//hide_st -- Cor#_di, Cor#_do [#:0-39]                   -- assigned in 'Cor_# process'
```

```
signal Cor0_di : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor1_di : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor2_di : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor3_di : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor4_di : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor5_di : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor6_di : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor7_di : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor8_di : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor9_di : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor10_di : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor11_di : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor12_di : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor13_di : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor14_di : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor15_di : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor16_di : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor17_di : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor18_di : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor19_di : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor20_di : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor21_di : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor22_di : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor23_di : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor24_di : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor25_di : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor26_di : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor27_di : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor28_di : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor29_di : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor30_di : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor31_di : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor32_di : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor33_di : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor34_di : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor35_di : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor36_di : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor37_di : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor38_di : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor39_di : std_logic_vector(31 downto 0):=(others=>'0');

signal Cor0_do : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor1_do : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor2_do : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor3_do : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor4_do : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor5_do : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor6_do : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor7_do : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor8_do : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor9_do : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor10_do : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor11_do : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor12_do : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor13_do : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor14_do : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor15_do : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor16_do : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor17_do : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor18_do : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor19_do : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor20_do : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor21_do : std_logic_vector(31 downto 0):=(others=>'0');
signal Cor22_do : std_logic_vector(31 downto 0):=(others=>'0');
```

```
    signal Cor23_do : std_logic_vector(31 downto 0):=(others=>'0');
    signal Cor24_do : std_logic_vector(31 downto 0):=(others=>'0');
    signal Cor25_do : std_logic_vector(31 downto 0):=(others=>'0');
    signal Cor26_do : std_logic_vector(31 downto 0):=(others=>'0');
    signal Cor27_do : std_logic_vector(31 downto 0):=(others=>'0');
    signal Cor28_do : std_logic_vector(31 downto 0):=(others=>'0');
    signal Cor29_do : std_logic_vector(31 downto 0):=(others=>'0');
    signal Cor30_do : std_logic_vector(31 downto 0):=(others=>'0');
    signal Cor31_do : std_logic_vector(31 downto 0):=(others=>'0');
    signal Cor32_do : std_logic_vector(31 downto 0):=(others=>'0');
    signal Cor33_do : std_logic_vector(31 downto 0):=(others=>'0');
    signal Cor34_do : std_logic_vector(31 downto 0):=(others=>'0');
    signal Cor35_do : std_logic_vector(31 downto 0):=(others=>'0');
    signal Cor36_do : std_logic_vector(31 downto 0):=(others=>'0');
    signal Cor37_do : std_logic_vector(31 downto 0):=(others=>'0');
    signal Cor38_do : std_logic_vector(31 downto 0):=(others=>'0');
    signal Cor39_do : std_logic_vector(31 downto 0):=(others=>'0');
    --//hide_sp

    --//----- Global signal -----
    signal ChipCounter : integer range 0 to (tapsize+MaxDe_chip) :=(tapsize+MaxDe_chip);
    signal SampleCounter : std_logic_vector(1 downto 0) := "00";

    --//----- Generate Reference_signal -----
    TYPE RefBufType IS ARRAY (0 to MaxDe_chip-1) OF std_logic_vector(1 downto 0);
    signal RefConj_I : std_logic_vector(1 downto 0) := "00";
    signal RefConj_Q : std_logic_vector(1 downto 0) := "00";
    signal RefBuf_I,RefBuf_Q : RefBufType := (others=>"00");

    --//----- p3 signal -----
    type state_type is (s1,s2,s3,s4,s5,s6,s7);
    signal state : state_type := s1;            -- default idle state.
    signal Correlator_en : std_logic := '0';        -- logic ['1' : enable],  ['0':disable]
    signal ClearAcc : std_logic := '0';         -- Accumulating register is cleared when logic '1'
    signal D_index : std_logic_vector(7 downto 0) :=(others=>'0');
    signal WrCol_pt : std_logic_vector(2 downto 0) := "101";

    -- Non-Coherent Averaging ---
    TYPE AvBufType IS ARRAY (0 to 4) OF std_logic_vector(7 downto 0);   -- Peak-Finding Window's tapsize
    signal AvBuf : AvBufType := (others=>"00000000");
    signal Acc_I,Acc_Q : std_logic_vector(15 downto 0) := (others=>'0');
    signal sumMean : std_logic_vector (14 downto 0) := (others=>'0');   -- [for mean-calculation]
    signal TH : std_logic_vector(7 downto 0) := (others=>'0');

    -- Local Peak Searching ---
    TYPE PeakIndexBufType IS ARRAY (0 to NumFinger-1) OF integer range 0 to MaxDe_sample;
    TYPE PeakPowerBufType IS ARRAY (0 to NumFinger-1) OF std_logic_vector(7 downto 0);
    signal PeakIndexBuf : PeakIndexBufType := (others=>0);
    signal PeakPowerBuf : PeakPowerBufType := (others=>"00000000");
    signal Finger_DelayIndex : PeakIndexBufType := (others=>0);

    --//----- p4-5 signal -----
    signal DelayCounter : integer range 0 to MaxDe_sample := MaxDe_sample;
    signal SL0_FrameSync_dummy : std_logic := '0';
    signal SL1_FrameSync_dummy : std_logic := '0';

    --//hide_sp --//************************************************//


BEGIN
    -- p1 : Sample Counter process.
    -- p2 : AccRAM's shared signal assignment process.
    -- p3 : State Machine process.
    -- p4-5 : Generate slave FrameSync signal.
```

```
-- Cor_# : individual signal assignment of each correlator.

--//hide_st --//*************** Concurrent Statements ***************//

--//----- Generate reference code of correlator -----
RefConj_I <= not(ScCode_I(0))&'1' when (ChipCounter<1280)AND(ScCode_I(1)='1') else
         ScCode_I         when (ChipCounter<tapsize)AND(ScCode_I(1)='1') else
         "00";

RefConj_Q <= ScCode_Q         when (ChipCounter<1280)AND(ScCode_Q(1)='1') else
         not(ScCode_Q(0))&'1' when (ChipCounter<tapsize)AND(ScCode_Q(1)='1') else
         "00";

--//----- RAM's instantiation : AccRAM & InsRAM -----
share_we <= Correlator_en OR ClearAcc;
--//hide_st -- Cor#_inst : PORT MAP
Cor0_inst :  Acc_ram PORT MAP (clk,share_we,share_wra,share_rda,Cor0_di,Cor0_do);
Cor1_inst :  Acc_ram PORT MAP (clk,share_we,share_wra,share_rda,Cor1_di,Cor1_do);
Cor2_inst :  Acc_ram PORT MAP (clk,share_we,share_wra,share_rda,Cor2_di,Cor2_do);
Cor3_inst :  Acc_ram PORT MAP (clk,share_we,share_wra,share_rda,Cor3_di,Cor3_do);
Cor4_inst :  Acc_ram PORT MAP (clk,share_we,share_wra,share_rda,Cor4_di,Cor4_do);
Cor5_inst :  Acc_ram PORT MAP (clk,share_we,share_wra,share_rda,Cor5_di,Cor5_do);
Cor6_inst :  Acc_ram PORT MAP (clk,share_we,share_wra,share_rda,Cor6_di,Cor6_do);
Cor7_inst :  Acc_ram PORT MAP (clk,share_we,share_wra,share_rda,Cor7_di,Cor7_do);
Cor8_inst :  Acc_ram PORT MAP (clk,share_we,share_wra,share_rda,Cor8_di,Cor8_do);
Cor9_inst :  Acc_ram PORT MAP (clk,share_we,share_wra,share_rda,Cor9_di,Cor9_do);
Cor10_inst : Acc_ram PORT MAP (clk,share_we,share_wra,share_rda,Cor10_di,Cor10_do);
Cor11_inst : Acc_ram PORT MAP (clk,share_we,share_wra,share_rda,Cor11_di,Cor11_do);
Cor12_inst : Acc_ram PORT MAP (clk,share_we,share_wra,share_rda,Cor12_di,Cor12_do);
Cor13_inst : Acc_ram PORT MAP (clk,share_we,share_wra,share_rda,Cor13_di,Cor13_do);
Cor14_inst : Acc_ram PORT MAP (clk,share_we,share_wra,share_rda,Cor14_di,Cor14_do);
Cor15_inst : Acc_ram PORT MAP (clk,share_we,share_wra,share_rda,Cor15_di,Cor15_do);
Cor16_inst : Acc_ram PORT MAP (clk,share_we,share_wra,share_rda,Cor16_di,Cor16_do);
Cor17_inst : Acc_ram PORT MAP (clk,share_we,share_wra,share_rda,Cor17_di,Cor17_do);
Cor18_inst : Acc_ram PORT MAP (clk,share_we,share_wra,share_rda,Cor18_di,Cor18_do);
Cor19_inst : Acc_ram PORT MAP (clk,share_we,share_wra,share_rda,Cor19_di,Cor19_do);
Cor20_inst : Acc_ram PORT MAP (clk,share_we,share_wra,share_rda,Cor20_di,Cor20_do);
Cor21_inst : Acc_ram PORT MAP (clk,share_we,share_wra,share_rda,Cor21_di,Cor21_do);
Cor22_inst : Acc_ram PORT MAP (clk,share_we,share_wra,share_rda,Cor22_di,Cor22_do);
Cor23_inst : Acc_ram PORT MAP (clk,share_we,share_wra,share_rda,Cor23_di,Cor23_do);
Cor24_inst : Acc_ram PORT MAP (clk,share_we,share_wra,share_rda,Cor24_di,Cor24_do);
Cor25_inst : Acc_ram PORT MAP (clk,share_we,share_wra,share_rda,Cor25_di,Cor25_do);
Cor26_inst : Acc_ram PORT MAP (clk,share_we,share_wra,share_rda,Cor26_di,Cor26_do);
Cor27_inst : Acc_ram PORT MAP (clk,share_we,share_wra,share_rda,Cor27_di,Cor27_do);
Cor28_inst : Acc_ram PORT MAP (clk,share_we,share_wra,share_rda,Cor28_di,Cor28_do);
Cor29_inst : Acc_ram PORT MAP (clk,share_we,share_wra,share_rda,Cor29_di,Cor29_do);
Cor30_inst : Acc_ram PORT MAP (clk,share_we,share_wra,share_rda,Cor30_di,Cor30_do);
Cor31_inst : Acc_ram PORT MAP (clk,share_we,share_wra,share_rda,Cor31_di,Cor31_do);
Cor32_inst : Acc_ram PORT MAP (clk,share_we,share_wra,share_rda,Cor32_di,Cor32_do);
Cor33_inst : Acc_ram PORT MAP (clk,share_we,share_wra,share_rda,Cor33_di,Cor33_do);
Cor34_inst : Acc_ram PORT MAP (clk,share_we,share_wra,share_rda,Cor34_di,Cor34_do);
Cor35_inst : Acc_ram PORT MAP (clk,share_we,share_wra,share_rda,Cor35_di,Cor35_do);
Cor36_inst : Acc_ram PORT MAP (clk,share_we,share_wra,share_rda,Cor36_di,Cor36_do);
Cor37_inst : Acc_ram PORT MAP (clk,share_we,share_wra,share_rda,Cor37_di,Cor37_do);
Cor38_inst : Acc_ram PORT MAP (clk,share_we,share_wra,share_rda,Cor38_di,Cor38_do);
Cor39_inst : Acc_ram PORT MAP (clk,share_we,share_wra,share_rda,Cor39_di,Cor39_do);
--//hide_sp

InsRAM_clk <= NOT(clk_ChipCodeGen);
InsBuf_ram : Ins_ram PORT MAP (InsRAM_clk,InsRAM_we,InsRAM_wra,D_index,InsRAM_di,InsRAM_do);

--//hide_st --//-----
Demux 80:1 for Acc_I&Q from AccRAM in order to Calculate Instantaneous Power -----
with CONV_INTEGER(D_index) select
```

```
    Acc_I <= Cor0_do(31 downto 16)     when  0   to 3,
         Cor1_do(31 downto 16)     when  4   to 7,
         Cor2_do(31 downto 16)     when  8   to 11,
         Cor3_do(31 downto 16)     when  12  to 15,
         Cor4_do(31 downto 16)     when  16  to 19,
         Cor5_do(31 downto 16)     when  20  to 23,
         Cor6_do(31 downto 16)     when  24  to 27,
         Cor7_do(31 downto 16)     when  28  to 31,
         Cor8_do(31 downto 16)     when  32  to 35,
         Cor9_do(31 downto 16)     when  36  to 39,
         Cor10_do(31 downto 16)     when  40 to 43,
         Cor11_do(31 downto 16)     when  44 to 47,
         Cor12_do(31 downto 16)     when  48 to 51,
         Cor13_do(31 downto 16)     when  52 to 55,
         Cor14_do(31 downto 16)     when  56 to 59,
         Cor15_do(31 downto 16)     when  60 to 63,
         Cor16_do(31 downto 16)     when  64 to 67,
         Cor17_do(31 downto 16)     when  68 to 71,
         Cor18_do(31 downto 16)     when  72 to 75,
         Cor19_do(31 downto 16)     when  76 to 79,
         Cor20_do(31 downto 16)     when  80 to 83,
         Cor21_do(31 downto 16)     when  84 to 87,
         Cor22_do(31 downto 16)     when  88 to 91,
         Cor23_do(31 downto 16)     when  92 to 95,
         Cor24_do(31 downto 16)     when  96 to 99,
         Cor25_do(31 downto 16)     when  100 to 103,
         Cor26_do(31 downto 16)     when  104 to 107,
         Cor27_do(31 downto 16)     when  108 to 111,
         Cor28_do(31 downto 16)     when  112 to 115,
         Cor29_do(31 downto 16)     when  116 to 119,
         Cor30_do(31 downto 16)     when  120 to 123,
         Cor31_do(31 downto 16)     when  124 to 127,
         Cor32_do(31 downto 16)     when  128 to 131,
         Cor33_do(31 downto 16)     when  132 to 135,
         Cor34_do(31 downto 16)     when  136 to 139,
         Cor35_do(31 downto 16)     when  140 to 143,
         Cor36_do(31 downto 16)     when  144 to 147,
         Cor37_do(31 downto 16)     when  148 to 151,
         Cor38_do(31 downto 16)     when  152 to 155,
         Cor39_do(31 downto 16)     when  156 to 159,
         (others=>'0')          when others;


with CONV_INTEGER(D_index) select
    Acc_Q <= Cor0_do(15 downto 0)     when  0   to 3,
         Cor1_do(15 downto 0)     when  4   to 7,
         Cor2_do(15 downto 0)     when  8   to 11,
         Cor3_do(15 downto 0)     when  12  to 15,
         Cor4_do(15 downto 0)     when  16  to 19,
         Cor5_do(15 downto 0)     when  20  to 23,
         Cor6_do(15 downto 0)     when  24  to 27,
         Cor7_do(15 downto 0)     when  28  to 31,
         Cor8_do(15 downto 0)     when  32  to 35,
         Cor9_do(15 downto 0)     when  36  to 39,
         Cor10_do(15 downto 0)     when  40 to 43,
         Cor11_do(15 downto 0)     when  44 to 47,
         Cor12_do(15 downto 0)     when  48 to 51,
         Cor13_do(15 downto 0)     when  52 to 55,
         Cor14_do(15 downto 0)     when  56 to 59,
         Cor15_do(15 downto 0)     when  60 to 63,
         Cor16_do(15 downto 0)     when  64 to 67,
         Cor17_do(15 downto 0)     when  68 to 71,
         Cor18_do(15 downto 0)     when  72 to 75,
         Cor19_do(15 downto 0)     when  76 to 79,
```

```
         Cor20_do(15 downto 0)     when  80 to 83,
         Cor21_do(15 downto 0)     when  84 to 87,
         Cor22_do(15 downto 0)     when  88 to 91,
         Cor23_do(15 downto 0)     when  92 to 95,
         Cor24_do(15 downto 0)     when  96 to 99,
         Cor25_do(15 downto 0)     when  100 to 103,
         Cor26_do(15 downto 0)     when  104 to 107,
         Cor27_do(15 downto 0)     when  108 to 111,
         Cor28_do(15 downto 0)     when  112 to 115,
         Cor29_do(15 downto 0)     when  116 to 119,
         Cor30_do(15 downto 0)     when  120 to 123,
         Cor31_do(15 downto 0)     when  124 to 127,
         Cor32_do(15 downto 0)     when  128 to 131,
         Cor33_do(15 downto 0)     when  132 to 135,
         Cor34_do(15 downto 0)     when  136 to 139,
         Cor35_do(15 downto 0)     when  140 to 143,
         Cor36_do(15 downto 0)     when  144 to 147,
         Cor37_do(15 downto 0)     when  148 to 151,
         Cor38_do(15 downto 0)     when  152 to 155,
         Cor39_do(15 downto 0)     when  156 to 159,
         (others=>'0')          when others;
--//hide_sp

--//----- Dummy Slave FramSync signal : Output of p4-5 -----
SL0_FrameSync <= SL0_FrameSync_dummy;
SL1_FrameSync <= SL1_FrameSync_dummy;

--//hide_sp --//************************************************//

--//hide_st --//*************** Sequential Statements ***************//
p1: PROCESS (clk)
BEGIN
    IF (clk'event AND clk='1') THEN
        if (FrameSync='1') then
            SampleCounter <= "00";
        else
            SampleCounter <= SampleCounter + '1';
        end if;
    END IF;
END PROCESS p1;


p2: PROCESS (clk)
BEGIN
    IF (clk'event AND clk='0') THEN
        if (state=s2) then                  -- [R/W]
            case SampleCounter is
            when "00" =>
                share_wra <= "00";
                share_rda <= "01";
            when "01" =>
                share_wra <= "01";
                share_rda <= "10";
            when "10" =>
                share_wra <= "10";
                share_rda <= "11";
            when others =>
                share_wra <= "11";
                share_rda <= "00";
                RefBuf_I <= RefConj_I & RefBuf_I(0 to MaxDe_chip-2);
                RefBuf_Q <= RefConj_Q & RefBuf_Q(0 to MaxDe_chip-2);
            end case;
        elsif ((state=s3)AND(SampleCounter="10")) then      -- [R]
            if (D_index = "000000000") then
```

```
                share_rda <= "00";
            else
                share_rda <= share_rda + '1';
            end if;
        elsif (state=s1) then                    -- [W]
            share_wra <= SampleCounter;
            RefBuf_I <= (others=>"00");
            RefBuf_Q <= (others=>"00");
        end if;
    END IF;
END PROCESS p2;

p3: PROCESS (clk_ChipCodeGen,reset)
variable InsPower : std_logic_vector (4 downto 0) := (others=>'0');
variable PeakIndex : integer range 0 to MaxDe_sample:=0;
variable PeakPower : std_logic_vector(7 downto 0):= "00000000";
variable DiffIndex0 : integer range 0 to MaxDe_sample := 0;
variable DiffIndex1 : integer range 0 to MaxDe_sample := 0;
variable MinIndex : std_logic := '0';
variable path_index : integer range 0 to NumFinger-1 := 0;
variable Old_delayindex : PeakIndexBufType :=(others=>0);
variable InsClear_index : std_logic_vector(7 downto 0) := (others=>'0');


BEGIN
    if (reset='1') then
        ChipCounter <= (tapsize+MaxDe_chip);
        Correlator_en <= '0';
        ClearAcc <= '1';
        InsClear_index := (others=>'0');
        WrCol_pt <= "110";
        AvBuf <= (others=>"00000000");
        sumMean <= (others=>'0');
        PeakIndexBuf <= (others=>0);
        PeakPowerBuf <= (others=>"00000000");
        Finger_DelayIndex <= (others=>0);
        PeakIndex := 0;
        PeakPower := "00000000";
        MinIndex := '0';
        state <= s7;
    elsif (clk_ChipCodeGen'event AND clk_ChipCodeGen='1') then

        case state is
          when s1 =>  --[idle state]
            if (FrameSync='1') then
                ChipCounter <= 0;
                Correlator_en <= '1';
                ClearAcc <= '0';
                state <= s2;
            else
                state <= s1;
            end if;

          when s2 =>  --[correlation state]
            if (ChipCounter<(tapsize+MaxDe_chip)) then
                ChipCounter <= ChipCounter + 1;
            else
                Correlator_en <= '0';
                D_index <= (others=>'0');
                if (WrCol_pt="110") then
                    WrCol_pt <= "000";
                else
                    WrCol_pt <= WrCol_pt + '1';
                end if;
```

```
        state <= s3;
    end if;

when s3 =>  --[power delay profile state]
    if (D_index < "10011111") then
        InsRAM_we <= '1';
        InsRAM_wra <= D_index;
        D_index <= D_index + '1';
        --//finding (local peak) in 5-tap peak-finding Window.
        if ((AvBuf(1)>AvBuf(0))AND(AvBuf(1)>AvBuf(2))) then
            PeakIndex := CONV_INTEGER(D_index-"10");
            PeakPower := AvBuf(1);
        elsif ((AvBuf(1)=AvBuf(2))AND(AvBuf(1)>AvBuf(0))AND(AvBuf(1)>AvBuf(3))) then
            if (AvBuf(0)>AvBuf(3)) then
                PeakIndex := CONV_INTEGER(D_index-"10");
                PeakPower := AvBuf(1);
            else
                PeakIndex := CONV_INTEGER(D_index-"11");
                PeakPower := AvBuf(2);
            end if;
        elsif ((AvBuf(1)=AvBuf(2))AND(AvBuf(1)=AvBuf(3))AND
        (AvBuf(1)>AvBuf(0))AND(AvBuf(1)>AvBuf(4))) then
            PeakIndex := CONV_INTEGER(D_index-"11");
            PeakPower := AvBuf(2);
        end if;
        -- Compare(local peak with PeakPowerBuf) for finding (#Finger) strongest peaks
        MinIndex := Min2(PeakPowerBuf(0),PeakPowerBuf(1));
        DiffIndex0 := abs(PeakIndex-PeakIndexBuf(0));
        DiffIndex1 := abs(PeakIndex-PeakIndexBuf(1));
        CASE MinIndex IS
                WHEN '0' =>
                    if ((PeakPower>PeakPowerBuf(0))and(DiffIndex1>3)) then
                        PeakIndexBuf(0) <= PeakIndex;
                    PeakPowerBuf(0) <= PeakPower;
                    end if;
                WHEN others =>
                    if ((PeakPower>PeakPowerBuf(1))and(DiffIndex0>3)) then
                        PeakIndexBuf(1) <= PeakIndex;
                    PeakPowerBuf(1) <= PeakPower;
                    end if;
        END CASE ;
            --//Shift 5-tap peak-finding Window.
        AvBuf(1 to 4) <= AvBuf(0 to 3);
        --Calculate AvPower for 0th-tap of window
           --//Calculate Instantaneous PowerDelay_profile
        InsPower := CorrRound(Acc_I) + CorrRound(Acc_Q);
        --//Cauculate Averageous PowerDelay_profile
        AvBuf(0) <= InsPowerExt(InsPower) + InsPowerExt(InsRAM_do(34 downto 30)) +
        InsPowerExt(InsRAM_do(29 downto 25)) + InsPowerExt(InsRAM_do(24 downto 20)) +
        InsPowerExt(InsRAM_do(19 downto 15)) + InsPowerExt(InsRAM_do(14 downto 10)) +
        InsPowerExt(InsRAM_do(9 downto 5)) + InsPowerExt(InsRAM_do(4 downto 0));
        sumMean <= sumMean + sumMeanExt(AvBuf(0));
        --//Shift InsBuf register...
        case WrCol_pt is
            when "000" => InsRAM_di
            <= InsRAM_do(34 downto 5) & InsPower;
            when "001" => InsRAM_di
            <= InsRAM_do(34 downto 10) & InsPower & InsRAM_do(4 downto 0);
            when "010" => InsRAM_di
            <= InsRAM_do(34 downto 15) & InsPower & InsRAM_do(9 downto 0);
            when "011" => InsRAM_di
            <= InsRAM_do(34 downto 20) & InsPower & InsRAM_do(14 downto 0);
            when "100" => InsRAM_di
            <= InsRAM_do(34 downto 25) & InsPower & InsRAM_do(19 downto 0);
```

```
            when "101" => InsRAM_di
            <= InsRAM_do(34 downto 30) & InsPower & InsRAM_do(24 downto 0);
            when "110" => InsRAM_di
            <= InsPower & InsRAM_do(29 downto 0);
            when others => InsRAM_di <= (others=>'0');
        end case;

    else
        insRAM_we <= '0';
        TH <= Cal_TH(sumMean);
        state <= s4;
    end if;

when s4 =>  --[finger assignment state(sub_0): compare with TH]
    Old_delayindex := Finger_DelayIndex;
    Finger_DelayIndex <= (others=>0);
        if (PeakPowerBuf(0)<TH) then
            PeakIndexBuf(0) <= 0;
        end if;

        if (PeakPowerBuf(1)<TH) then
            PeakIndexBuf(1) <= 0;
        end if;
    path_index := 0;
    state <= s5;

when s5 =>  --[finger assignment state(sub_1): assign old delay time]
    if (PeakIndexBuf(path_index)/=0) then
        if (PeakIndexBuf(path_index)=Old_delayindex(0)) then
            Finger_DelayIndex(0) <= PeakIndexBuf(path_index);
            PeakIndexBuf(path_index) <= 0;
        elsif (PeakIndexBuf(path_index)=Old_delayindex(1)) then
            Finger_DelayIndex(1) <= PeakIndexBuf(path_index);
            PeakIndexBuf(path_index) <= 0;
        end if;
    end if;
    -------------------------
    if (path_index=(NumFinger-1)) then
            path_index := 0;
            state <= s6;
        else
            path_index := path_index+1;
        end if;

when s6 =>  --[finger assignment state(sub_2): assign new dalay time]
        if (PeakIndexBuf(path_index)/=0) then
            if (Finger_DelayIndex(0)=0) then
                Finger_DelayIndex(0) <= PeakIndexBuf(path_index);
            elsif (Finger_DelayIndex(1)=0) then
                Finger_DelayIndex(1) <= PeakIndexBuf(path_index);
            end if;
        end if;
        -------------------------
        if (path_index=(NumFinger-1)) then
            --clear buffer
            ClearAcc <= '1';
            AvBuf <= (others=>"00000000");
            sumMean <= (others=>'0');
            PeakIndexBuf <= (others=>0);
        PeakPowerBuf <= (others=>"00000000");
        PeakIndex := 0;
        PeakPower := "00000000";
        MinIndex := '0';
        state <= s1;
```

```
                else
                    path_index := path_index+1;
                end if;

          when s7 =>  --[Reset state for clearing InsRAM about 320 clk_Chip]
            if (InsClear_index <= "10011111") then
                InsRAM_we <= '1';
                InsRAM_wra <= InsClear_index;
                InsRAM_di <= (others=>'0');
                InsClear_index := InsClear_index + '1';
            else
                InsRAM_we <= '0';
                state <= s1;
            end if;
        end case;
    end if;
END PROCESS p3;


p4: PROCESS (clk)
BEGIN
    IF (clk'event AND clk='1') THEN
        if (FrameSync='1') then
            DelayCounter <= 1;
        elsif (DelayCounter<MaxDe_sample) then
            DelayCounter <= DelayCounter + 1;
        end if;
    END IF;
END PROCESS p4;


p5: PROCESS (clk)
BEGIN
    IF (clk'event AND clk='0') THEN
        if (DelayCounter=Finger_DelayIndex(0)) then
            SL0_FrameSync_dummy <= '1';
        else
            SL0_FrameSync_dummy <= '0';
        end if;

        if (DelayCounter=Finger_DelayIndex(1)) then
            SL1_FrameSync_dummy <= '1';
        else
            SL1_FrameSync_dummy <= '0';
        end if;
    END IF;
END PROCESS p5;


--//hide_st  --////// -- Correlator Banks -- ///////
Cor_0: PROCESS (clk)
BEGIN
    IF (clk'event AND clk='0') THEN
        if (state=s2) then
            Cor0_di(31 downto 16) <=
            Correlate_I(I_channel,Q_channel,RefConj_I,RefConj_Q,Cor0_do(31 downto 16));
            Cor0_di(15 downto 0)  <=
            Correlate_Q(I_channel,Q_channel,RefConj_I,RefConj_Q,Cor0_do(15 downto 0));
        elsif (state=s1) then
            Cor0_di <= (others=>'0');
        end if;
    END IF;
END PROCESS Cor_0;
```

```
Cor_1: PROCESS (clk)
BEGIN
    IF (clk'event AND clk='0') THEN
        if (state=s2) then
            Cor1_di(31 downto 16) <=
            Correlate_I(I_channel,Q_channel,RefBuf_I(0),RefBuf_Q(0),Cor1_do(31 downto 16));
            Cor1_di(15 downto 0)  <=
            Correlate_Q(I_channel,Q_channel,RefBuf_I(0),RefBuf_Q(0),Cor1_do(15 downto 0));
        elsif (state=s1) then
            Cor1_di <= (others=>'0');
        end if;
    END IF;
END PROCESS Cor_1;

Cor_2: PROCESS (clk)
BEGIN
    IF (clk'event AND clk='0') THEN
        if (state=s2) then
            Cor2_di(31 downto 16) <=
            Correlate_I(I_channel,Q_channel,RefBuf_I(1),RefBuf_Q(1),Cor2_do(31 downto 16));
            Cor2_di(15 downto 0)  <=
            Correlate_Q(I_channel,Q_channel,RefBuf_I(1),RefBuf_Q(1),Cor2_do(15 downto 0));
        elsif (state=s1) then
            Cor2_di <= (others=>'0');
        end if;
    END IF;
END PROCESS Cor_2;

Cor_3: PROCESS (clk)
BEGIN
    IF (clk'event AND clk='0') THEN
        if (state=s2) then
            Cor3_di(31 downto 16) <=
            Correlate_I(I_channel,Q_channel,RefBuf_I(2),RefBuf_Q(2),Cor3_do(31 downto 16));
            Cor3_di(15 downto 0)  <=
            Correlate_Q(I_channel,Q_channel,RefBuf_I(2),RefBuf_Q(2),Cor3_do(15 downto 0));
        elsif (state=s1) then
            Cor3_di <= (others=>'0');
        end if;
    END IF;
END PROCESS Cor_3;

Cor_4: PROCESS (clk)
BEGIN
    IF (clk'event AND clk='0') THEN
        if (state=s2) then
            Cor4_di(31 downto 16) <=
            Correlate_I(I_channel,Q_channel,RefBuf_I(3),RefBuf_Q(3),Cor4_do(31 downto 16));
            Cor4_di(15 downto 0)  <=
            Correlate_Q(I_channel,Q_channel,RefBuf_I(3),RefBuf_Q(3),Cor4_do(15 downto 0));
        elsif (state=s1) then
            Cor4_di <= (others=>'0');
        end if;
    END IF;
END PROCESS Cor_4;

Cor_5: PROCESS (clk)
BEGIN
    IF (clk'event AND clk='0') THEN
        if (state=s2) then
            Cor5_di(31 downto 16) <=
            Correlate_I(I_channel,Q_channel,RefBuf_I(4),RefBuf_Q(4),Cor5_do(31 downto 16));
            Cor5_di(15 downto 0)  <=
            Correlate_Q(I_channel,Q_channel,RefBuf_I(4),RefBuf_Q(4),Cor5_do(15 downto 0));
```

```
        elsif (state=s1) then
            Cor5_di <= (others=>'0');
        end if;
    END IF;
END PROCESS Cor_5;

Cor_6: PROCESS (clk)
BEGIN
    IF (clk'event AND clk='0') THEN
        if (state=s2) then
            Cor6_di(31 downto 16) <=
            Correlate_I(I_channel,Q_channel,RefBuf_I(5),RefBuf_Q(5),Cor6_do(31 downto 16));
            Cor6_di(15 downto 0)  <=
            Correlate_Q(I_channel,Q_channel,RefBuf_I(5),RefBuf_Q(5),Cor6_do(15 downto 0));
        elsif (state=s1) then
            Cor6_di <= (others=>'0');
        end if;
    END IF;
END PROCESS Cor_6;

Cor_7: PROCESS (clk)
BEGIN
    IF (clk'event AND clk='0') THEN
        if (state=s2) then
            Cor7_di(31 downto 16) <=
            Correlate_I(I_channel,Q_channel,RefBuf_I(6),RefBuf_Q(6),Cor7_do(31 downto 16));
            Cor7_di(15 downto 0)  <=
            Correlate_Q(I_channel,Q_channel,RefBuf_I(6),RefBuf_Q(6),Cor7_do(15 downto 0));
        elsif (state=s1) then
            Cor7_di <= (others=>'0');
        end if;
    END IF;
END PROCESS Cor_7;

Cor_8: PROCESS (clk)
BEGIN
    IF (clk'event AND clk='0') THEN
        if (state=s2) then
            Cor8_di(31 downto 16) <=
            Correlate_I(I_channel,Q_channel,RefBuf_I(7),RefBuf_Q(7),Cor8_do(31 downto 16));
            Cor8_di(15 downto 0)  <=
            Correlate_Q(I_channel,Q_channel,RefBuf_I(7),RefBuf_Q(7),Cor8_do(15 downto 0));
        elsif (state=s1) then
            Cor8_di <= (others=>'0');
        end if;
    END IF;
END PROCESS Cor_8;

Cor_9: PROCESS (clk)
BEGIN
    IF (clk'event AND clk='0') THEN
        if (state=s2) then
            Cor9_di(31 downto 16) <=
            Correlate_I(I_channel,Q_channel,RefBuf_I(8),RefBuf_Q(8),Cor9_do(31 downto 16));
            Cor9_di(15 downto 0)  <=
            Correlate_Q(I_channel,Q_channel,RefBuf_I(8),RefBuf_Q(8),Cor9_do(15 downto 0));
        elsif (state=s1) then
            Cor9_di <= (others=>'0');
        end if;
    END IF;
END PROCESS Cor_9;

Cor_10: PROCESS (clk)
BEGIN
```

```
    IF (clk'event AND clk='0') THEN
        if (state=s2) then
            Cor10_di(31 downto 16) <=
            Correlate_I(I_channel,Q_channel,RefBuf_I(9),RefBuf_Q(9),Cor10_do(31 downto 16));
            Cor10_di(15 downto 0)  <=
            Correlate_Q(I_channel,Q_channel,RefBuf_I(9),RefBuf_Q(9),Cor10_do(15 downto 0));
        elsif (state=s1) then
            Cor10_di <= (others=>'0');
        end if;
    END IF;
END PROCESS Cor_10;

Cor_11: PROCESS (clk)
BEGIN
    IF (clk'event AND clk='0') THEN
        if (state=s2) then
            Cor11_di(31 downto 16) <=
            Correlate_I(I_channel,Q_channel,RefBuf_I(10),RefBuf_Q(10),Cor11_do(31 downto 16));
            Cor11_di(15 downto 0)  <=
            Correlate_Q(I_channel,Q_channel,RefBuf_I(10),RefBuf_Q(10),Cor11_do(15 downto 0));
        elsif (state=s1) then
            Cor11_di <= (others=>'0');
        end if;
    END IF;
END PROCESS Cor_11;

Cor_12: PROCESS (clk)
BEGIN
    IF (clk'event AND clk='0') THEN
        if (state=s2) then
            Cor12_di(31 downto 16) <=
            Correlate_I(I_channel,Q_channel,RefBuf_I(11),RefBuf_Q(11),Cor12_do(31 downto 16));
            Cor12_di(15 downto 0)  <=
            Correlate_Q(I_channel,Q_channel,RefBuf_I(11),RefBuf_Q(11),Cor12_do(15 downto 0));
        elsif (state=s1) then
            Cor12_di <= (others=>'0');
        end if;
    END IF;
END PROCESS Cor_12;

Cor_13: PROCESS (clk)
BEGIN
    IF (clk'event AND clk='0') THEN
        if (state=s2) then
            Cor13_di(31 downto 16) <=
            Correlate_I(I_channel,Q_channel,RefBuf_I(12),RefBuf_Q(12),Cor13_do(31 downto 16));
            Cor13_di(15 downto 0)  <=
            Correlate_Q(I_channel,Q_channel,RefBuf_I(12),RefBuf_Q(12),Cor13_do(15 downto 0));
        elsif (state=s1) then
            Cor13_di <= (others=>'0');
        end if;
    END IF;
END PROCESS Cor_13;

Cor_14: PROCESS (clk)
BEGIN
    IF (clk'event AND clk='0') THEN
        if (state=s2) then
            Cor14_di(31 downto 16) <=
            Correlate_I(I_channel,Q_channel,RefBuf_I(13),RefBuf_Q(13),Cor14_do(31 downto 16));
            Cor14_di(15 downto 0)  <=
            Correlate_Q(I_channel,Q_channel,RefBuf_I(13),RefBuf_Q(13),Cor14_do(15 downto 0));
        elsif (state=s1) then
            Cor14_di <= (others=>'0');
```

```
        end if;
    END IF;
END PROCESS Cor_14;

Cor_15: PROCESS (clk)
BEGIN
    IF (clk'event AND clk='0') THEN
        if (state=s2) then
            Cor15_di(31 downto 16) <=
            Correlate_I(I_channel,Q_channel,RefBuf_I(14),RefBuf_Q(14),Cor15_do(31 downto 16));
            Cor15_di(15 downto 0)  <=
            Correlate_Q(I_channel,Q_channel,RefBuf_I(14),RefBuf_Q(14),Cor15_do(15 downto 0));
        elsif (state=s1) then
            Cor15_di <= (others=>'0');
        end if;
    END IF;
END PROCESS Cor_15;

Cor_16: PROCESS (clk)
BEGIN
    IF (clk'event AND clk='0') THEN
        if (state=s2) then
            Cor16_di(31 downto 16) <=
            Correlate_I(I_channel,Q_channel,RefBuf_I(15),RefBuf_Q(15),Cor16_do(31 downto 16));
            Cor16_di(15 downto 0)  <=
            Correlate_Q(I_channel,Q_channel,RefBuf_I(15),RefBuf_Q(15),Cor16_do(15 downto 0));
        elsif (state=s1) then
            Cor16_di <= (others=>'0');
        end if;
    END IF;
END PROCESS Cor_16;

Cor_17: PROCESS (clk)
BEGIN
    IF (clk'event AND clk='0') THEN
        if (state=s2) then
            Cor17_di(31 downto 16) <=
            Correlate_I(I_channel,Q_channel,RefBuf_I(16),RefBuf_Q(16),Cor17_do(31 downto 16));
            Cor17_di(15 downto 0)  <=
            Correlate_Q(I_channel,Q_channel,RefBuf_I(16),RefBuf_Q(16),Cor17_do(15 downto 0));
        elsif (state=s1) then
            Cor17_di <= (others=>'0');
        end if;
    END IF;
END PROCESS Cor_17;

Cor_18: PROCESS (clk)
BEGIN
    IF (clk'event AND clk='0') THEN
        if (state=s2) then
            Cor18_di(31 downto 16) <=
            Correlate_I(I_channel,Q_channel,RefBuf_I(17),RefBuf_Q(17),Cor18_do(31 downto 16));
            Cor18_di(15 downto 0)  <=
            Correlate_Q(I_channel,Q_channel,RefBuf_I(17),RefBuf_Q(17),Cor18_do(15 downto 0));
        elsif (state=s1) then
            Cor18_di <= (others=>'0');
        end if;
    END IF;
END PROCESS Cor_18;

Cor_19: PROCESS (clk)
BEGIN
    IF (clk'event AND clk='0') THEN
        if (state=s2) then
```

```
            Cor19_di(31 downto 16) <=
            Correlate_I(I_channel,Q_channel,RefBuf_I(18),RefBuf_Q(18),Cor19_do(31 downto 16));
            Cor19_di(15 downto 0)  <=
            Correlate_Q(I_channel,Q_channel,RefBuf_I(18),RefBuf_Q(18),Cor19_do(15 downto 0));
        elsif (state=s1) then
            Cor19_di <= (others=>'0');
        end if;
    END IF;
END PROCESS Cor_19;

Cor_20: PROCESS (clk)
BEGIN
    IF (clk'event AND clk='0') THEN
        if (state=s2) then
            Cor20_di(31 downto 16) <=
            Correlate_I(I_channel,Q_channel,RefBuf_I(19),RefBuf_Q(19),Cor20_do(31 downto 16));
            Cor20_di(15 downto 0)  <=
            Correlate_Q(I_channel,Q_channel,RefBuf_I(19),RefBuf_Q(19),Cor20_do(15 downto 0));
        elsif (state=s1) then
            Cor20_di <= (others=>'0');
        end if;
    END IF;
END PROCESS Cor_20;

Cor_21: PROCESS (clk)
BEGIN
    IF (clk'event AND clk='0') THEN
        if (state=s2) then
            Cor21_di(31 downto 16) <=
            Correlate_I(I_channel,Q_channel,RefBuf_I(20),RefBuf_Q(20),Cor21_do(31 downto 16));
            Cor21_di(15 downto 0)  <=
            Correlate_Q(I_channel,Q_channel,RefBuf_I(20),RefBuf_Q(20),Cor21_do(15 downto 0));
        elsif (state=s1) then
            Cor21_di <= (others=>'0');
        end if;
    END IF;
END PROCESS Cor_21;

Cor_22: PROCESS (clk)
BEGIN
    IF (clk'event AND clk='0') THEN
        if (state=s2) then
            Cor22_di(31 downto 16) <=
            Correlate_I(I_channel,Q_channel,RefBuf_I(21),RefBuf_Q(21),Cor22_do(31 downto 16));
            Cor22_di(15 downto 0)  <=
            Correlate_Q(I_channel,Q_channel,RefBuf_I(21),RefBuf_Q(21),Cor22_do(15 downto 0));
        elsif (state=s1) then
            Cor22_di <= (others=>'0');
        end if;
    END IF;
END PROCESS Cor_22;

Cor_23: PROCESS (clk)
BEGIN
    IF (clk'event AND clk='0') THEN
        if (state=s2) then
            Cor23_di(31 downto 16) <=
            Correlate_I(I_channel,Q_channel,RefBuf_I(22),RefBuf_Q(22),Cor23_do(31 downto 16));
            Cor23_di(15 downto 0)  <=
            Correlate_Q(I_channel,Q_channel,RefBuf_I(22),RefBuf_Q(22),Cor23_do(15 downto 0));
        elsif (state=s1) then
            Cor23_di <= (others=>'0');
        end if;
    END IF;
```

```
END PROCESS Cor_23;

Cor_24: PROCESS (clk)
BEGIN
    IF (clk'event AND clk='0') THEN
        if (state=s2) then
            Cor24_di(31 downto 16) <=
            Correlate_I(I_channel,Q_channel,RefBuf_I(23),RefBuf_Q(23),Cor24_do(31 downto 16));
            Cor24_di(15 downto 0)  <=
            Correlate_Q(I_channel,Q_channel,RefBuf_I(23),RefBuf_Q(23),Cor24_do(15 downto 0));
        elsif (state=s1) then
            Cor24_di <= (others=>'0');
        end if;
    END IF;
END PROCESS Cor_24;

Cor_25: PROCESS (clk)
BEGIN
    IF (clk'event AND clk='0') THEN
        if (state=s2) then
            Cor25_di(31 downto 16) <=
            Correlate_I(I_channel,Q_channel,RefBuf_I(24),RefBuf_Q(24),Cor25_do(31 downto 16));
            Cor25_di(15 downto 0)  <=
            Correlate_Q(I_channel,Q_channel,RefBuf_I(24),RefBuf_Q(24),Cor25_do(15 downto 0));
        elsif (state=s1) then
            Cor25_di <= (others=>'0');
        end if;
    END IF;
END PROCESS Cor_25;

Cor_26: PROCESS (clk)
BEGIN
    IF (clk'event AND clk='0') THEN
        if (state=s2) then
            Cor26_di(31 downto 16) <=
            Correlate_I(I_channel,Q_channel,RefBuf_I(25),RefBuf_Q(25),Cor26_do(31 downto 16));
            Cor26_di(15 downto 0)  <=
            Correlate_Q(I_channel,Q_channel,RefBuf_I(25),RefBuf_Q(25),Cor26_do(15 downto 0));
        elsif (state=s1) then
            Cor26_di <= (others=>'0');
        end if;
    END IF;
END PROCESS Cor_26;

Cor_27: PROCESS (clk)
BEGIN
    IF (clk'event AND clk='0') THEN
        if (state=s2) then
            Cor27_di(31 downto 16) <=
            Correlate_I(I_channel,Q_channel,RefBuf_I(26),RefBuf_Q(26),Cor27_do(31 downto 16));
            Cor27_di(15 downto 0)  <=
            Correlate_Q(I_channel,Q_channel,RefBuf_I(26),RefBuf_Q(26),Cor27_do(15 downto 0));
        elsif (state=s1) then
            Cor27_di <= (others=>'0');
        end if;
    END IF;
END PROCESS Cor_27;

Cor_28: PROCESS (clk)
BEGIN
    IF (clk'event AND clk='0') THEN
        if (state=s2) then
            Cor28_di(31 downto 16) <=
            Correlate_I(I_channel,Q_channel,RefBuf_I(27),RefBuf_Q(27),Cor28_do(31 downto 16));
```

```
            Cor28_di(15 downto 0)  <=
            Correlate_Q(I_channel,Q_channel,RefBuf_I(27),RefBuf_Q(27),Cor28_do(15 downto 0));
        elsif (state=s1) then
            Cor28_di <= (others=>'0');
        end if;
    END IF;
END PROCESS Cor_28;

Cor_29: PROCESS (clk)
BEGIN
    IF (clk'event AND clk='0') THEN
        if (state=s2) then
            Cor29_di(31 downto 16) <=
            Correlate_I(I_channel,Q_channel,RefBuf_I(28),RefBuf_Q(28),Cor29_do(31 downto 16));
            Cor29_di(15 downto 0)  <=
            Correlate_Q(I_channel,Q_channel,RefBuf_I(28),RefBuf_Q(28),Cor29_do(15 downto 0));
        elsif (state=s1) then
            Cor29_di <= (others=>'0');
        end if;
    END IF;
END PROCESS Cor_29;

Cor_30: PROCESS (clk)
BEGIN
    IF (clk'event AND clk='0') THEN
        if (state=s2) then
            Cor30_di(31 downto 16) <=
            Correlate_I(I_channel,Q_channel,RefBuf_I(29),RefBuf_Q(29),Cor30_do(31 downto 16));
            Cor30_di(15 downto 0)  <=
            Correlate_Q(I_channel,Q_channel,RefBuf_I(29),RefBuf_Q(29),Cor30_do(15 downto 0));
        elsif (state=s1) then
            Cor30_di <= (others=>'0');
        end if;
    END IF;
END PROCESS Cor_30;

Cor_31: PROCESS (clk)
BEGIN
    IF (clk'event AND clk='0') THEN
        if (state=s2) then
            Cor31_di(31 downto 16) <=
            Correlate_I(I_channel,Q_channel,RefBuf_I(30),RefBuf_Q(30),Cor31_do(31 downto 16));
            Cor31_di(15 downto 0)  <=
            Correlate_Q(I_channel,Q_channel,RefBuf_I(30),RefBuf_Q(30),Cor31_do(15 downto 0));
        elsif (state=s1) then
            Cor31_di <= (others=>'0');
        end if;
    END IF;
END PROCESS Cor_31;

Cor_32: PROCESS (clk)
BEGIN
    IF (clk'event AND clk='0') THEN
        if (state=s2) then
            Cor32_di(31 downto 16) <=
            Correlate_I(I_channel,Q_channel,RefBuf_I(31),RefBuf_Q(31),Cor32_do(31 downto 16));
            Cor32_di(15 downto 0)  <=
            Correlate_Q(I_channel,Q_channel,RefBuf_I(31),RefBuf_Q(31),Cor32_do(15 downto 0));
        elsif (state=s1) then
            Cor32_di <= (others=>'0');
        end if;
    END IF;
END PROCESS Cor_32;
```

```
Cor_33: PROCESS (clk)
BEGIN
    IF (clk'event AND clk='0') THEN
        if (state=s2) then
            Cor33_di(31 downto 16) <=
            Correlate_I(I_channel,Q_channel,RefBuf_I(32),RefBuf_Q(32),Cor33_do(31 downto 16));
            Cor33_di(15 downto 0)  <=
            Correlate_Q(I_channel,Q_channel,RefBuf_I(32),RefBuf_Q(32),Cor33_do(15 downto 0));
        elsif (state=s1) then
            Cor33_di <= (others=>'0');
        end if;
    END IF;
END PROCESS Cor_33;

Cor_34: PROCESS (clk)
BEGIN
    IF (clk'event AND clk='0') THEN
        if (state=s2) then
            Cor34_di(31 downto 16) <=
            Correlate_I(I_channel,Q_channel,RefBuf_I(33),RefBuf_Q(33),Cor34_do(31 downto 16));
            Cor34_di(15 downto 0)  <=
            Correlate_Q(I_channel,Q_channel,RefBuf_I(33),RefBuf_Q(33),Cor34_do(15 downto 0));
        elsif (state=s1) then
            Cor34_di <= (others=>'0');
        end if;
    END IF;
END PROCESS Cor_34;

Cor_35: PROCESS (clk)
BEGIN
    IF (clk'event AND clk='0') THEN
        if (state=s2) then
            Cor35_di(31 downto 16) <=
            Correlate_I(I_channel,Q_channel,RefBuf_I(34),RefBuf_Q(34),Cor35_do(31 downto 16));
            Cor35_di(15 downto 0)  <=
            Correlate_Q(I_channel,Q_channel,RefBuf_I(34),RefBuf_Q(34),Cor35_do(15 downto 0));
        elsif (state=s1) then
            Cor35_di <= (others=>'0');
        end if;
    END IF;
END PROCESS Cor_35;

Cor_36: PROCESS (clk)
BEGIN
    IF (clk'event AND clk='0') THEN
        if (state=s2) then
            Cor36_di(31 downto 16) <=
            Correlate_I(I_channel,Q_channel,RefBuf_I(35),RefBuf_Q(35),Cor36_do(31 downto 16));
            Cor36_di(15 downto 0)  <=
            Correlate_Q(I_channel,Q_channel,RefBuf_I(35),RefBuf_Q(35),Cor36_do(15 downto 0));
        elsif (state=s1) then
            Cor36_di <= (others=>'0');
        end if;
    END IF;
END PROCESS Cor_36;

Cor_37: PROCESS (clk)
BEGIN
    IF (clk'event AND clk='0') THEN
        if (state=s2) then
            Cor37_di(31 downto 16) <=
            Correlate_I(I_channel,Q_channel,RefBuf_I(36),RefBuf_Q(36),Cor37_do(31 downto 16));
            Cor37_di(15 downto 0)  <=
            Correlate_Q(I_channel,Q_channel,RefBuf_I(36),RefBuf_Q(36),Cor37_do(15 downto 0));
```

```
            elsif (state=s1) then
                Cor37_di <= (others=>'0');
            end if;
        END IF;
    END PROCESS Cor_37;

    Cor_38: PROCESS (clk)
    BEGIN
        IF (clk'event AND clk='0') THEN
            if (state=s2) then
                Cor38_di(31 downto 16) <=
                Correlate_I(I_channel,Q_channel,RefBuf_I(37),RefBuf_Q(37),Cor38_do(31 downto 16));
                Cor38_di(15 downto 0)  <=
                Correlate_Q(I_channel,Q_channel,RefBuf_I(37),RefBuf_Q(37),Cor38_do(15 downto 0));
            elsif (state=s1) then
                Cor38_di <= (others=>'0');
            end if;
        END IF;
    END PROCESS Cor_38;

    Cor_39: PROCESS (clk)
    BEGIN
        IF (clk'event AND clk='0') THEN
            if (state=s2) then
                Cor39_di(31 downto 16) <=
                Correlate_I(I_channel,Q_channel,RefBuf_I(38),RefBuf_Q(38),Cor39_do(31 downto 16));
                Cor39_di(15 downto 0)  <=
                Correlate_Q(I_channel,Q_channel,RefBuf_I(38),RefBuf_Q(38),Cor39_do(15 downto 0));
            elsif (state=s1) then
                Cor39_di <= (others=>'0');
            end if;
        END IF;
    END PROCESS Cor_39;

    --//hide_sp  --////////////////////////////////////

    --//hide_sp --//*************************************************//

END behavior;

//*************************************************//

LIBRARY ieee; use ieee.std_logic_1164.all; use
ieee.std_logic_unsigned.all;

ENTITY Acc_ram is
    PORT (
        clk : IN std_logic;
        we  : IN std_logic;
        wraddress : IN std_logic_vector(1 downto 0);
        rdaddress : IN std_logic_vector(1 downto 0);
        di  : IN std_logic_vector(31 downto 0);
        do  : OUT std_logic_vector(31 downto 0)
    );
END Acc_ram;

ARCHITECTURE rtl OF Acc_ram IS
    type ram_type is array (3 downto 0) of std_logic_vector(31 downto 0);
    signal RAM : ram_type :=
            (
            "00000000000000000000000000000000",
            "00000000000000000000000000000000",
            "00000000000000000000000000000000",
            "00000000000000000000000000000000");
```

```
    signal read_rdaddress : std_logic_vector(1 downto 0) := "00";
BEGIN
    process (clk)
    begin
        if (clk'event and clk='1') then
            if (we='1') then
                RAM(conv_integer(wraddress)) <= di;
            end if;
            read_rdaddress <= rdaddress;
        end if;
    end process;
    do <= RAM(conv_integer(read_rdaddress));
END rtl;

//****************************************************//

LIBRARY ieee; use ieee.std_logic_1164.all; use
ieee.std_logic_unsigned.all;

ENTITY Ins_ram is
    PORT (
        clk : IN std_logic;
        we  : IN std_logic;
        wraddress : IN std_logic_vector(7 downto 0);
        rdaddress : IN std_logic_vector(7 downto 0);
        di  : IN std_logic_vector(34 downto 0);
        do  : OUT std_logic_vector(34 downto 0)
    );
END Ins_ram;

ARCHITECTURE rtl OF Ins_ram IS
    constant MaxDe_sample : integer := 159;    -- Maximum-DelaySpread(samples)
    type ram_type is array (MaxDe_sample downto 0) of std_logic_vector(34 downto 0);
    signal RAM : ram_type :=
        (
        "00000000000000000000000000000000000",
        "00000000000000000000000000000000000",
        "00000000000000000000000000000000000",
        "00000000000000000000000000000000000",
        "00000000000000000000000000000000000",
        "00000000000000000000000000000000000",
        "00000000000000000000000000000000000",
        "00000000000000000000000000000000000",
        "00000000000000000000000000000000000",
        "00000000000000000000000000000000000",
        "00000000000000000000000000000000000",
        "00000000000000000000000000000000000",
        "00000000000000000000000000000000000",
        "00000000000000000000000000000000000",
        "00000000000000000000000000000000000",
        "00000000000000000000000000000000000",
        "00000000000000000000000000000000000",
        "00000000000000000000000000000000000",
        "00000000000000000000000000000000000",
        "00000000000000000000000000000000000",
        "00000000000000000000000000000000000",
        "00000000000000000000000000000000000",
        "00000000000000000000000000000000000",
        "00000000000000000000000000000000000",
        "00000000000000000000000000000000000",
        "00000000000000000000000000000000000",
        "00000000000000000000000000000000000",
        "00000000000000000000000000000000000",
        "00000000000000000000000000000000000",
```

```
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
"00000000000000000000000000000000000",
```

```
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
"000000000000000000000000000000000000000",
```

```
        "00000000000000000000000000000000",
        "00000000000000000000000000000000",
        "00000000000000000000000000000000");
    signal read_rdaddress : std_logic_vector(7 downto 0) := "00000000";
BEGIN
    process (clk)
    begin
        if (clk'event and clk='1') then
            if (we='1') then
                RAM(conv_integer(wraddress)) <= di;
            end if;
            read_rdaddress <= rdaddress;
        end if;
    end process;
    do <= RAM(conv_integer(read_rdaddress));
END rtl;

%%%%%

-- revised for CLK input is 3.84 MHz LIBRARY ieee; USE
ieee.std_logic_1164.all; USE ieee.std_logic_arith.all; USE
ieee.std_logic_signed.all;

ENTITY ULPS_ScLongCodeGen IS
    PORT (  clk : IN std_logic;
        reset : IN std_logic;
        FrameSync : IN std_logic;
        ScCode_number : IN std_logic_vector(23 downto 0);   -- 24-bit UL Scrambling Code Number.

        Code_I : OUT std_logic_vector(0 to 1);
        Code_Q : OUT std_logic_vector(0 to 1)
        );
END ULPS_ScLongCodeGen ;

ARCHITECTURE behavior OF ULPS_ScLongCodeGen IS
    signal Xint : std_logic_vector(24 downto 0):= (24=>'1', others=>'0');
    signal sign_flag : std_logic := '0';
    signal Clong2even : std_logic := '0';
    signal Code_I_dum : std_logic_vector(0 to 1) := "00";
    signal Code_Q_dum : std_logic_vector(0 to 1) := "00";
BEGIN
    Code_I <= Code_I_dum;
    Code_Q <= Code_Q_dum;

    p1: PROCESS (clk,reset)
        variable out_en : std_logic := '0';
        variable RegX : std_logic_vector(24 downto 0);
        variable RegY : std_logic_vector(24 downto 0);
        variable FwX, FwY : std_logic := '0';
        variable FbX1, FbX2, FbY1, FbY2, FbY3, FbY4 : std_logic := '0';
        variable Clong1, Clong2 : std_logic := '0';
    BEGIN
        Xint(23 downto 0) <= ScCode_number;

        -- Check Async reset.
        IF (reset = '1') THEN
            Xint <= (24='1', others='0');
            sign_flag <= '0';
            out_en := '0';
            Code_I_dum <= "00";
            Code_Q_dum <= "00";
            Clong2even <= '0';
        -- Every clk operation.
        ELSIF (clk'event AND clk='1') THEN
```

```
IF (FrameSync = '1') THEN
    out_en := '1';
    RegX := Xint;
    RegY := (others => '1');
    sign_flag <= '1';

    Clong1 := (RegX(0) xor '1');
    Clong2 := (RegX(4) xor RegX(7) xor RegX(18) xor '1');
    Clong2even <= Clong2;

    -- send out UL complex Scrambling Code
    Code_I_dum <= Clong1 & "1";
    if (Clong1 = '0') then
        Code_Q_dum <= Clong2 & "1";
    else
        Code_Q_dum <= (not(Clong2)) & "1";
    end if;
ELSE
    -- Sending Code_output of the chips in this frame according to 'output enable'
    IF (out_en = '1') THEN
        -- Shift RegX & RegY ------------------------
        FbX1 := RegX(0);
        FbX2 := RegX(3);
        RegX(23 downto 0) := RegX(24 downto 1);
        RegX(24) := FbX1 xor FbX2;

        FbY1 := RegY(0);
        FbY2 := RegY(1);
        FbY3 := RegY(2);
        FbY4 := RegY(3);
        RegY(23 downto 0) := RegY(24 downto 1);
        RegY(24) := FbY1 xor FbY2 xor FbY3 xor FbY4;
        ------------------------------------------

        -- Send out UL complex Scrambling Code
        FwX := RegX(4) xor RegX(7) xor RegX(18);
        FwY := RegY(4) xor RegY(6) xor RegY(17);
        Clong1 := RegX(0) xor RegY(0);
        Clong2 := FwX xor FwY;

        -- Send out Code_I_dum -----
        Code_I_dum <= Clong1 & "1";
        -- Send out Code_Q_dum -----
        if (sign_flag = '0') then  --(+)use the present Clong2
            if (Clong1 = '0') then
                Code_Q_dum <= Clong2 & "1";
            else
                Code_Q_dum <= (not(Clong2)) & "1";
            end if;
        else --  sign_flg = '1'  (-) use the past Clong2 <= Clong2even
            if (Clong1 = '0') then
                Code_Q_dum <= (not(Clong2even)) & "1";
            else
                Code_Q_dum <= Clong2even & "1";
            end if;
        end if;

        sign_flag <= sign_flag xor '1';
        Clong2even <= Clong2;
    ELSE
        Code_I_dum <= "00";
        Code_Q_dum <= "00";
    END IF;
END IF;
```

```
        END IF;
    END PROCESS p1;
END behavior;
```