Appendix I

Maximum Ratio Combining Used in Rake Receiver by Using VHDL Code

Program Description

Input:

- Reset signal.
- Flag signal for the 1^{st} finger.
- Frame synchronization signal for the 1^{st} finger.
- Clock signal of I channel for the 1^{st} finger.
- Equalized I data from the 1^{st} finger (14 bits).
- Clock signal of Q channel for the 1^{st} finger.
- Equalized Q data from the 1^{st} finger (14 bits).
- Frame synchronization signal for the 2^{nd} finger.
- Clock signal of I channel for the 2^{nd} finger.
- Equalized I data from the 2^{nd} finger (14 bits).
- Clock signal of Q channel for the 2^{nd} finger.
- Equalized Q data from the 2^{nd} finger (14 bits).
- Frame synchronization signal for the 3^{rd} finger.
- Clock signal of I channel for the 3^{rd} finger.
- Equalized I data from the 3^{rd} finger (14 bits).
- Clock signal of Q channel for the 3^{rd} finger.
- Equalized Q data from the 3^{rd} finger (14 bits).
- Spreading factor (3 bits).

Output:

- Frame synchronization signal.
- Clock signal of Q channel.
- Decoded I data.
- Clock signal of Q channel.
- Decoded Q data.

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

ENTITY Maximum_Ratio_Combiner IS
    Port (  reset : IN std_logic;
        Flag_MSpath : IN std_logic;
        FrameSync_MS : IN std_logic;
        clk_MS : IN std_logic;
        MS_I : IN std_logic_vector(13 downto 0);
        clk_MS_Q : IN std_logic;
        MS_Q : IN std_logic_vector(13 downto 0);

        FrameSync_SL1 : IN std_logic;
        clk_SL1 : IN std_logic;
        SL1_I : IN std_logic_vector(13 downto 0);
        clk_SL1_Q : IN std_logic;
        SL1_Q : IN std_logic_vector(13 downto 0);

        FrameSync_SL2 : IN std_logic;
        clk_SL2 : IN std_logic;
        SL2_I : IN std_logic_vector(13 downto 0);
        clk_SL2_Q : IN std_logic;
        SL2_Q : IN std_logic_vector(13 downto 0);

        SF : IN std_logic_vector(2 downto 0);

        FrameSync_MRC : OUT std_logic;
        clk_MRC : OUT std_logic;
        MRC_I : OUT std_logic;
        clk_MRC_Q : OUT std_logic;
        MRC_Q : OUT std_logic
        );
END Maximum_Ratio_Combiner ;

ARCHITECTURE behavior OF Maximum_Ratio_Combiner IS
    FUNCTION MrcExt (a:std_logic_vector) RETURN std_logic_vector IS
            variable tmp : std_logic_vector(14 downto 0);
        BEGIN
            tmp := a(13)&a;
            return tmp;
    END MrcExt;
    constant MaxDelaySpread : integer := 20;
    constant Default_DisableNo : integer := 9600;
                            -- Note! Counter_MRC cannot reach this value.

    -- The signal of P1 : Check Disable and beyond MaxDelaySpread.
    signal DelaySL1 : integer range 0 to MaxDelaySpread := 0;  -- default value = 0 delay.
    signal DelaySL2 : integer range 0 to MaxDelaySpread := 0;
    signal SL1_en : std_logic := '0';
    signal SL2_en : std_logic := '0';

    -- The signal of P2 : Check shortened new SL_FrameSync.
    signal DisableNo_SL1 : integer range 0 to Default_DisableNo := Default_DisableNo;
    signal DisableNo_Sl2 : integer range 0 to Default_DisableNo := Default_DisableNo;
    signal MaxSym_Frame : integer range 74 to (Default_DisableNo-1);

    -- The signal of P3 : Check lengthened new SL_FrameSync.
    signal Flag_A : std_logic := '0';    -- flag of Master finger related to slave1_finger
    signal Flag_B : std_logic := '0';    -- flag of Slave_1 finger
    signal Flag_C : std_logic := '0';    -- flag of Master finger related to slave2_finger
    signal Flag_D : std_logic := '0';    -- flag of Slave_2 finger
    signal Flag_AxorB : std_logic := '0';  -- flag for Buffer_condition of slave_1 finger
    signal Flag_CxorD : std_logic := '0';  -- flag for Buffer_condition of slave_2 finger
```

```
    signal LenEn_SL1 : std_logic := '0';
    signal LenEn_SL2 : std_logic := '0';

    -- The signal of P_Combine.
    TYPE EqDPCHBufType IS ARRAY (0 to MaxDelaySpread) OF std_logic_vector(13 downto 0);
    signal MSBuf_I,MSBuf_Q : EqDPCHBufType := (others=>"00000000000000");
    signal SL1Buf_I,SL1Buf_Q : EqDPCHBufType := (others=>"00000000000000");
    signal SL2Buf_I,SL2Buf_Q : EqDPCHBufType := (others=>"00000000000000");
    signal indexMS : integer range 0 to MaxDelaySpread;
    signal clk_en : std_logic := '0';

    -- dummy output signal --
    signal FrameSync_MRC_dum : std_logic := '0';
    signal MRC_I_dum : std_logic_vector(14 downto 0) := (others=>'0');
    signal MRC_Q_dum : std_logic_vector(14 downto 0) := (others=>'0');

BEGIN
    FrameSync_MRC <= FrameSync_MRC_dum;
    MRC_I <= MRC_I_dum(14);
    MRC_Q <= MRC_Q_dum(14);

    clk_MRC <= NOT(clk_MS) when clk_en='1' else
           '0';
    clk_MRC_Q <= NOT(clk_MS_Q) when clk_en='1' else
           '0';

    with SF select
           MaxSym_Frame <= 9599 when "000",
                   4799 when "001",
                   2399 when "010",
                   1199 when "011",
                    599 when "100",
                    299 when "101",
                    149 when "110",
                     74 when others;

    with SF select
           indexMS <=  20 when "000",
                  10 when "001",
                   5 when "010",
                   3 when "011",
                   2 when "100",
                   1 when others;

    -- Process 1 :      [disabled]
    -- Estimate slave finger's delay which related to master finger's.
    -- Check slave finger is disabled, or has delay beyond Maximum_delay_spread.
    p1_SL1: PROCESS (clk_MS,reset)
        variable count : integer range 0 to MaxDelaySpread+1 := MaxDelaySpread+1;
        variable flag : std_logic := '0';
    BEGIN
        IF (reset='1') THEN
            SL1_en <= '0';
            DelaySL1 <= 0;
            count := MaxDelaySpread+1;
            flag := '0';
        ELSIF (clk_MS'event AND clk_MS='1') THEN
            if (FrameSync_MS='1') then
                count := 0;
                flag := '1';
            end if;

            if (FrameSync_SL1='1') then
                 DelaySL1 <= count;
```

```
             flag := '0';
        end if;

        if (count<indexMS) then
            count := count + 1;
        elsif (count=indexMS) then
            count := MaxDelaySpread+1;
            if (flag='0') then
                SL1_en <= '1';      -- Enable Slave_finger
            else
                SL1_en <= '0';  -- Disable Slave _finger
            end if;
        end if;
    END IF;
END PROCESS p1_SL1;

p1_SL2: PROCESS (clk_MS,reset)
    variable count : integer range 0 to MaxDelaySpread+1 := MaxDelaySpread+1;
    variable flag : std_logic := '0';
BEGIN
    IF (reset='1') THEN
        SL2_en <= '0';
        DelaySL2 <= 0;
        count := MaxDelaySpread+1;
        flag := '0';
    ELSIF (clk_MS'event AND clk_MS='1') THEN
        if (FrameSync_MS='1') then
            count := 0;
            flag := '1';
        end if;

        if (FrameSync_SL2='1') then
             DelaySL2 <= count;
             flag := '0';
        end if;

        if (count<indexMS) then
            count := count + 1;
        elsif (count=indexMS) then
            count := MaxDelaySpread+1;
            if (flag='0') then
                SL2_en <= '1';      -- Enable Slave_finger
            else
                SL2_en <= '0';  -- Disable Slave _finger
            end if;
        end if;
    END IF;
END PROCESS p1_SL2;

-- Process 2 :  [shortened]
p2_SL1: PROCESS (clk_SL1,reset)
    variable count : integer range 0 to (Default_DisableNo-1) :=    (Default_DisableNo-1);
BEGIN
    IF (reset='1') THEN
        DisableNo_SL1 <= Default_DisableNo;
        count := Default_DisableNo-1;
    ELSIF (clk_SL1'event AND clk_SL1='1') THEN
        -- Condition for assigning Disable_Symbol_No.
        -- & increase 'count' variable every rising_edge of clk_SL1
        if (FrameSync_SL1='1') then
            if (count < MaxSym_Frame) then
                DisableNo_SL1 <= count;
            else --(>= MaxSym_Frame)
                DisableNo_SL1 <= Default_DisableNo;
```

```
            end if;
            count := 1;
        elsif (count = MaxSym_Frame) then
            DisableNo_SL1 <= Default_DisableNo;
        elsif (count < MaxSym_Frame) then
            count := count +1;
        end if;
    END IF;
END PROCESS p2_SL1;

p2_SL2: PROCESS (clk_SL2,reset)
    variable count : integer range 0 to (Default_DisableNo-1) :=    (Default_DisableNo-1);
BEGIN
    IF (reset='1') THEN
        DisableNo_SL2 <= Default_DisableNo;
        count := Default_DisableNo-1;
    ELSIF (clk_SL2'event AND clk_SL2='1') THEN
        -- Condition for assigning Disable_Symbol_No.
        -- & increase 'count' variable every rising_edge of clk_SL2
        if (FrameSync_SL2='1') then
            if (count < MaxSym_Frame) then
                DisableNo_SL2 <= count;
            else --(>= MaxSym_Frame)
                DisableNo_SL2 <= Default_DisableNo;
            end if;
            count := 1;
        elsif (count = MaxSym_Frame) then
            DisableNo_SL2 <= Default_DisableNo;
        elsif (count < MaxSym_Frame) then
            count := count +1;
        end if;
    END IF;
END PROCESS p2_SL2;

-- Process 3 :  [lengthend]
--      Check whether slave finger is lengthened?  or Missed order btw MS_clk and SL_clk
Flag_AxorB <= Flag_A xor Flag_B;
Flag_CxorD <= Flag_C xor Flag_D;
-- Flag_A Process : flag of Master finger related to slave1_finger ------
p3SL1_1: PROCESS (clk_MS)
BEGIN
    IF (clk_MS'event and clk_MS='1') THEN
        if (FrameSync_SL1='1') then
            Flag_A <= '1';
        else
            Flag_A <= Flag_A xor '1';
        end if;
    END IF;
END PROCESS p3SL1_1;
-- Flag_B Process : flag of Slave_1 finger ------
p3SL1_2: PROCESS (clk_SL1)
BEGIN
    IF (clk_SL1'event and clk_SL1='1') THEN
        if (FrameSync_SL1='1') then
            Flag_B <= '1';
        else
            Flag_B <= Flag_B xor '1';
        end if;
    END IF;
END PROCESS p3SL1_2;

p3SL1_3: PROCESS (clk_MS,reset)
    variable Mode_xor : std_logic:= '0';  -- 0: simultaneous, 1: non-simultaneous
    variable index : integer range 0 to 2 := 0;
```

```
BEGIN
    IF (reset='1') THEN
        LenEn_SL1 <= '0';
    ELSIF (clk_MS'event and clk_MS='0') THEN
        if (FrameSync_SL1='1') then
            index := 0;
        end if;

        if (index<2) then
            LenEn_SL1 <= '1';  -- Enable Slave_finger
            Mode_xor := Flag_AxorB;
            index := index+1;
        elsif (Mode_xor='0' and Flag_AxorB='1') then
            LenEn_SL1 <= '0';  -- Disable Slave_finger
        elsif (Mode_xor='1' and Flag_AxorB='0') then
            LenEn_SL1 <= '0';  -- Disable Slave_finger
        end if;
    END IF;
END PROCESS p3SL1_3;


-- Flag_C Process : flag of Master finger related to slave2_finger ------
p3SL2_1: PROCESS (clk_MS)
BEGIN
    IF (clk_MS'event and clk_MS='1') THEN
        if (FrameSync_SL2='1') then
            Flag_C <= '1';
        else
            Flag_C <= Flag_C xor '1';
        end if;
    END IF;
END PROCESS p3SL2_1;
-- Flag_D Process : flag of Slave_2 finger ------
p3SL2_2: PROCESS (clk_SL2)
BEGIN
    IF (clk_SL2'event and clk_SL2='1') THEN
        if (FrameSync_SL2='1') then
            Flag_D <= '1';
        else
            Flag_D <= Flag_D xor '1';
        end if;
    END IF;
END PROCESS p3SL2_2;

p3SL2_3: PROCESS (clk_MS,reset)
    variable Mode_xor : std_logic:= '0';  -- 0: simultaneous, 1: non-simultaneous
    variable index : integer range 0 to 2 := 0;
BEGIN
    IF (reset='1') THEN
        LenEn_SL2 <= '0';
    ELSIF (clk_MS'event and clk_MS='0') THEN
        if (FrameSync_SL2='1') then
            index := 0;
        end if;

        if (index<2) then
            LenEn_SL2 <= '1';  -- Enable Slave_finger
            Mode_xor := Flag_CxorD;
            index := index+1;
        elsif (Mode_xor='0' and Flag_CxorD='1') then
            LenEn_SL2 <= '0';  -- Disable Slave_finger
        elsif (Mode_xor='1' and Flag_CxorD='0') then
            LenEn_SL2 <= '0';  -- Disable Slave_finger
        end if;
```

```
    END IF;
END PROCESS p3SL2_3;


-- Process_Combine : Combining each finger together and use clk_MS(finger#1)
-- as the reference clock.
P_Combine: PROCESS (clk_MS,reset)
    variable DelayIndex : integer range 0 to MaxDelaySpread := 0;
    variable FrmFlag : std_logic := '0';
    variable indexSL1 : integer range 0 to MaxDelaySpread := MaxDelaySpread;
    variable indexSL2 : integer range 0 to MaxDelaySpread := MaxDelaySpread;
    variable Counter_MRC : integer range 0 to (Default_DisableNo-1) := 0;
    variable TmpSL1_I, TmpSL1_Q : std_logic_vector(13 downto 0);
    variable TmpSL2_I, TmpSL2_Q : std_logic_vector(13 downto 0);
BEGIN
    IF (reset='1') THEN
        clk_en <= '0';
        FrmFlag := '0';
        FrameSync_MRC_dum <= '0';
        MRC_I_dum <= (others=>'0');
        MRC_Q_dum <= (others=>'0');

        MSBuf_I <= (others=>"00000000000000");
        MSBuf_Q <= (others=>"00000000000000");
        SL1Buf_I <= (others=>"00000000000000");
        SL1Buf_Q <= (others=>"00000000000000");
        SL2Buf_I <= (others=>"00000000000000");
        SL2Buf_Q <= (others=>"00000000000000");
    ELSIF (clk_MS'event AND clk_MS='1') THEN
    -- Create 'FrameSync_MRC' & Enable 'clk_MRC' & Set 'indexSL1' & reset 'Counter_MRC' --
        IF (FrameSync_MS='1') THEN
            FrmFlag := '1';
            DelayIndex := 0;
        ELSIF (FrmFlag='1') THEN
            IF (DelayIndex=indexMS) THEN
                FrameSync_MRC_dum <= '1';
                FrmFlag := '0';
                clk_en <= '1';

                Counter_MRC := 0;
                indexSL1 := indexMS - DelaySL1;
                indexSL2 := indexMS - DelaySL2;
            ELSE
                DelayIndex := DelayIndex+1;
            END IF;
        ELSE
            FrameSync_MRC_dum <= '0';
        END IF;

            -- Calculate MRC_DPCH_I&Q --
        if ((Counter_MRC<DisableNo_SL1)AND(SL1_en='1')) then
                TmpSL1_I := SL1Buf_I(indexSL1);
                TmpSL1_Q := SL1Buf_Q(indexSL1);
            else
                TmpSL1_I := (others=>'0');
                TmpSL1_Q := (others=>'0');
            end if;

            if ((Counter_MRC<DisableNo_SL2)AND(SL2_en='1')) then
                TmpSL2_I := SL2Buf_I(indexSL2);
                TmpSL2_Q := SL2Buf_Q(indexSL2);
            else
                TmpSL2_I := (others=>'0');
                TmpSL2_Q := (others=>'0');
```

```
            end if;

            MRC_I_dum <= MrcExt(MSBuf_I(indexMS)) + MrcExt(TmpSL1_I) + MrcExt(TmpSL2_I);
            MRC_Q_dum <= MrcExt(MSBuf_Q(indexMS)) + MrcExt(TmpSL1_Q) + MrcExt(TmpSL2_Q);

        -- Shift Register MS&SL1 Buffer --
        MSBuf_I <= MS_I & MSBuf_I(0 to MaxDelaySpread-1);
        MSBuf_Q <= MS_Q & MSBuf_Q(0 to MaxDelaySpread-1);
        if((LenEn_SL1='1')AND(Flag_MSpath='0')) then
            SL1Buf_I <= SL1_I & SL1Buf_I(0 to MaxDelaySpread-1);
            SL1Buf_Q <= SL1_Q & SL1Buf_Q(0 to MaxDelaySpread-1);
        else
            SL1Buf_I <= "00000000000000" & SL1Buf_I(0 to MaxDelaySpread-1);
            SL1Buf_Q <= "00000000000000" & SL1Buf_Q(0 to MaxDelaySpread-1);
        end if;

        if((LenEn_SL2='1')AND(Flag_MSpath='0')) then
            SL2Buf_I <= SL2_I & SL2Buf_I(0 to MaxDelaySpread-1);
            SL2Buf_Q <= SL2_Q & SL2Buf_Q(0 to MaxDelaySpread-1);
        else
            SL2Buf_I <= "00000000000000" & SL2Buf_I(0 to MaxDelaySpread-1);
            SL2Buf_Q <= "00000000000000" & SL2Buf_Q(0 to MaxDelaySpread-1);
        end if;

        -- Increase CounterSym_MS --
        if (Counter_MRC < MaxSym_Frame) then
            Counter_MRC := Counter_MRC+1;
        else
            Counter_MRC := MaxSym_Frame;
        end if;
    END IF;
END PROCESS P_Combine;

END behavior;
```