Final Project

HW based FIR filter with kaiser window
(400kHz BW, 600kHz Sampling, 200kHz Symbol)

Department of Electronic Engineering

20203033 주 동 준　　20203023 우 상 욱

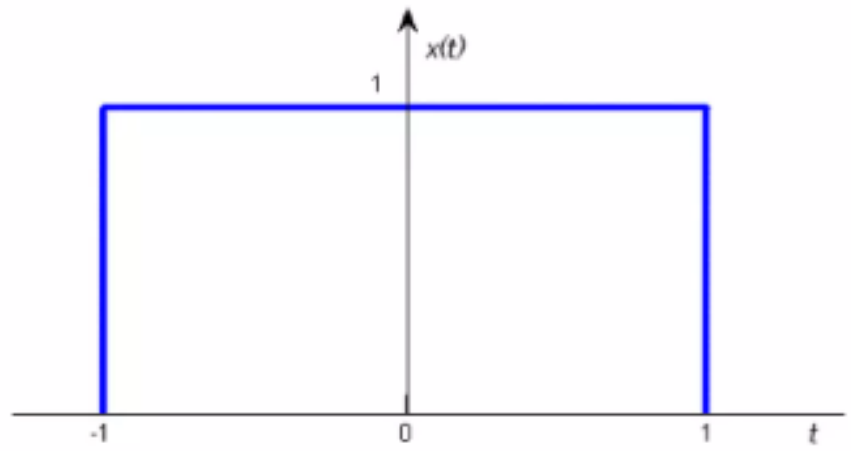
숭실대학교 Soongsil University

Contents

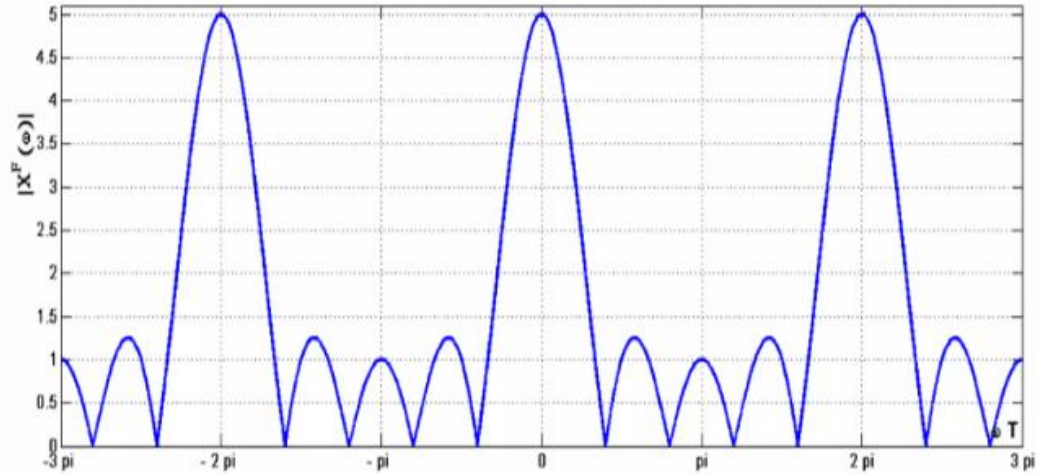
Introduction

Discrete-Time Fourier Transform (D.T.F.T.)

$$X^f(\omega) = \sum_{n=-\infty}^{+\infty} x[n]\, e^{-j\omega n}$$

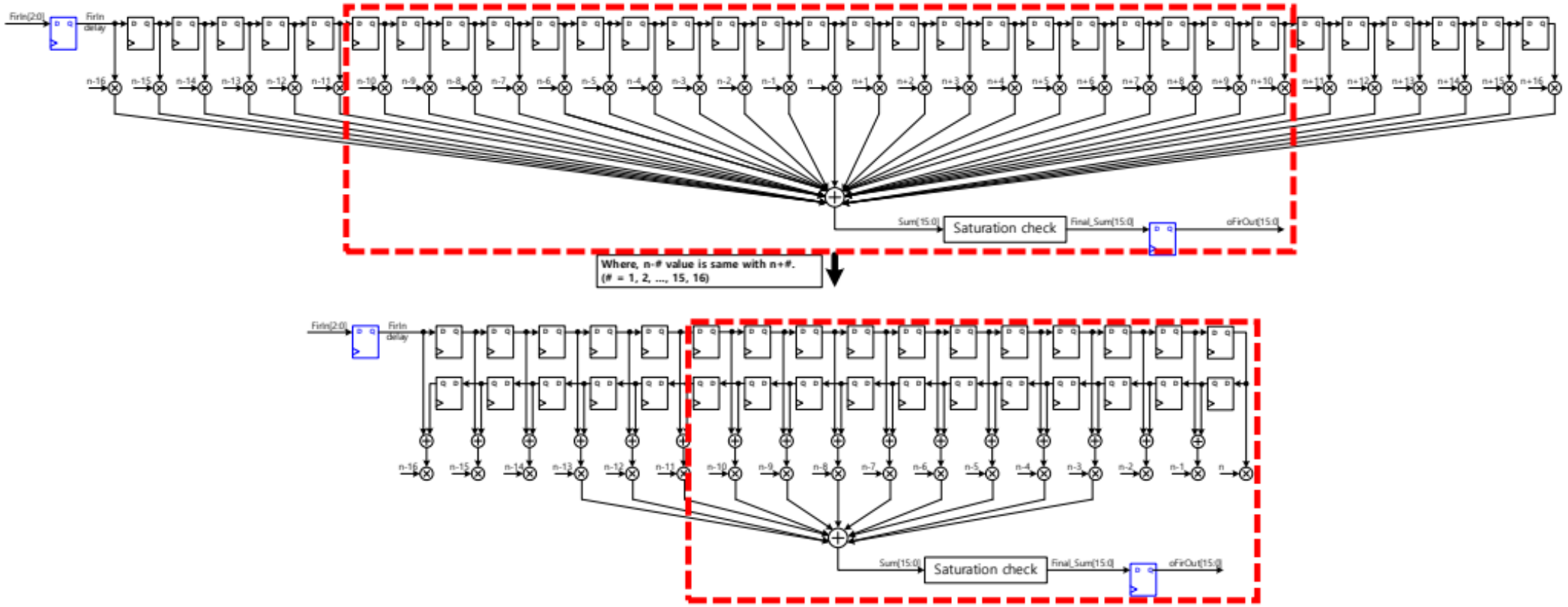


Time domain



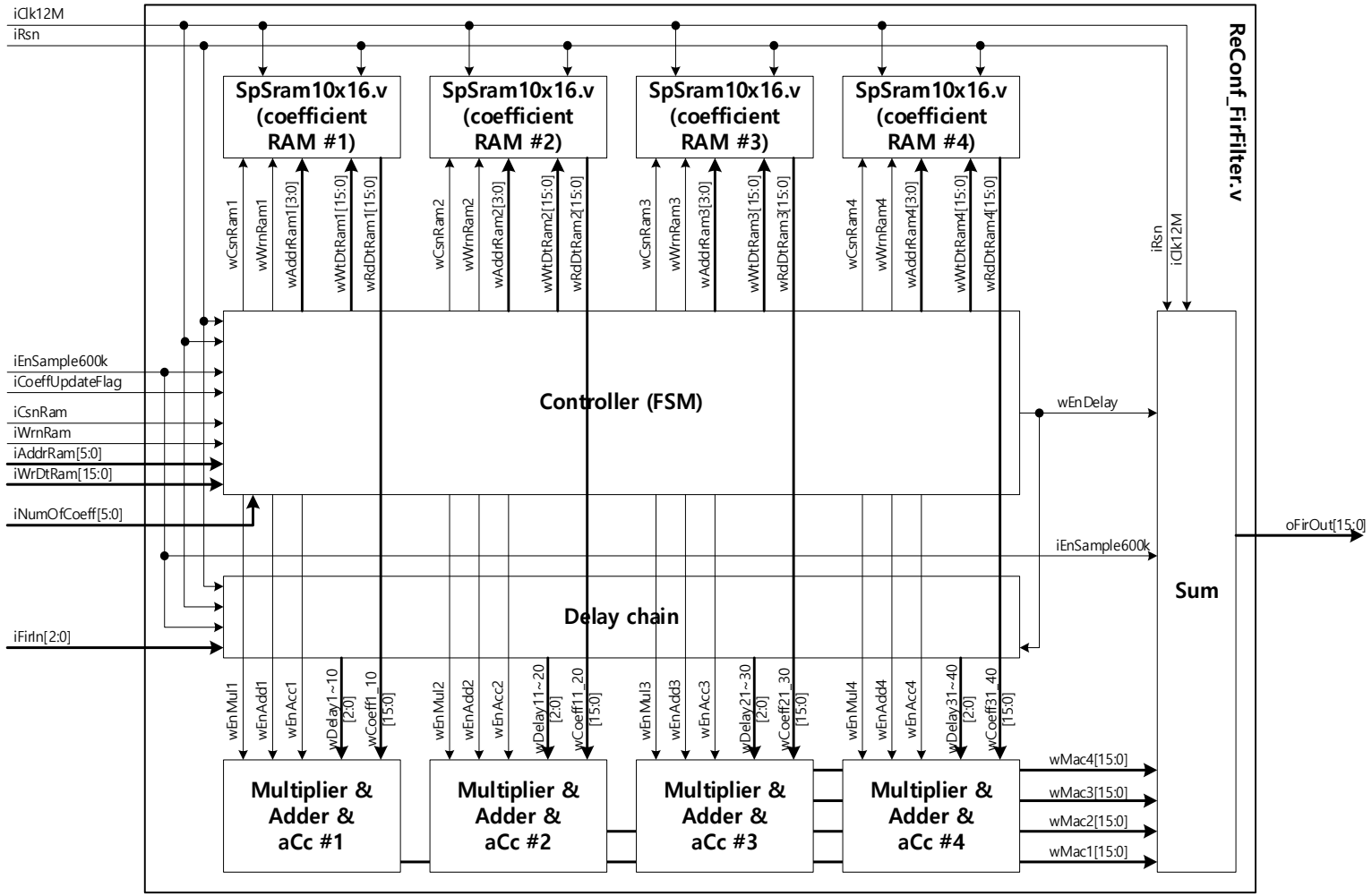
Frequency domain

Introduction



Symmetrical structure => Fold filter coefficient

Structure (Block diagram)



iClk12M : Reference Clock

iEnSample600k : Shift FirIn in Delay Chain

iNumOfCoeff : Number of Coefficient (Variable)

iCoeffUpdate : Update SP-SRAM (Critical signal)

iCsnRam : Select SP-SRAM

iWrnRam : R/W SP-SRAM

(Low => Write High => Read)

wEnMul : Start Multiplication

wEnAdd : Start Addition

wEnAcc : Start Accumulation

wEnDelay : Operate modules

(DelayChain, Multiplier, Accumulator, Sum)

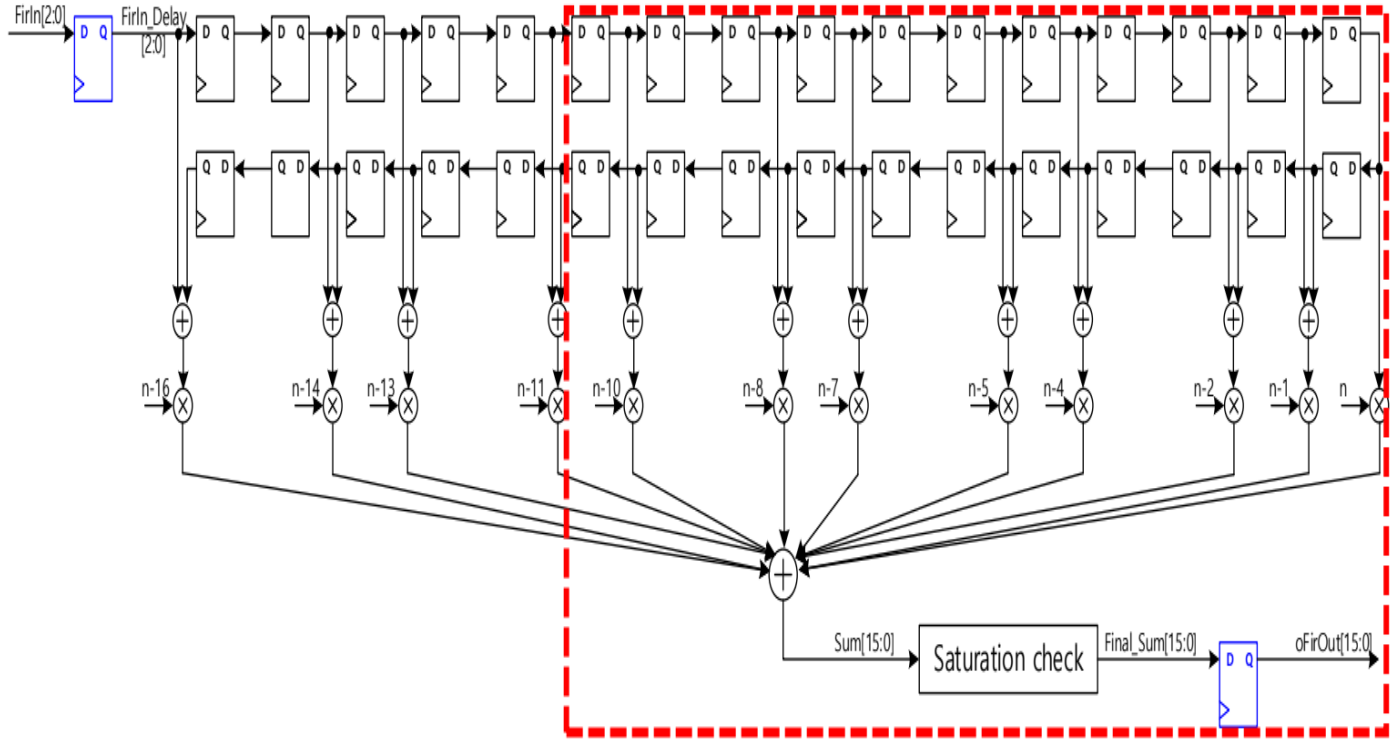
rCnt_0 : Check Order of Coeff Written at SRAM (40EA)
Check Order of Coeff Read at SRAM (10EA)

rCnt_1 : Check Number of iEnSample600k (80EA)

Structure (Block diagram)

Delay Chain

79-Tap Coefficient (Window size : 40EA)

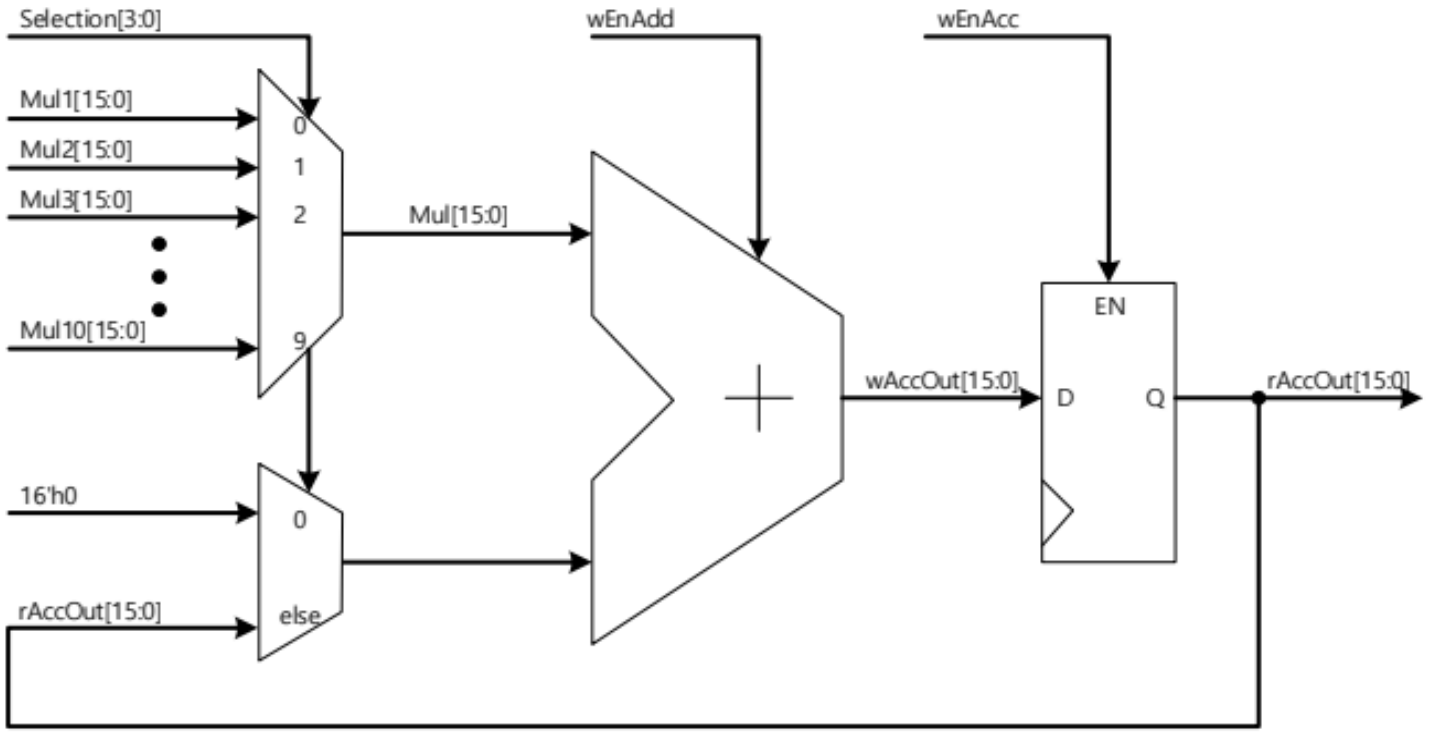



```
assign wDelay0 =  rShifter[ 0] + rShifter[78];
assign wDelay1 =  rShifter[ 1] + rShifter[77];
assign wDelay2 =  rShifter[ 2] + rShifter[76];
assign wDelay3 =  rShifter[ 3] + rShifter[75];
assign wDelay4 =  rShifter[ 4] + rShifter[74];
assign wDelay5 =  rShifter[ 5] + rShifter[73];
assign wDelay6 =  rShifter[ 6] + rShifter[72];
assign wDelay7 =  rShifter[ 7] + rShifter[71];
assign wDelay8 =  rShifter[ 8] + rShifter[70];
assign wDelay9 =  rShifter[ 9] + rShifter[69];
assign wDelay10 = rShifter[10] + rShifter[68];
assign wDelay11 = rShifter[11] + rShifter[67];
assign wDelay12 = rShifter[12] + rShifter[66];
assign wDelay13 = rShifter[13] + rShifter[65];
assign wDelay14 = rShifter[14] + rShifter[64];
assign wDelay15 = rShifter[15] + rShifter[63];
assign wDelay16 = rShifter[16] + rShifter[62];
assign wDelay17 = rShifter[17] + rShifter[61];
assign wDelay18 = rShifter[18] + rShifter[60];
assign wDelay19 = rShifter[19] + rShifter[59];
assign wDelay20 = rShifter[20] + rShifter[58];
assign wDelay21 = rShifter[21] + rShifter[57];
assign wDelay22 = rShifter[22] + rShifter[56];
assign wDelay23 = rShifter[23] + rShifter[55];
assign wDelay24 = rShifter[24] + rShifter[54];
assign wDelay25 = rShifter[25] + rShifter[53];
assign wDelay26 = rShifter[26] + rShifter[52];
assign wDelay27 = rShifter[27] + rShifter[51];
assign wDelay28 = rShifter[28] + rShifter[50];
assign wDelay29 = rShifter[29] + rShifter[49];
assign wDelay30 = rShifter[30] + rShifter[48];
assign wDelay31 = rShifter[31] + rShifter[47];
assign wDelay32 = rShifter[32] + rShifter[46];
assign wDelay33 = rShifter[33] + rShifter[45];
assign wDelay34 = rShifter[34] + rShifter[44];
assign wDelay35 = rShifter[35] + rShifter[43];
assign wDelay36 = rShifter[36] + rShifter[42];
assign wDelay37 = rShifter[37] + rShifter[41];
assign wDelay38 = rShifter[38] + rShifter[40];
assign wDelay39 = rShifter[39];
```

Structure (Block diagram)

Accumulator

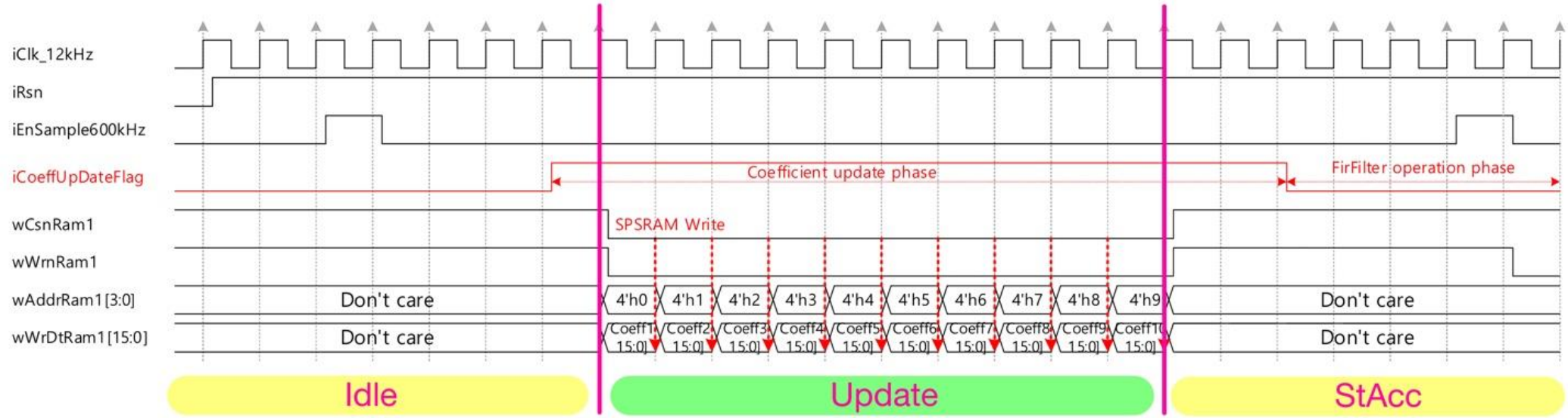
Multiplier & Adder (by 10X1 Mux)




```
// Mux Select
/*************************************************************/
  assign wMul = (iSelection == 4'd0) ? iMul_0 :
                (iSelection == 4'd1) ? iMul_1 :
               (iSelection == 4'd2) ? iMul_2 :
               (iSelection == 4'd3) ? iMul_3 :
               (iSelection == 4'd4) ? iMul_4 :
               (iSelection == 4'd5) ? iMul_5 :
               (iSelection == 4'd6) ? iMul_6 :
               (iSelection == 4'd7) ? iMul_7 :
               (iSelection == 4'd8) ? iMul_8 :
               (iSelection == 4'd9) ? iMul_9 : 16'd0;
/*************************************************************/
// Final output
/*************************************************************/
  always @(posedge iClk12M) begin
  if(iEnAdd)begin
   oAccOut <= wAccOut;
   iSelection <= iSelection + 1'b1;
  end

  else if( (iEnMul == 1'b1) && (iEnAdd == 1'b0) )
  begin
    oAccOut <= oAccOut;
    iSelection <= 4'd0;
  end

  else if(iSelection==4'd10) begin
   iSelection <= 4'd10;
  end

  else
   oAccOut <= oAccOut;

  end
```

Structure (Timing chart)

1. Coefficient update phase



Structure (State Configuration)

1. Coefficient update phase

Under CoeffNum : Correct Address

Over CoeffNum : Ordered Address (by rCnt_0)

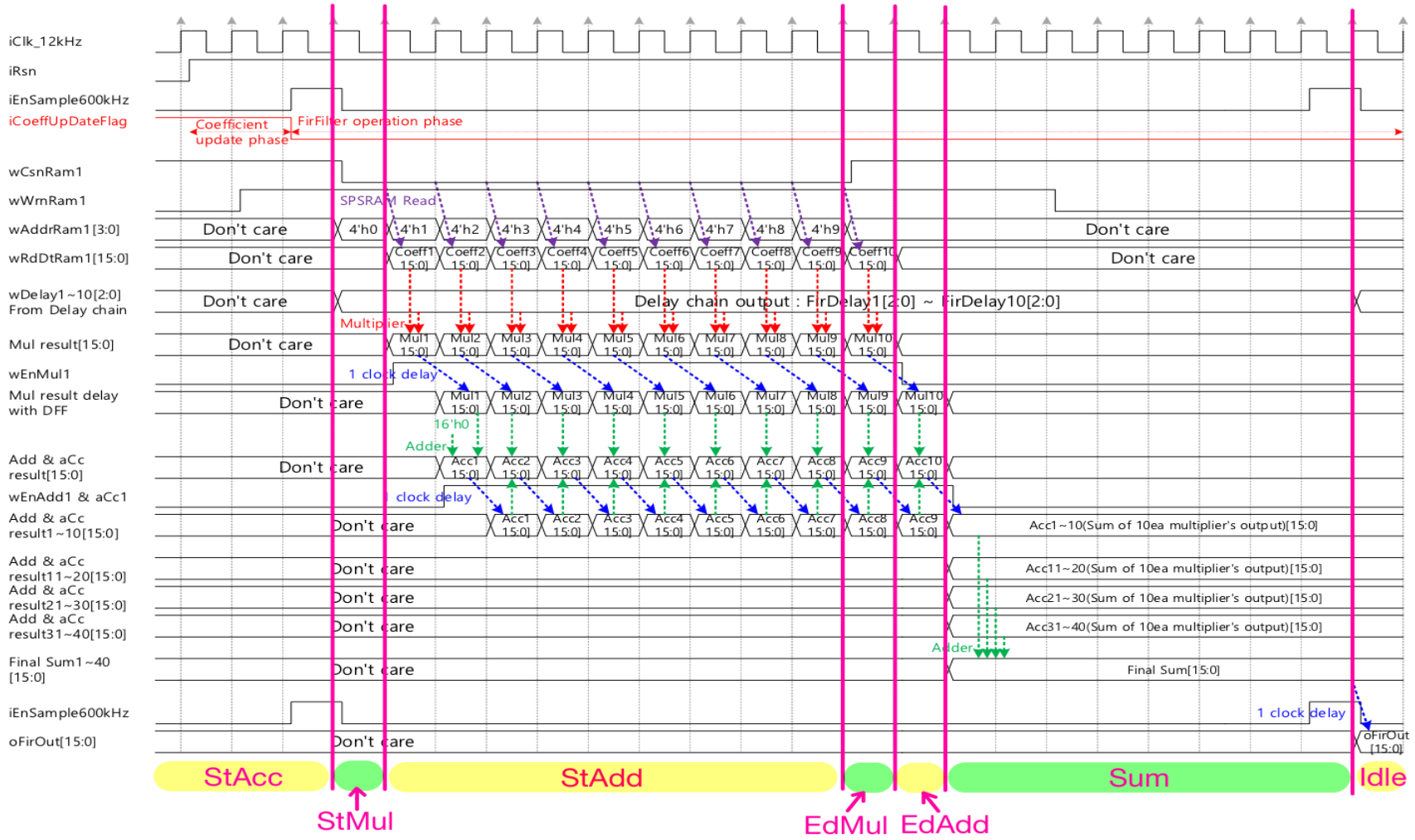
Error : Error Address

```
p_Update :
begin
  if(rCnt_0 <= (rNumOfCoeff - 6'd1) )
  begin
    rAddrRam_0 <= iAddrRam;
    rAddrRam_1 <= iAddrRam;
    rAddrRam_2 <= iAddrRam;
    rAddrRam_3 <= iAddrRam;

    rWrDtRam   <= iWrDtRam;

  end


  else
  begin
      if(rCnt_0 < 6'd9)
      begin
        rAddrRam_0 <= {p_SelRam_0, 4'h0} + rCnt_0;          // rAddrRam[5:0], p_SelRam[1:0], rCnt_0[5:0]
        rAddrRam_1 <= {p_SelRam_0, 4'h0} + rCnt_0;          // Prevent Error
        rAddrRam_2 <= {p_SelRam_0, 4'h0} + rCnt_0;          // Prevent Error
        rAddrRam_3 <= {p_SelRam_0, 4'h0} + rCnt_0;          // Prevent Error

        rWrDtRam   <= 16'h0000;
      end

      else if(rCnt_0 < 6'd19)
      begin
        rAddrRam_0 <= {p_SelRam_1, 4'h0} + (rCnt_0 - 6'd 9);    // Prevent Error
        rAddrRam_1 <= {p_SelRam_1, 4'h0} + (rCnt_0 - 6'd 9);    // rAddrRam[5:0], p_SelRam[1:0], rCnt_0[5:0]
        rAddrRam_2 <= {p_SelRam_1, 4'h0} + (rCnt_0 - 6'd 9);    // Prevent Error
        rAddrRam_3 <= {p_SelRam_1, 4'h0} + (rCnt_0 - 6'd 9);    // Prevent Error

        rWrDtRam   <= 16'h0000;
      end

      else if(rCnt_0 < 6'd29)
      begin
        rAddrRam_0 <= {p_SelRam_2, 4'h0} + (rCnt_0 - 6'd19);    // Prevent Error
        rAddrRam_1 <= {p_SelRam_2, 4'h0} + (rCnt_0 - 6'd19);    // Prevent Error
        rAddrRam_2 <= {p_SelRam_2, 4'h0} + (rCnt_0 - 6'd19);    // rAddrRam[5:0], p_SelRam[1:0], rCnt_0[5:0]
        rAddrRam_3 <= {p_SelRam_2, 4'h0} + (rCnt_0 - 6'd19);    // Prevent Error
```

```
      else if(rCnt_0 < 6'd39)
      begin
        rAddrRam_0 <= {p_SelRam_3, 4'h0} + (rCnt_0 - 6'd29);    // Prevent Error
        rAddrRam_1 <= {p_SelRam_3, 4'h0} + (rCnt_0 - 6'd29);    // Prevent Error
        rAddrRam_2 <= {p_SelRam_3, 4'h0} + (rCnt_0 - 6'd29);    // Prevent Error
        rAddrRam_3 <= {p_SelRam_3, 4'h0} + (rCnt_0 - 6'd29);    // rAddrRam[5:0], p_SelRam[1:0], rCnt_0[5:0]

        rWrDtRam   <= 16'h0000;
      end

      else
      begin
        rAddrRam_0 <= {p_SelRam_0, 4'hF};
        rAddrRam_1 <= {p_SelRam_1, 4'hF};
        rAddrRam_2 <= {p_SelRam_2, 4'hF};
        rAddrRam_3 <= {p_SelRam_3, 4'hF};

        rWrDtRam   <= 16'h0000;
      end

  end

  if(rCnt_0 == 6'd39)
  begin
    rCsnRam <= 1'b1;
    rWrnRam <= 1'b1;

    rCnt_0  <= 6'd39;
  end

  else
  begin
    rCsnRam <= 1'b0;

    rCnt_0  <= rCnt_0 + 6'd1;
  end

end
```

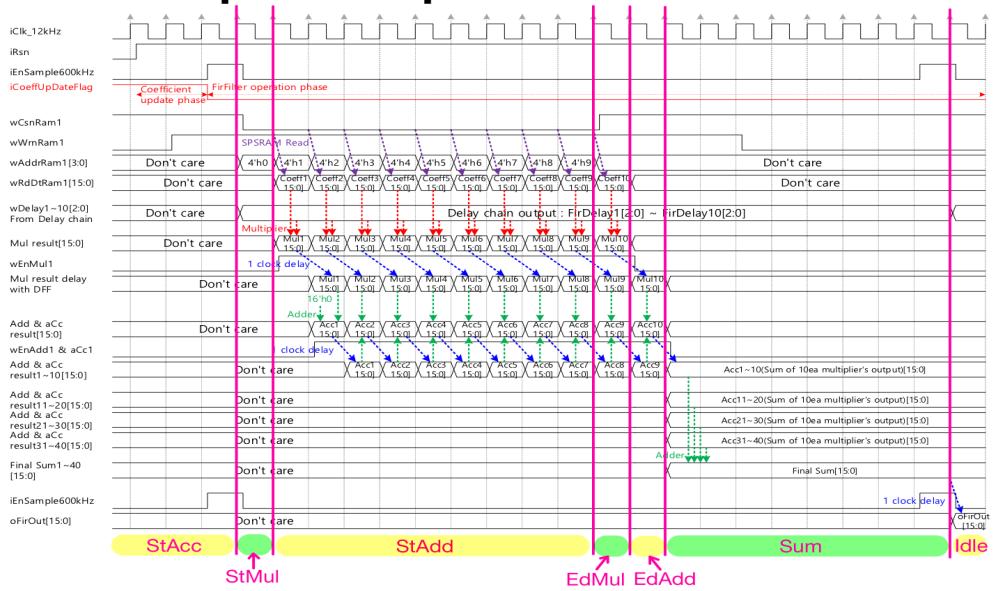
Structure (Timing chart)

2. Filter operation phase



Structure (State Configuration)

2. Filter operation phase



rCnt_0 : Read Coeff from SRAM (10EA)
rCnt_1 : Number of iEnSample600k (80EA)

rEnable : Enable signal for output stability

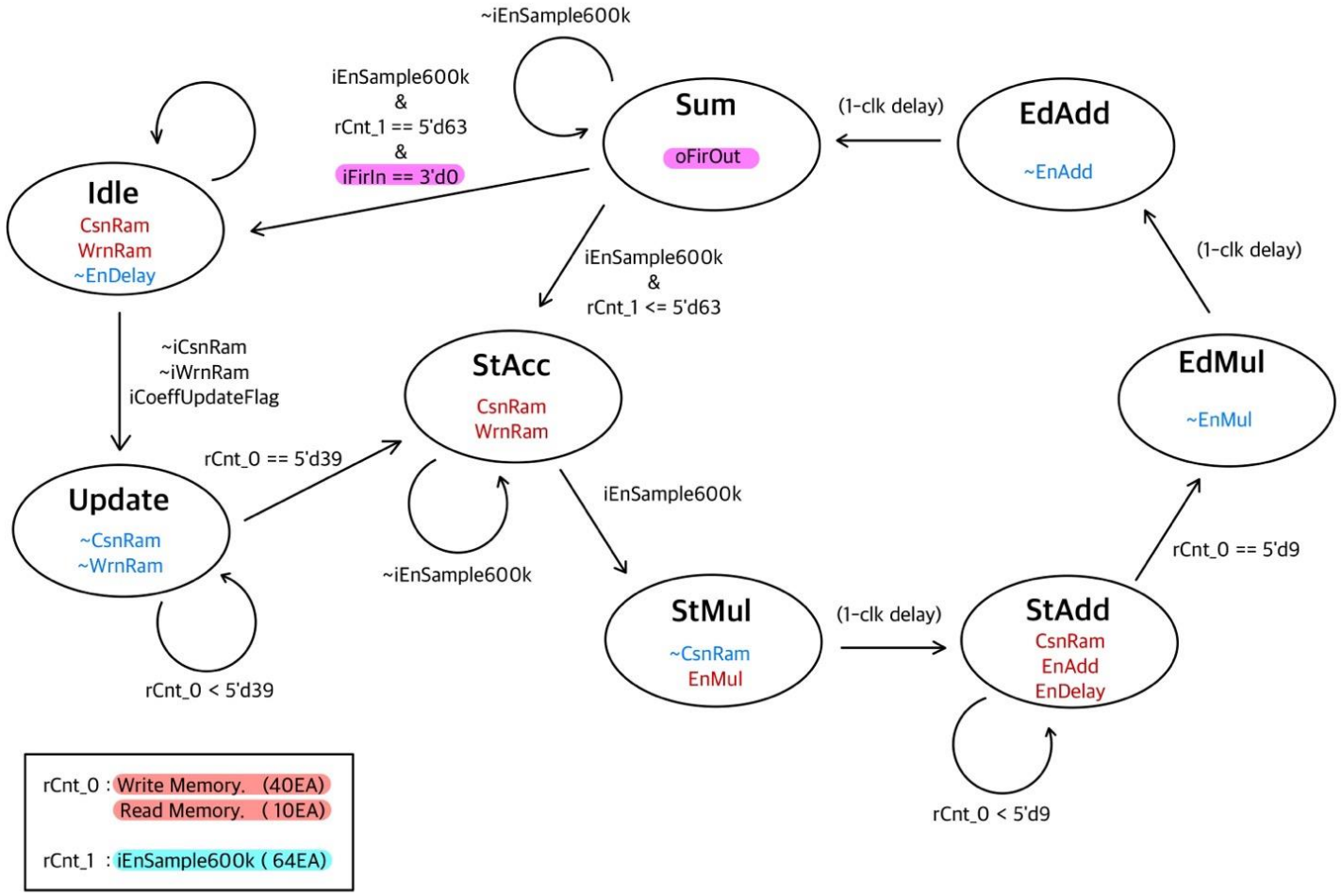
```
p_StAcc :
begin
  if(iEnSample600k == 1'b1)
  begin
    rCsnRam   <= 1'b0;

    rCnt_0    <= rCnt_0 + 6'd1;          // Update Counter_0 for save Adder
    rCnt_1    <= rCnt_1 + 6'd1;          // Update Counter_1

    rEnDelay  <= 1'b1;                   // EnDelay  ON
    rEnMul    <= 1'b0;                   // EnMul    OFF
    rEnAdd    <= 1'b0;                   // EnAdd    OFF
    rEnAcc    <= 1'b0;                   // EnAcc    OFF

    rAddrRam_0 <= 6'h00;
    rAddrRam_1 <= 6'h10;
    rAddrRam_2 <= 6'h20;
    rAddrRam_3 <= 6'h30;
  end

  else if(rBufState == 1'b1)
  begin
    rAddrRam_0 <= rAddrRam_0 + 6'd1;
    rAddrRam_1 <= rAddrRam_1 + 6'd1;
    rAddrRam_2 <= rAddrRam_2 + 6'd1;
    rAddrRam_3 <= rAddrRam_3 + 6'd1;

    rEnDelay  <= 1'b1;                   // EnDelay  ON
    rEnMul    <= 1'b1;                   // EnMul    ON
    rEnAdd    <= 1'b0;                   // EnAdd    OFF
    rEnAcc    <= 1'b0;                   // EnAcc    OFF
  end
```

```
  else
  begin
    rCsnRam   <= 1'b1;

    rCnt_0    <= 6'd0;                   // Reset Counter_0 for save Adder
    rCnt_1    <= rCnt_1;                 // Suspend Counter_1

    rAddrRam_0 <= 6'h05;
    rAddrRam_1 <= 6'h1F;
    rAddrRam_2 <= 6'h2F;
    rAddrRam_3 <= 6'h3F;

    rEnDelay  <= 1'b1;                   // EnDelay  ON
    rEnMul    <= 1'b0;                   // EnMul    OFF
    rEnAdd    <= 1'b0;                   // EnAdd    OFF
    rEnAcc    <= 1'b0;                   // EnAcc    OFF
  end

end
```

```
p_EdMul :
  begin
    rCsnRam   <= 1'b1;
    rWrnRam   <= 1'b1;

    rCnt_0    <= 6'd0;                   // Reset Counter_0

    rEnDelay  <= 1'b1;                   // EnDelay  ON  (Last)
    rEnMul    <= 1'b0;                   // EnMul    OF
    rEnAdd    <= 1'b1;                   // EnAdd    ON  (Last)

  end


p_EdAdd  :
  begin
    rEnDelay  <= 1'b1;                   // EnDelay  ON
    rEnMul    <= 1'b0;                   // EnMul    OFF
    rEnAdd    <= 1'b0;                   // EnAdd    OFF
    rEnAcc    <= 1'b0;                   // EnAcc    OFF

  end


p_Sum    :
  begin
    rEnDelay  <= 1'b1;                   // EnDelay  ON
    rEnMul    <= 1'b0;                   // EnMul    OFF
    rEnAdd    <= 1'b0;                   // EnAdd    OFF
    rEnAcc    <= 1'b0;                   // EnAcc    OFF

    rCnt_0    <= rCnt_0 + 6'd1;

  end
```

Structure (FSM)

Finite State Machine



[000] Idle : Wait for start (iCoeffUpdateFlag)

[001] Update : Update the coefficients SRAM

[010] StAcc : Wait for start (iEnSample600k)

[011] StMul : Start Multiplication

[100] StAcc : Start Addition & Accumulation

[101] EdMul : End Multiplication

[110] EdAdd : End Addition & Accumulation

[111] Sum : Summation & Next state decision

Verification

Pre-simulation #01

(NumOfCoeff : 17 EA iFirIn : +1 / -1 Coefficient : sinc)

Pre-simulation #02

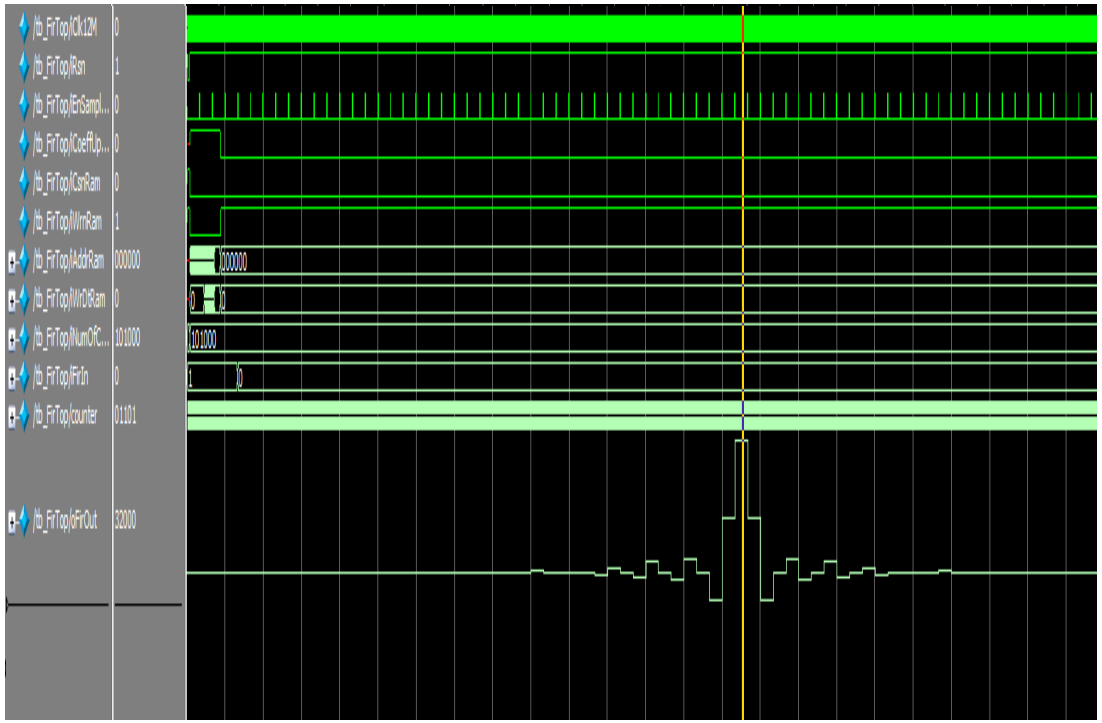
(NumOfCoeff : 17 EA iFirIn : +3 / -3 Coefficient : sinc)

Tools : Quartus Model-Sim

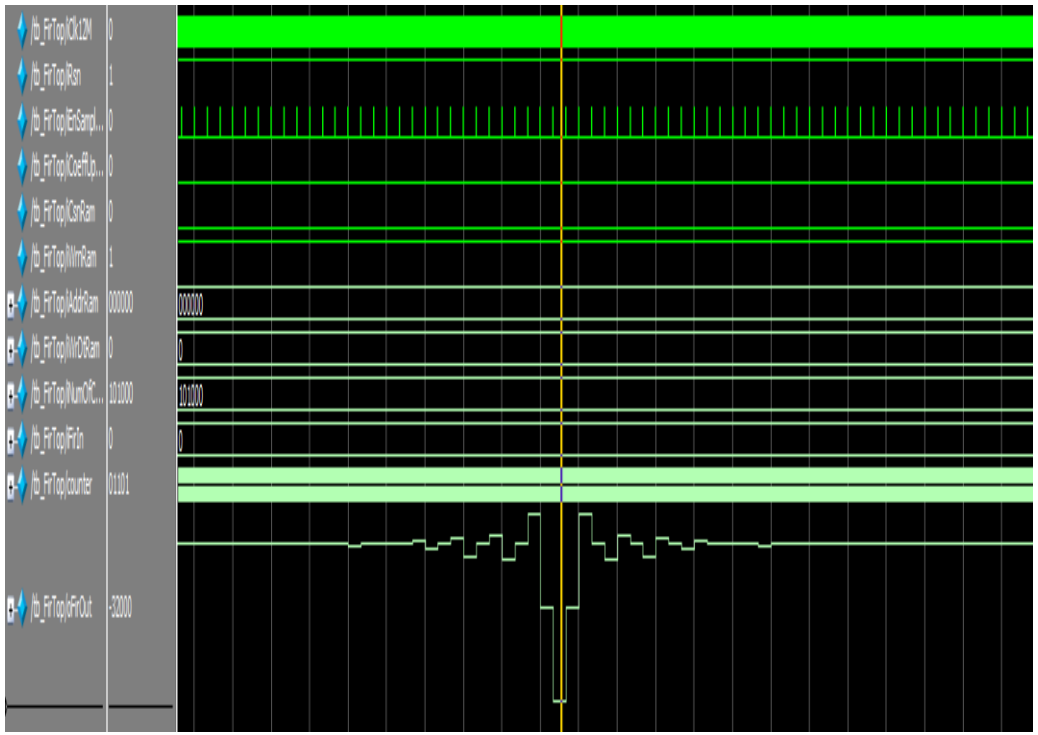
Verification

Pre-simulation #01 (NumOfCoeff : 17 EA)

ⅰ) iFirIn == 3'b001 (Coeff : 0-40, Peak coeff : 32000)



ⅱ) iFirIn == 3'b111 (Coeff : 0-40, Peak coeff : -32000)

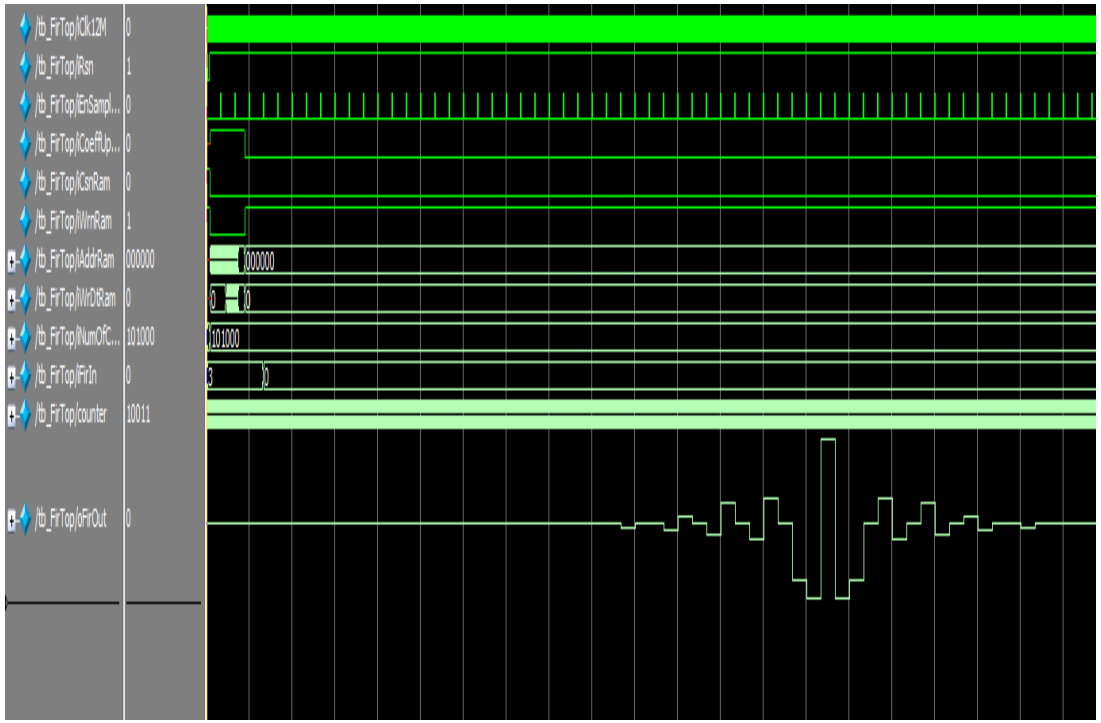


Tools : Quartus Model-Sim

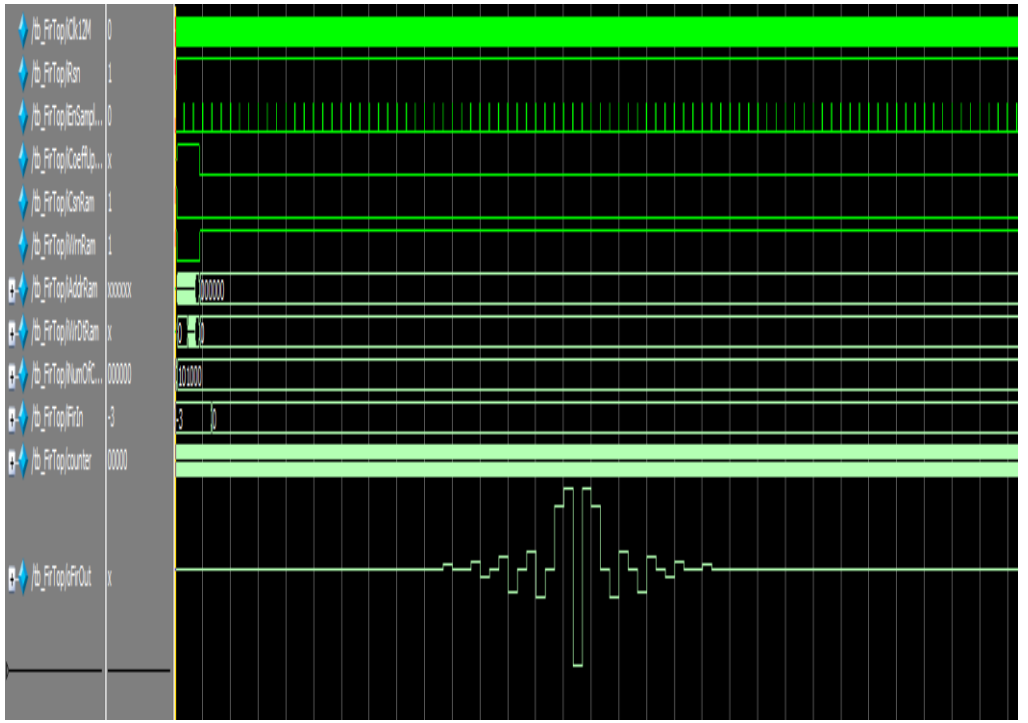
Verification

Pre-simulation #02 (NumOfCoeff : 17 EA)

ⅰ) iFirIn == 3'b011 (Coeff : 0-40, Peak coeff : 96000)



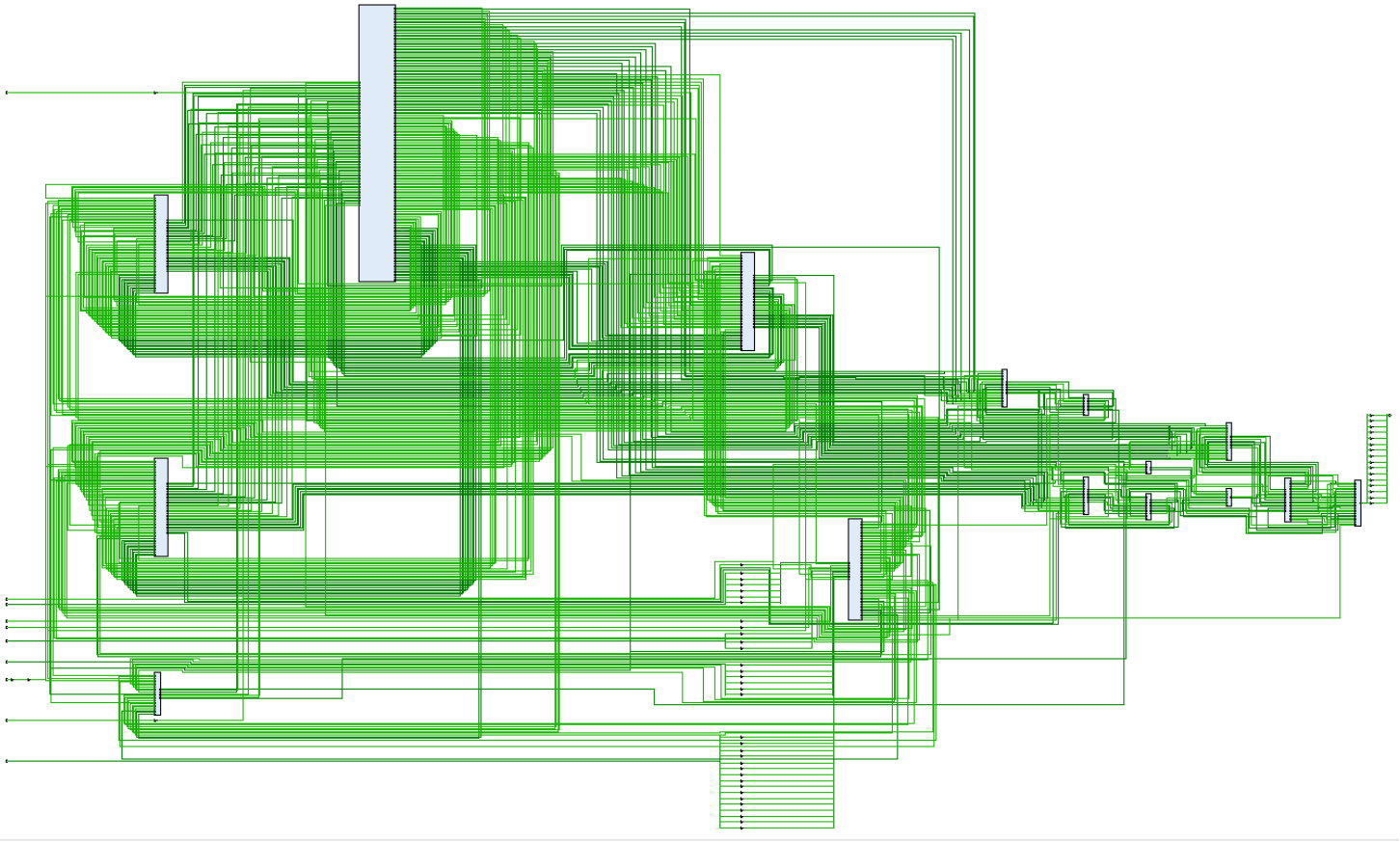
ⅱ) iFirIn == 3'b101 (Coeff : 0-40, Peak coeff : -96000)



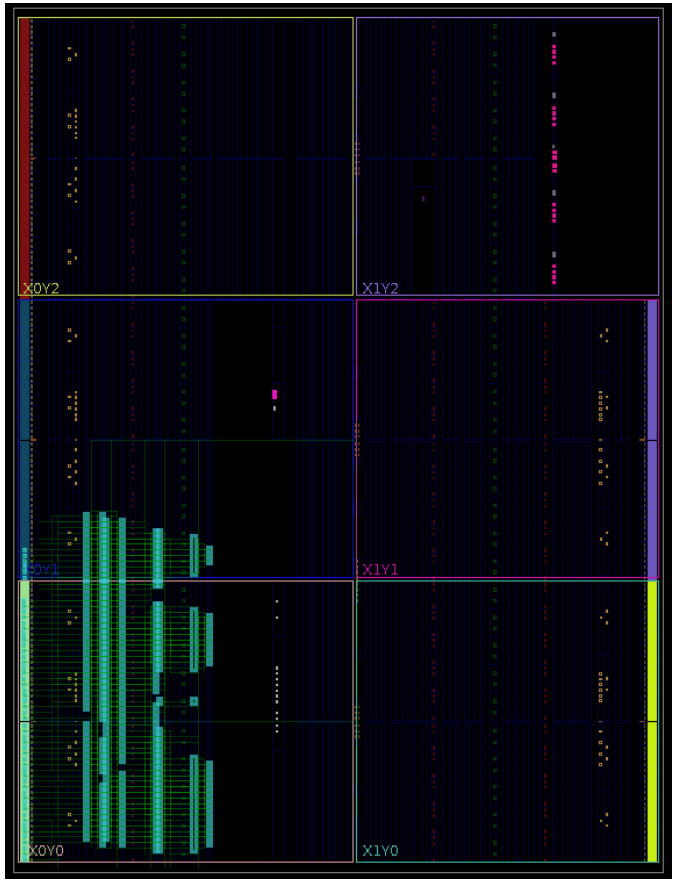
Tools : Quartus Model-Sim

Verification

1) Synthesis



2) Implementation



Tools : Xilinx Vivado

Parts : xc7a35tftg256-1

Verification

Input : Data-set

Post-simulation #01

(NumOfCoeff : 32 EA iFirIn : +1)

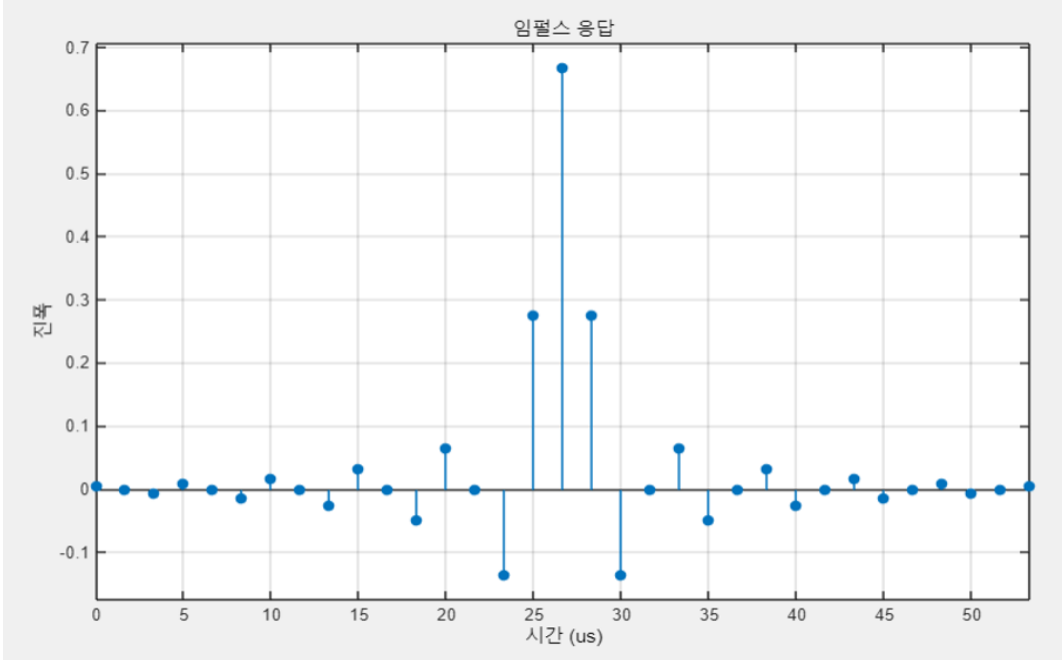
Post-simulation #02

(NumOfCoeff : 32 EA iFirIn : +3)

Tools : Xilinx Vivado Parts : xc7a35tftg256-1

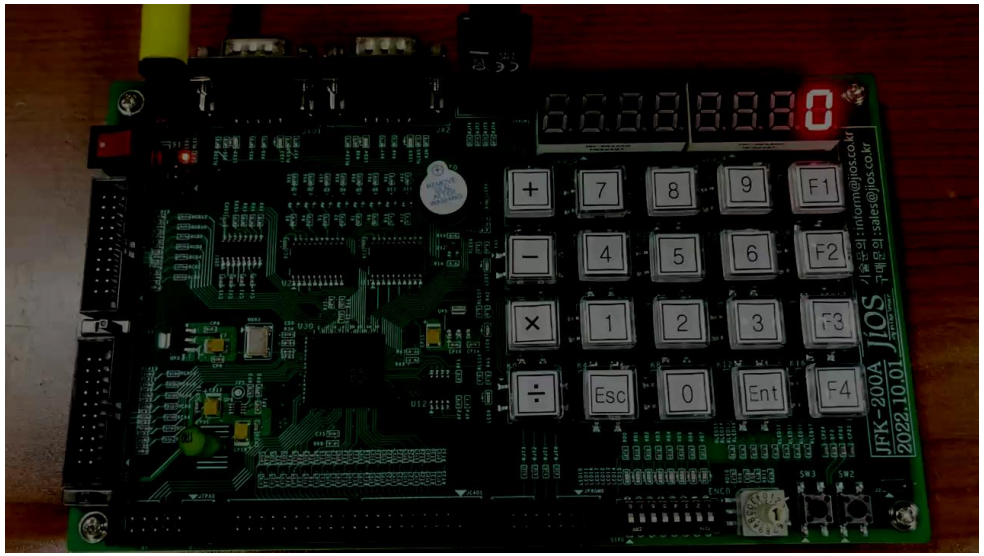
FIR filter with kaiser window

n	Integer	Binary
n-16	3	00_0000_0011
n-15	0	00_0000_0000
n-14	6	11_1111_1010
n-13	6	00_0000_0110
n-12	0	00_0000_0000
n-11	11	11_1111_0101
n-10	13	00_0000_1101
n-9	0	00_0000_0000
n-8	19	11_1110_1101
n-7	24	00_0010_1000
n-6	0	00_0000_0000
n-5	37	11_1110_0101
n-4	48	00_0011_0000
n-3	0	00_0000_0000
n-2	102	11_1001_1010
n-1	206	00_1100_1110
n	500	01_1111_0100

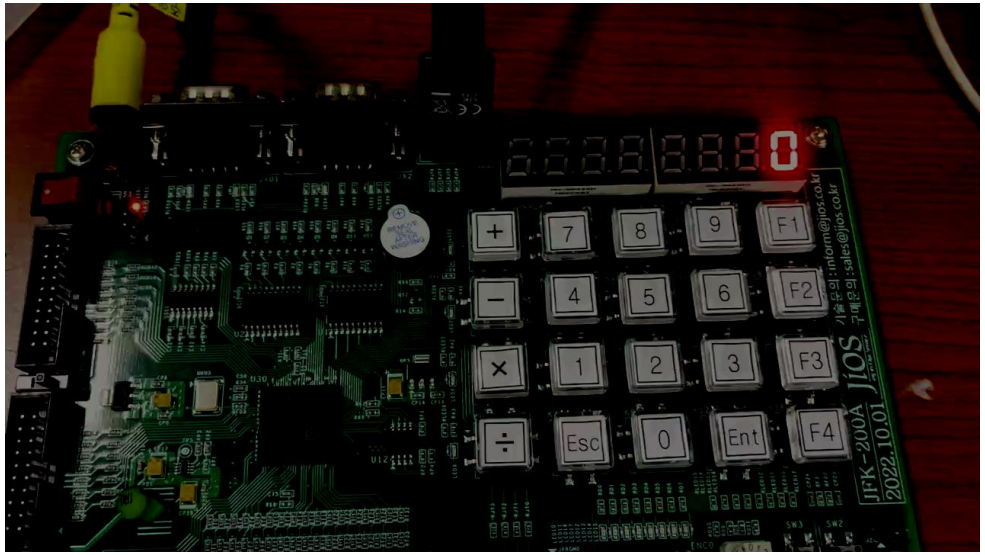


Verification

Post-simulation #01



Post-simulation #02



Tools : Xilinx Vivado

Parts : xc7a35tftg256-1

Conclusion

Problem

Randomly specified address : Adderess of SP-SRAM

Conclusion

Improvement

Randomly specified address => APB Protocol Module

Thank you

Department of Electronic Engineering

20203033 주 동 준　20203023 우 상 욱

숭실대학교 Soongsil University