Design Wave 設計コンテスト 2005

課題：ディジタル FM レシーバ

氏名：稲葉　秀一(INABA Hidekazu)

区分：社会人部門

概略

受信機の IF 信号(455kHz)を取り込み、FPGA を用いてディジタル方式の復調を行った後に、パソコンのサウンドカードで取り込み AFSK 信号を復調する系を作成した。

この系を用いて実際に 430MHz 帯の電波を復調し、設計した FPGA 回路が正しく動作することを確認した。

回路構成

受信機には ICOM 製の IC-R100 を用いた。この受信機には狭帯域 FM を受信するため、モトローラ製の MC-3357 が実装されている。この IC のリミッタアンプ出力を取り出せるよう改造を行っている。

受信機からの IF 信号を A/D 変換し、FPGA に取り込んでいる。A/D 変換には NEC 製のμPC659A を使用している。

FPGA にてクワドラチャ検波による復調を行っている。

復調した信号は FPGA から PWM で出力し、これを RC フィルタに通すことで音声帯域のアナログ信号に変換した。

この信号をパソコンのサウンドカードより取り込み、スピーカを鳴らすことで音声信号を、SV2AGW の作成したソフトウェアで AX.25 での通信をモニタしている。

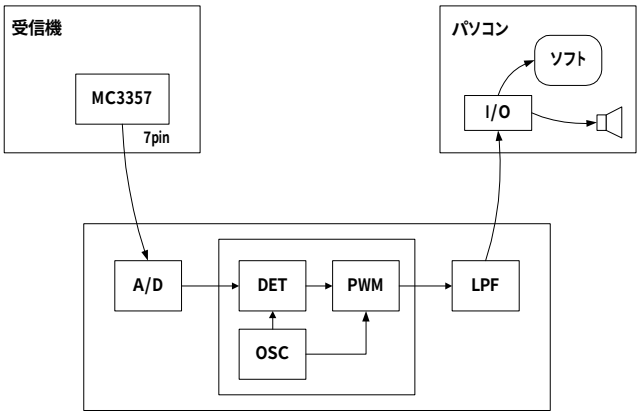


図 1 全体の構成

入力回路

受信機から取り込むIF信号は 455kHzであり、これを16MHzで 8 ビットのサンプリングを行っている。この信号はIF増幅後にリミッタ回路を通っているため、振幅は0.8V_{P-P}程度で一定となっており、A/D変換ICは入力の1V_{P-P}を 00～FFに変換するので、取り込む際に飽和することは無い。

この IC からの出力は符号なし 8 ビットであるため、今回の課題で示されているフォーマットに変換し、FPGA 内の復調処理回路に渡している。

復調回路

復調回路には、CQ 出版の Xilinx 製の XC2S150 が実装されている「Soartan-II 評価キット」を利用した。

クワドラチャ検波を行うため、入力されたデータは FPGA のブロック RAM を利用した回路で遅延させている。この遅延はアナログ回路では同調回路を用い 90 ° であるが、今回の回路では遅延量を増やしている。

これは IF 信号の 455kHz に対して、変調する周波数は数 kHz までなことから、多くの場合には同一方向に位相は変化していることを利用し、乗算回路から得られる値を大きくするためである。

乗算した結果には必要となる成分以外に、高周波成分が含まれていることから、これをローパスフィルタで取り除く必要があるが、回路規模を小さくするため、一定区間の乗算結果を全て加算することで平滑化することで実現している。

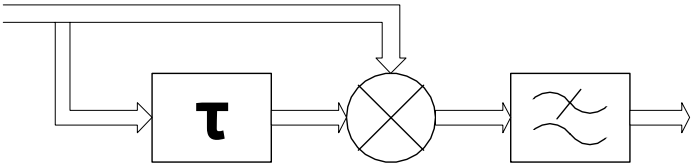
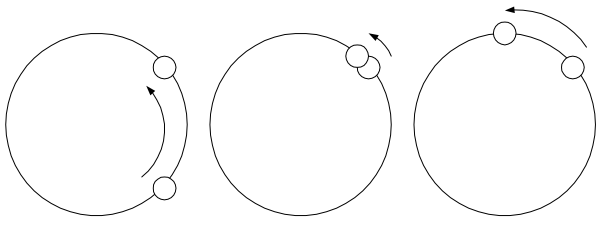


図 2 復調回路の構成



90 ° 後　　変調あり　　複数サイクル後

図 3 位相の変化

出力回路

課題では復調した信号がディジタルなままであるため、これを音声として聞くためには D/A 変換する必要がある。このため検波した信号の値によりパルス幅を変化させる PWM 回路を FPGA に実装し出力とした。

この信号をカットオフ周波数 4kHz とした、抵抗とコンデンサによるパッシブフィルタを通し、DC カットした信号を基板回路の出力とした。

また基板上に実装された LED を、復調したデータにより点灯させ、簡易的にセンタメータおよびデビエーションのモニタとなるようにした。

基板からの出力はインピーダンスが高くスピーカを鳴らすことはできないので、パソコンのサウンドカードのライン入力へ接続し増幅して音として聞くことを可能とした。またパソコンに取り込んで、アマチュア無線を使用して AX.25 で行われている通信をデコードできるようにした。

実機での動作

取り込んだ AFSK の信号は図のようであり、また図のようにデコードされデータとしても取り出すことができた。

写真 1 評価系

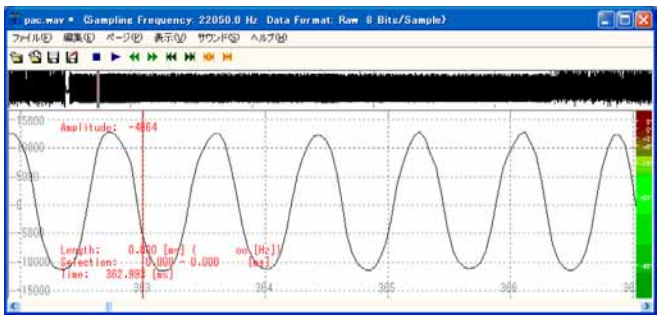


図 1 復調波形

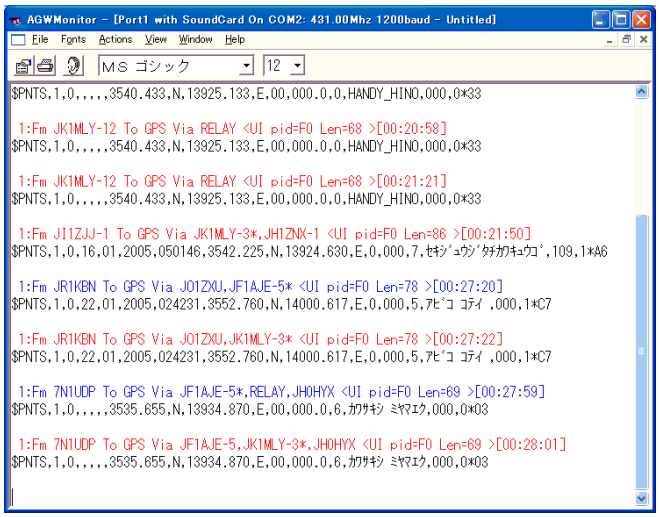


図 1 復調データ

FPGA 回路の規模と遅延

FPGA の設計には Xilinx の WebPACK ISE(6.3.03i)を利用した。基準となる 50 入力の XOR 回路の結果は以下である。

これに対し課題の条件に合わせた回路を合成した結果は以下である。

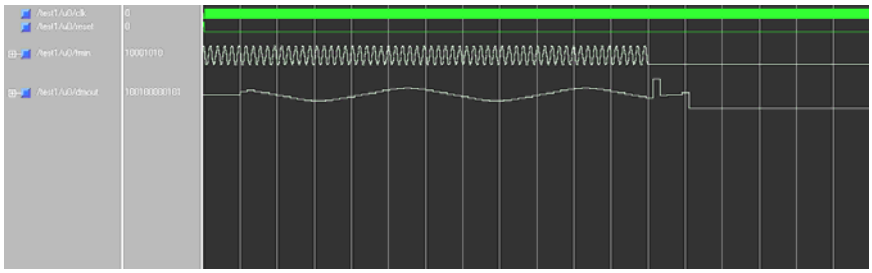
また実機で動作させる全ての回路を含んだ状態の結果は以下である。

ソースコード

課題に示されている条件に従い設計した回路のソースコードをリスト 1,2 に示す。また実機で動作させた回路のソースコードをリスト 3～5 に示す。

シミュレーション

　提供されている FM 変調波のデータを用い、シミュレーションした結果を図に示す。



まとめ

```
--------------------------------------------------------------------------------
-- DW2005 FM reciver for Simuration module
-- Ver 00
-- 19 Dec. 2004
--------------------------------------------------------------------------------
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY RECV is
port (
        CLK     : in std_logic;
        RESET   : in std_logic;
        FMIN    : in std_logic_vector(7 downto 0 );
        DMOUT   : out std_logic_vector(11 downto 0 )
        );
END RECV;

--data format
--FMIN(8,0,t)
-- S.XXXXXXX
-- -1 to 1-(1/2**7)
--DMOUT(12,4,t)
-- SXXXX.XXXXXXX
-- -16 to 16-(1/2**7)

ARCHITECTURE RTL of RECV is

COMPONENT RAMB
port (
        CLK     : in std_logic;
        DIN     : in std_logic_vector(7 downto 0);
        DOUT    : out std_logic_vector(7 downto 0);
        WE              : in std_logic;
        WADR    : in std_logic_vector(9 downto 0);
        RADR    : in std_logic_vector(9 downto 0)
        );
END COMPONENT;

--------------------------------------------------------------------------------
```

```
--
-- Signals
--
signal CUR_S    : std_logic_vector(7 downto 0);         --signed data
signal DLY_S    : std_logic_vector(7 downto 0);         --current / delayed
signal CUR_U    : std_logic_vector(7 downto 0);         --unsigned data
signal DLY_U    : std_logic_vector(7 downto 0);         --current / delayed
signal MIX      : std_logic_vector(15 downto 0);                --mixing
signal MIX_R    : std_logic_vector(8 downto 0);         --round
signal xMIX_R   : std_logic_vector(8 downto 0);         --invert
signal SIGN0    : std_logic_vector(1 downto 0);         --sign
signal SIGN1    : std_logic_vector(1 downto 0);         --1D
signal R_ADR    : std_logic_vector(9 downto 0);         --write address
signal W_ADR    : std_logic_vector(9 downto 0);         --read address
signal DAT_R    : std_logic_vector(9 downto 0);         --raw data
signal CNT              : std_logic_vector(5 downto 0);         --counter
signal SUM              : std_logic_vector(11 downto 0);        --sum up
signal REG              : std_logic_vector(11 downto 0);        --data register
signal EN               : std_logic;                                            --enable

-------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------
--
-- Start of circuit description
--
BEGIN

        process(RESET, CLK)
        begin
                if(RESET = '1') then
                        CUR_S <= (others => '0');
                elsif (CLK'event and CLK='1') then
                        CUR_S <= FMIN;
                end if;
        end process;

--
-- delay line
--
        process(RESET, CLK)
        begin
```

```
		if(RESET = '1') then
				R_ADR <= (others => '0');
		elsif (CLK'event and CLK='1') then
--			if(R_ADR < "0110100011") then	--16/1*26.25=420 000-1A3
--			if(R_ADR < "0000100011") then	--16/1*2.25=68 000-23
--			if(R_ADR < "0001100011") then	--16/1*6.25=100 000-63
			if(R_ADR < "0001000011") then	--16/1*4.25=100 000-43
				R_ADR <= R_ADR + "0000000001";
			else
				R_ADR <= (others => '0');
			end if;
		end if;
	end process;

	process(RESET, CLK)
	begin
		if(RESET = '1') then
			W_ADR <= (others => '0');
		elsif (CLK'event and CLK='1') then
			W_ADR <= R_ADR;
		end if;
	end process;

U21 : RAMB
	port map (
		CLK	=> CLK,
		DIN	=> CUR_S,
		DOUT	=> DLY_S,
		WE		=> '1',
		WADR	=> W_ADR,
		RADR	=> R_ADR
	);

--
-- mixer(unsigned)
--
	process(RESET, CLK)
	begin
		if(RESET = '1') then
				CUR_U <= (others => '0');
		elsif (CLK'event and CLK='1') then
```

```
                if(CUR_S(7)='0') then
                        CUR_U <= "0" &  CUR_S(6 downto 0);
                else
                        CUR_U <= "0" & (CUR_S(6 downto 0) xor "1111111");
                end if;
        end if;
    end process;

    process(RESET, CLK)
    begin
        if(RESET = '1') then
                        DLY_U <= (others => '0');
        elsif (CLK'event and CLK='1') then
                if(DLY_S(7)='0') then
                        DLY_U <= "0" &  DLY_S(6 downto 0);
                else
                        DLY_U <= "0" & (DLY_S(6 downto 0) xor "1111111");
                end if;
        end if;
    end process;

    process(RESET, CLK)
    begin
        if(RESET = '1') then
                MIX <= (others => '0');
        elsif (CLK'event and CLK='1') then
                MIX <= CUR_U * DLY_U;
        end if;
    end process;

--
-- round(signed)
--
--       MIX_R <= MIX(10 downto 2);
         MIX_R <= MIX(13 downto 5);
        xMIX_R <= MIX_R xor "111111111";

    process(RESET, CLK)
    begin
        if(RESET = '1') then
                SIGN0<= (others => '0');
```

```
                SIGN1<= (others => '0');
        elsif (CLK'event and CLK='1') then
                SIGN0 <= CUR_S(7) & DLY_S(7);
                SIGN1 <= SIGN0;
        end if;
    end process;

    process(RESET, CLK)
    begin
        if(RESET = '1') then
                DAT_R <= (others => '0');
        elsif (CLK'event and CLK='1') then
                if(MIX_R = "0000000000") then
                DAT_R <= (others => '0');
                else
                case SIGN1 is
                        when "01"   => DAT_R <= "1" & xMIX_R;
                        when "10"   => DAT_R <= "1" & xMIX_R;
                        when others => DAT_R <= "0" &  MIX_R;
                end case;
                end if;
        end if;
    end process;

--
-- average
--
    process(RESET, CLK)
    begin
        if(RESET = '1') then
                CNT <= (others => '0');
        elsif (CLK'event and CLK='1') then
                if(EN = '1') then
                CNT <= (others => '0');
                else
                CNT <= CNT + "000001";
                end if;
        end if;
    end process;

    EN <= '1' when (CNT >= "001111") else '0';      --16/1=16 00-0F
```

```
	process(RESET, CLK)
	begin
		if(RESET = '1') then
				SUM <= (others => '0');
				REG <= (others => '0');
		elsif (CLK'event and CLK='1') then
			if(EN = '1') then
				SUM <= (others => '0');
				REG <= SUM;
			else
				SUM <= SUM + (DAT_R(9) & DAT_R(9) & DAT_R);
			end if;

		end if;

	end process;

	DMOUT <= REG;

END RTL ;
```

```
----------------------------------------------------------------------------
-- DW2005 FM reciver for Spartan-2 Memory module
-- Ver 00 memory
-- 19 Dec. 2004
----------------------------------------------------------------------------
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
LIBRARY UNISIM;
USE UNISIM.Vcomponents.ALL;

ENTITY RAMB is
port (
        CLK     : in std_logic;
        DIN     : in std_logic_vector(7 downto 0);
        DOUT    : out std_logic_vector(7 downto 0);
        WE              : in std_logic;
        WADR    : in std_logic_vector(9 downto 0);
        RADR    : in std_logic_vector(9 downto 0)
        );
end RAMB;

ARCHITECTURE RTL of RAMB is

COMPONENT RAMB4_S4_S4
port (
        DOA     : out STD_LOGIC_VECTOR (3 downto 0);
        DOB     : out STD_LOGIC_VECTOR (3 downto 0);
        ADDRA   : in STD_LOGIC_VECTOR (9 downto 0);
        ADDRB   : in STD_LOGIC_VECTOR (9 downto 0);
        CLKA    : in STD_ULOGIC;
        CLKB    : in STD_ULOGIC;
        DIA     : in STD_LOGIC_VECTOR (3 downto 0);
        DIB     : in STD_LOGIC_VECTOR (3 downto 0);
        ENA     : in STD_ULOGIC;
        ENB     : in STD_ULOGIC;
        RSTA    : in STD_ULOGIC;
        RSTB    : in STD_ULOGIC;
```

```
        WEA     : in STD_ULOGIC;
        WEB     : in STD_ULOGIC
        );
END COMPONENT;

-------------------------------------------------------------------------------
--
-- Signals
--
signal DIH      : std_logic_vector(3 downto 0 );
signal DIL      : std_logic_vector(3 downto 0 );
signal DOH      : std_logic_vector(3 downto 0 );
signal DOL      : std_logic_vector(3 downto 0 );

-------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------
--
-- Start of circuit description
--
BEGIN

        DIH     <= DIN(7 downto 4);
        DIL     <= DIN(3 downto 0);
        DOUT    <= DOH & DOL;

U211 : RAMB4_S4_S4
        port map (
                DOA     => open,
                DOB     => DOH,
                ADDRA   => WADR,
                ADDRB   => RADR,
                CLKA    => CLK,
                CLKB    => CLK,
                DIA     => DIH,
                DIB     => "0000",
                ENA     => '1',
                ENB     => '1',
                RSTA    => '0',
                RSTB    => '0',
                WEA     => WE,
                WEB     => '0'
```

```
        );

U212 : RAMB4_S4_S4
        port map (
                DOA     => open,
                DOB     => DOL,
                ADDRA   => WADR,
                ADDRB   => RADR,
                CLKA    => CLK,
                CLKB    => CLK,
                DIA     => DIL,
                DIB     => "0000",
                ENA     => '1',
                ENB     => '1',
                RSTA    => '0',
                RSTB    => '0',
                WEA     => WE,
                WEB     => '0'
        );

END RTL ;
```

```
--------------------------------------------------------------------------------

-- DW2005 FM reciver for Spartan-2 Top module

-- Ver 00

-- 19 Dec. 2004

--------------------------------------------------------------------------------

LIBRARY IEEE;

USE IEEE.STD_LOGIC_1164.ALL;

USE IEEE.STD_LOGIC_UNSIGNED.ALL;

LIBRARY UNISIM;

USE UNISIM.Vcomponents.ALL;



ENTITY TOP is

port (
        SYSCLK  : in std_logic;                                 --32MHz
        xRESET  : in std_logic;                                 --SW1
        FMIN    : in std_logic_vector(7 downto 0);      --for A/D
        CLKOUT  : out std_logic;                                --16MHz
        LED_S4  : out std_logic;                                --LED5-12
        LED5    : out std_logic;
        LED6    : out std_logic;
        LED7    : out std_logic;
        LED8    : out std_logic;
        LED9    : out std_logic;
        LED10   : out std_logic;
        LED11   : out std_logic;
        LED12   : out std_logic;
        PWMOUT  : out std_logic;                                --cut off 4kHz
        DMOUT   : out std_logic_vector(11 downto 0)     --for D/A
```

```
        );
END TOP;



ARCHITECTURE RTL of TOP is



COMPONENT RECV

port (

        CLK     : in std_logic;
        RESET   : in std_logic;
        FMIN    : in std_logic_vector(7 downto 0);
                --FMIN(8,0,t)

                -- S.XXXXXXX

                -- -1 to 1-(1/2**7)

        DMOUT   : out std_logic_vector(11 downto 0)
                --DMOUT(12,4,t)

                -- SXXXX.XXXXXXX

                -- -16 to 16-(1/2**7)

        );

END COMPONENT;



COMPONENT PWM

port (

        CLK     : in std_logic;
        RESET   : in std_logic;
```

```
        DATIN   : in std_logic_vector(11 downto 0);
        PWMOUT  : out STD_LOGIC
        );
END COMPONENT;

------------------------------------------------------------------------------
--
-- Signals
--
signal iCLK     : std_logic;
signal CLK              : std_logic;
signal RESET    : std_logic;

signal iFMIN    : std_logic_vector(7 downto 0);
signal iDMOUT   : std_logic_vector(11 downto 0);

------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------
--
-- Start of circuit description
--
```

```
BEGIN

        RESET   <= not xRESET;

--center meter
        LED_S4  <= '0';

        LED12   <= not iDMOUT(11) and     iDMOUT(10) and     iDMOUT(9);
        LED11   <= not iDMOUT(11) and     iDMOUT(10) and not iDMOUT(9);
        LED10   <= not iDMOUT(11) and not iDMOUT(10) and     iDMOUT(9);
        LED9    <= not iDMOUT(11) and not iDMOUT(10) and not iDMOUT(9);
        LED8    <=     iDMOUT(11) and     iDMOUT(10) and     iDMOUT(9);
        LED7    <=     iDMOUT(11) and     iDMOUT(10) and not iDMOUT(9);
        LED6    <=     iDMOUT(11) and not iDMOUT(10) and     iDMOUT(9);
        LED5    <=     iDMOUT(11) and not iDMOUT(10) and not iDMOUT(9);

--clock generator
        process(RESET, SYSCLK)
        begin
                if(RESET = '1') then
                        iCLK <= '0';
```

```
        elsif (SYSCLK'event and SYSCLK='1') then
                iCLK <= not iCLK;
        end if;
    end process;

U1 : BUFG
    port map(
        I => iCLK,
        O => CLK
    );

--A/D converter
    process(RESET, CLK)
    begin
        if(RESET = '1') then
                iFMIN <= (others => '0');
        elsif (CLK'event and CLK='0') then
                iFMIN <= (not FMIN(7)) & FMIN(6 downto 0);
        end if;
    end process;
```

```
        CLKOUT <= iCLK;


--FM receiver
U2 : RECV
        port map(
                CLK     => CLK,
                RESET   => RESET,
                FMIN    => iFMIN,
                DMOUT   => iDMOUT
        );

        DMOUT <= iDMOUT;


--D/A converter
U3 : PWM
        port map(
                CLK     => CLK,
                RESET   => RESET,
                DATIN   => iDMOUT,
                PWMOUT  => PWMOUT
```

```
        );


END RTL ;
```

```
--------------------------------------------------------------------------------
-- DW2005 FM reciver for Spartan-3 RECV module
-- Ver 00
-- 19 Dec. 2004
--------------------------------------------------------------------------------
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY RECV is
port (
        CLK     : in std_logic;
        RESET   : in std_logic;
        FMIN    : in std_logic_vector(7 downto 0 );
        DMOUT   : out std_logic_vector(11 downto 0 )
        );
END RECV;

--data format
--FMIN(8,0,t)
-- S.XXXXXXX
-- -1 to 1-(1/2**7)
--DMOUT(12,4,t)
-- SXXXX.XXXXXXX
-- -16 to 16-(1/2**7)

ARCHITECTURE RTL of RECV is

COMPONENT RAMB
port (
        CLK     : in std_logic;
        DIN     : in std_logic_vector(7 downto 0);
        DOUT    : out std_logic_vector(7 downto 0);
        WE              : in std_logic;
        WADR    : in std_logic_vector(9 downto 0);
        RADR    : in std_logic_vector(9 downto 0)
        );
END COMPONENT;
```

```
--------------------------------------------------------------------------------
--
-- Signals
--
signal CUR_S    : std_logic_vector(7 downto 0);         --signed data
signal DLY_S    : std_logic_vector(7 downto 0);         --current / delayed
signal CUR_U    : std_logic_vector(7 downto 0);         --unsigned data
signal DLY_U    : std_logic_vector(7 downto 0);         --current / delayed
signal MIX      : std_logic_vector(15 downto 0);                --mixing
signal MIX_R    : std_logic_vector(8 downto 0);         --round
signal xMIX_R   : std_logic_vector(8 downto 0);         --invert
signal SIGN0    : std_logic_vector(1 downto 0);         --sign
signal SIGN1    : std_logic_vector(1 downto 0);         --1D
signal R_ADR    : std_logic_vector(9 downto 0);         --write address
signal W_ADR    : std_logic_vector(9 downto 0);         --read address
signal DAT_R    : std_logic_vector(9 downto 0);         --raw data
signal CNT              : std_logic_vector(5 downto 0);         --counter
signal SUM              : std_logic_vector(11 downto 0);        --sum up
signal REG              : std_logic_vector(11 downto 0);        --data register
signal EN               : std_logic;                                            --enable

--------------------------------------------------------------------------------------------------
------------------------------------------------------------------
--
-- Start of circuit description
--
BEGIN

        process(RESET, CLK)
        begin
                if(RESET = '1') then
                        CUR_S <= (others => '0');
                elsif (CLK'event and CLK='1') then
                        CUR_S <= FMIN;
                end if;
        end process;

--
-- delay line
--
        process(RESET, CLK)
```

```
	begin
		if(RESET = '1') then
				R_ADR <= (others => '0');
		elsif (CLK'event and CLK='1') then
			if(R_ADR < "1110011010") then   --16000/455*26.25=923 000-39A
				R_ADR <= R_ADR + "0000000001";
			else
				R_ADR <= (others => '0');
			end if;
		end if;
	end process;

	process(RESET, CLK)
	begin
		if(RESET = '1') then
			W_ADR <= (others => '0');
		elsif (CLK'event and CLK='1') then
			W_ADR <= R_ADR;
		end if;
	end process;

U21 : RAMB
	port map (
		CLK	=> CLK,
		DIN	=> CUR_S,
		DOUT	=> DLY_S,
		WE		=> '1',
		WADR	=> W_ADR,
		RADR	=> R_ADR
	);

--
-- mixer(unsigned)
--
	process(RESET, CLK)
	begin
		if(RESET = '1') then
				CUR_U <= (others => '0');
		elsif (CLK'event and CLK='1') then
			if(CUR_S(7)='0') then
				CUR_U <= "0" &  CUR_S(6 downto 0);
```

```
                else
                        CUR_U <= "0" & (CUR_S(6 downto 0) xor "1111111");
                end if;
        end if;
    end process;

    process(RESET, CLK)
    begin
        if(RESET = '1') then
                        DLY_U <= (others => '0');
        elsif (CLK'event and CLK='1') then
                if(DLY_S(7)='0') then
                        DLY_U <= "0" &  DLY_S(6 downto 0);
                else
                        DLY_U <= "0" & (DLY_S(6 downto 0) xor "1111111");
                end if;
        end if;
    end process;

    process(RESET, CLK)
    begin
        if(RESET = '1') then
                MIX <= (others => '0');
        elsif (CLK'event and CLK='1') then
                MIX <= CUR_U * DLY_U;
        end if;
    end process;

--
-- round(signed)
--
     MIX_R <= MIX(10 downto 2);
    xMIX_R <= MIX_R xor "111111111";

    process(RESET, CLK)
    begin
        if(RESET = '1') then
                SIGN0<= (others => '0');
                SIGN1<= (others => '0');
        elsif (CLK'event and CLK='1') then
                SIGN0 <= CUR_S(7) & DLY_S(7);
```

```
                SIGN1 <= SIGN0;
        end if;
    end process;

    process(RESET, CLK)
    begin
        if(RESET = '1') then
                DAT_R <= (others => '0');
        elsif (CLK'event and CLK='1') then
                if(MIX_R = "0000000000") then
                        DAT_R <= (others => '0');
                else
                        case SIGN1 is
                                when "01"   => DAT_R <= "1" & xMIX_R;
                                when "10"   => DAT_R <= "1" & xMIX_R;
                                when others => DAT_R <= "0" &  MIX_R;
                        end case;
                end if;
        end if;
    end process;

--
-- average
--
    process(RESET, CLK)
    begin
        if(RESET = '1') then
                CNT <= (others => '0');
        elsif (CLK'event and CLK='1') then
                if(EN = '1') then
                        CNT <= (others => '0');
                else
                        CNT <= CNT + "000001";
                end if;
        end if;
    end process;

    EN <= '1' when (CNT >= "100011") else '0';      --16000/455=35 00-23

    process(RESET, CLK)
    begin
```

```
            if(RESET = '1') then
                        SUM <= (others => '0');
                        REG <= (others => '0');
            elsif (CLK'event and CLK='1') then
                    if(EN = '1') then
                            SUM <= (others => '0');
                            REG <= SUM;
                    else
                            SUM <= SUM + (DAT_R(9) & DAT_R(9) & DAT_R);
                    end if;

            end if;

    end process;

    DMOUT <= REG;

END RTL ;
```

```
LIBRARY IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY PWM IS
	PORT(
		CLK		: IN STD_LOGIC ;
		RESET		: IN STD_LOGIC ;
		DATIN		: IN std_logic_vector(11 downto 0);
		PWMOUT		: OUT STD_LOGIC
	);
END PWM;


ARCHITECTURE RTL of PWM is

--
	SIGNAL cnt_d1	: STD_LOGIC_VECTOR(2 DOWNTO 0);
	SIGNAL cnt_d2	: STD_LOGIC_VECTOR(2 DOWNTO 0);
	SIGNAL rc_d1	: STD_LOGIC;
	SIGNAL rc_d2	: STD_LOGIC;
	SIGNAL width	: STD_LOGIC_VECTOR(5 DOWNTO 0);
	SIGNAL pulse		: STD_LOGIC;

begin

--
	process(RESET, CLK) begin
		if(RESET = '1') then
				width <= (others => '0');
		elsif(CLK'event and CLK = '1') then
			if((rc_d1 = '1') and (rc_d2 = '1'))then
				width <= (not DATIN(11)) & DATIN(10 downto 6);
			end if;
		end if;
	end process;

-- counter (16M/16 => 1M)
	process(RESET, CLK) begin
		if(RESET = '1') then
```

```
                cnt_d1 <= (others => '0');
        elsif(CLK'event and CLK = '1') then
                cnt_d1 <= cnt_d1 + 1;
        end if;
    end process;

    rc_d1 <= '1' when (cnt_d1 = "111") else '0';

-- counter (250k/8 => 64k)
    process(RESET, CLK) begin
        if(RESET = '1') then
                cnt_d2 <= (others => '0');
        elsif(CLK'event and CLK = '1') then
                if(rc_d1 = '1') then
                        cnt_d2 <= cnt_d2 + 1;
                end if;
        end if;
    end process;

    rc_d2 <= '1' when (cnt_d2 = "111") else '0';

--
    process(RESET, CLK) begin
        if(RESET = '1') then
                        pulse <= '0';
        elsif(CLK'event and CLK = '1') then
                if((rc_d1 = '1') and (rc_d2 = '1'))then
                        pulse <= '1';
                elsif(width = (cnt_d2 & cnt_d1)) then
                        pulse <= '0';
                end if;
        end if;
    end process;

-- PWM
    PWMOUT <= pulse;

end RTL;
```