FPGA based Frequency/Phase Shift Keying Modulation

Abstract :

Binary FSK(Frequency Shift Keying) and PSK(Phase Shift keying) are extensively used in telecommunication for digital data transmission . FSK/PSK modulator circuits usually employ analogue components such as VCO (voltage control oscillator) , RF Mixer and Level-shifter with limited frequency range and whose performance drift with variation in environmental temperature and duty cycle. An FPGA based all digital implementation ensures constant optimum performance and re-configurability .

This article discusses practical application of a combined Binary FSK and PSK modulator it highlights how embedded resources can be used to implement an all-digital FSK / PSK modulator , which modulates serial data transmission of a UART(Universal Asynchronous Receiver and Transmitter). In this application the modulation scheme targets Altera's Cyclone-IV FPGA populated on a DE2-115 development board from Terasic inc and a HSMC (High Speed Mezzaine Card). System block diagram shows how a LUT based NCO is used to generate the sine wave with varying frequency and then with varying phase. Extensive testing and its results are discussed to verify the proposed modulation.

Frequency Shift Keying Modulation

Frequency-shift keying (FSK) is a frequency modulation scheme in which digital information is transmitted through discrete frequency changes of a carrier wave. The simplest FSK is binary *FSK* (BFSK). BFSK literally implies using a pair of discrete frequencies to transmit binary (0s and 1s) information. With this scheme, the "1" is called the mark frequency and the "0" is called the space frequency. The time domain of an FSK modulated carrier is illustrated in the figure 1 below.

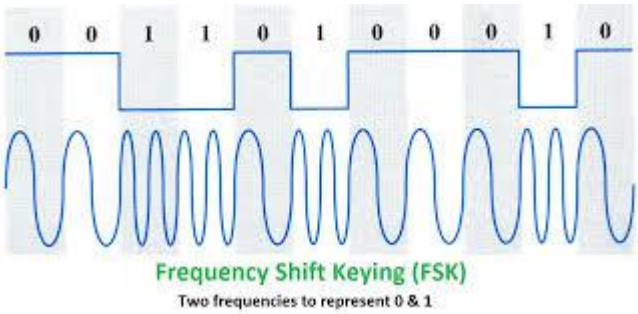
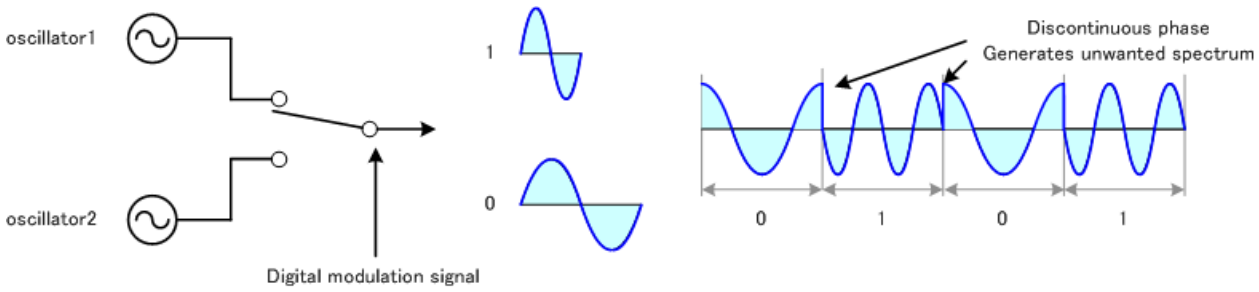


Figure 1

The figure 2 below is the system for switching the transmitter according to the modulating signal level. If the switching timing in the synchronization and modulating signal of the two oscillators is not good, the continuity of the phase between bits cannot be maintained as shown in the figure, resulting in an unnecessary spectrum that is not in fact used. The out-of-band unnecessary spectrum interferes with adjacent channels, and this spectrum is called a spurious emission.

Figure 2



Phase Shift Keying Modulation

Phase-shift keying (PSK) is a method of digital communication in which the phase of a transmitted signal is varied to convey information. There are several methods that can be used to accomplish PSK. The simplest PSK technique is called binary phase-shift keying (BPSK). It uses two opposite signal phases (0 and 180 degrees). The digital signal is broken up time wise into individual bits (binary digits). The state of each bit is determined according to the state of the preceding bit. If the phase of the wave does not change, then the signal state stays the same (0 or 1). If the phase of the wave changes by 180 degrees -- that is, if the phase reverses -- then the signal state changes (from 0 to 1, or from 1 to 0). Because there are two possible wave phases, BPSK is sometimes called biphase modulation. The time domain trace of an PSK modulated carrier is illustrated in the figure 1.1 below.

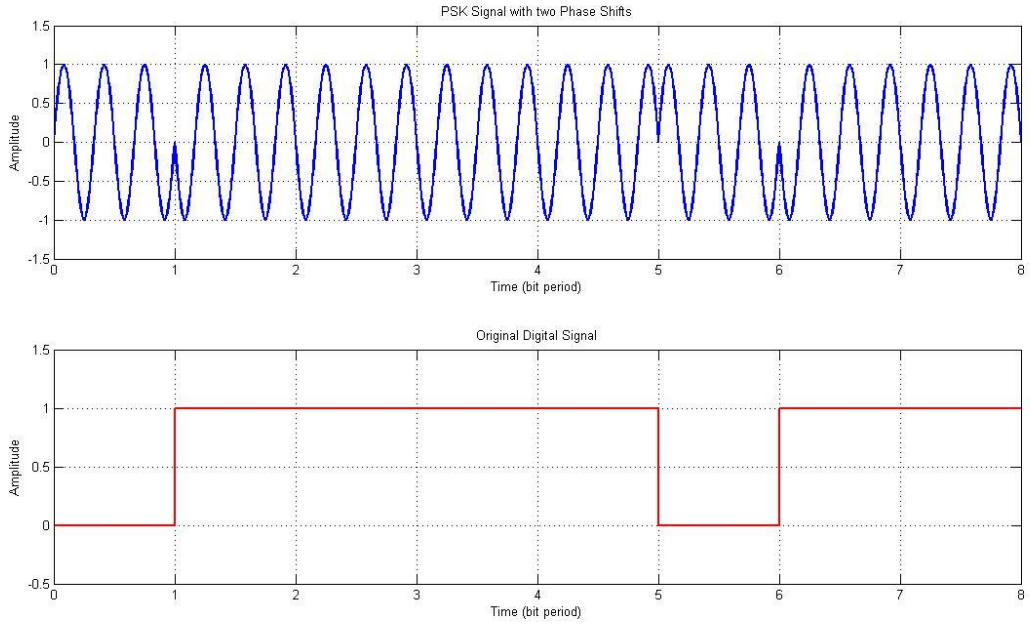


Figure 1.1

DE2-115 Board

The Cyclone EP4CE115 device equipped on the DE2-115 features 114,480 logic elements (LEs), the largest offered in the Cyclone IV E series, up to 3.9-Mbits of RAM, and 266 multipliers. In addition, it delivers an unprecedented combination of low cost and functionality, and lower power compared to previous generation Cyclone devices. The DE2-115 adopts similar features from the earlier DE2 series primarily the DE2-70, as well as additional interfaces to support mainstream protocols including Gigabit Ethernet (GbE). A High-Speed Mezzanine Card (HSMC) connector is provided to support additional functionality and connectivity via HSMC daughter cards and cables. For large-scale ASIC prototype development, a connection can be made with two or more FPGA-based boards by means of a HSMC cable through the HSMC connector. Figure 4 shows block diagram of DE2-115 board .

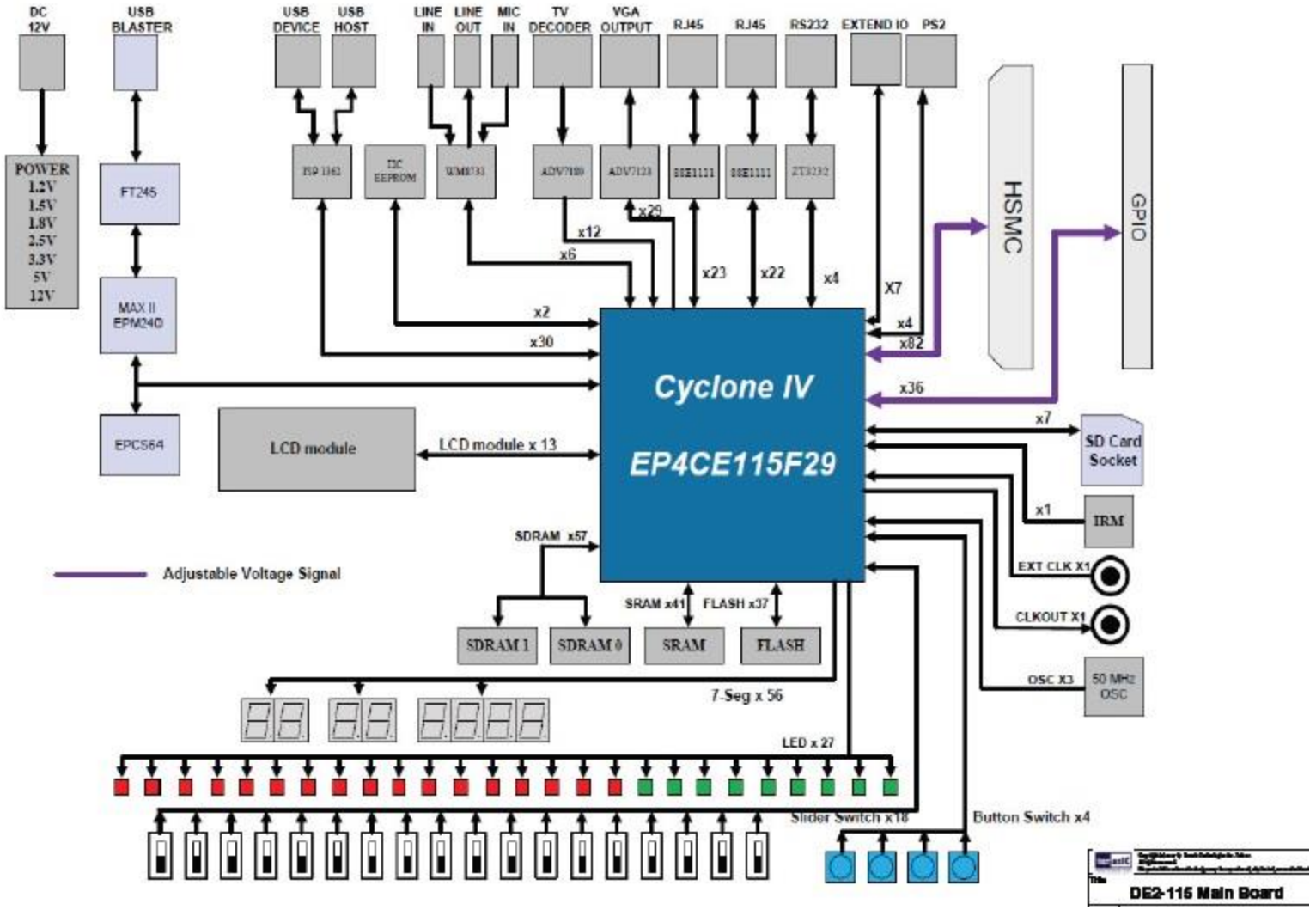

DC 12V
USB BLASTER
USB DEVICE
USB HOST
LINE IN
LINE OUT
MIC IN
TV DECODER
VGA OUTPUT
RJ45
RJ45
RS232
EXTEND IO
PS2
POWER
1.2V
1.5V
1.8V
2.5V
3.3V
5V
12V
FT245
MAX II EPM240
EPCS64
I2C EEPROM
ADV7123
88E1111
88E1111
ZT3232
HSMC
GPIO
x29
x12
x6
x23
x22
x4
X7
x4
x82
x2
x30
x36
Cyclone IV
EP4CE115F29
LCD module
LCD module x 13
x7
SD Card Socket
x1
IRM
EXT CLK X1
CLKOUT X1
OSC X3
50 MHz OSC
SDRAM x57
SRAM x41
FLASH x37
SDRAM 1
SDRAM 0
SRAM
FLASH
Adjustable Voltage Signal
7-Seg x 56
LED x 27
Slider Switch x18
Button Switch x4
DE2-115 Main Board

Figure 4

High Speed Mezzanine Card (HSMC)

The Data Conversion HSMC was created to provide a set of Analog to Digital and Digital to Analog interfaces including an Audio Codec interface.

The Data Conversion HSMC contains the following components.
Interfaces

1. HSMC Interface
2. Audio CODEC Interface
3. External Clock In Interface
4. External Clock Out Interface
5. ADC Channel A and B Input Interface
6. DAC Channel A and B Output Interface

Power supply
I2C Serial EEPROM

Block diagram shown in figure 5 shows the components on HSMC card and their interfaces.

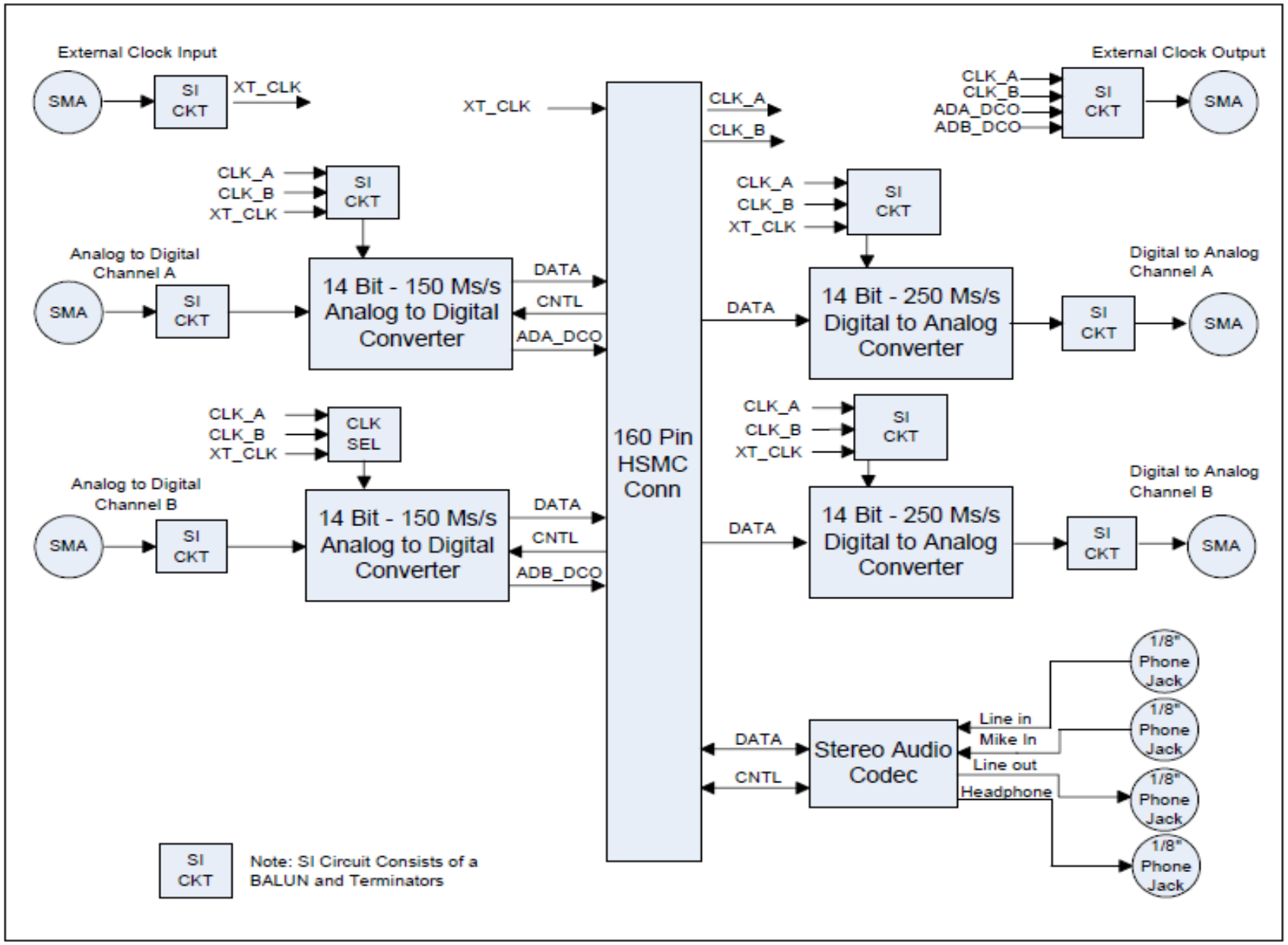

External Clock Input
SMA
SI CKT
XT_CLK
XT_CLK
CLK_A
CLK_B
External Clock Output
CLK_A
CLK_B
ADA_DCO
ADB_DCO
SI CKT
SMA
CLK_A
CLK_B
XT_CLK
SI CKT
Analog to Digital Channel A
SMA
SI CKT
14 Bit - 150 Ms/s Analog to Digital Converter
DATA
CNTL
ADA_DCO
CLK_A
CLK_B
XT_CLK
SI CKT
DATA
14 Bit - 250 Ms/s Digital to Analog Converter
SI CKT
SMA
Digital to Analog Channel A
CLK_A
CLK_B
XT_CLK
CLK SEL
Analog to Digital Channel B
SMA
SI CKT
14 Bit - 150 Ms/s Analog to Digital Converter
DATA
CNTL
ADB_DCO
160 Pin HSMC Conn
CLK_A
CLK_B
XT_CLK
SI CKT
DATA
14 Bit - 250 Ms/s Digital to Analog Converter
SI CKT
SMA
Digital to Analog Channel B
DATA
CNTL
Stereo Audio Codec
Line in
Mike In
Line out
Headphone
1/8" Phone Jack
1/8" Phone Jack
1/8" Phone Jack
1/8" Phone Jack
SI CKT
Note: SI Circuit Consists of a BALUN and Terminators

Figure 5

FPGA Implementation of FSK Modulator

Here we will be targeting a Cyclone-IV FPGA on development board DE2-115 for the implementation of FSK Modulator. The basic modules required for this implementation are :

1. UART (Universal Asynchronous Receiver and Transmitter)
2. Numerically Controller Oscillator (NCO)
3. Phase Locked Loop (PLL)
4. UP-Counter
5. High Speed Mezzanine Card (HSMC)

The figure 3 below shows how these modules are connected.

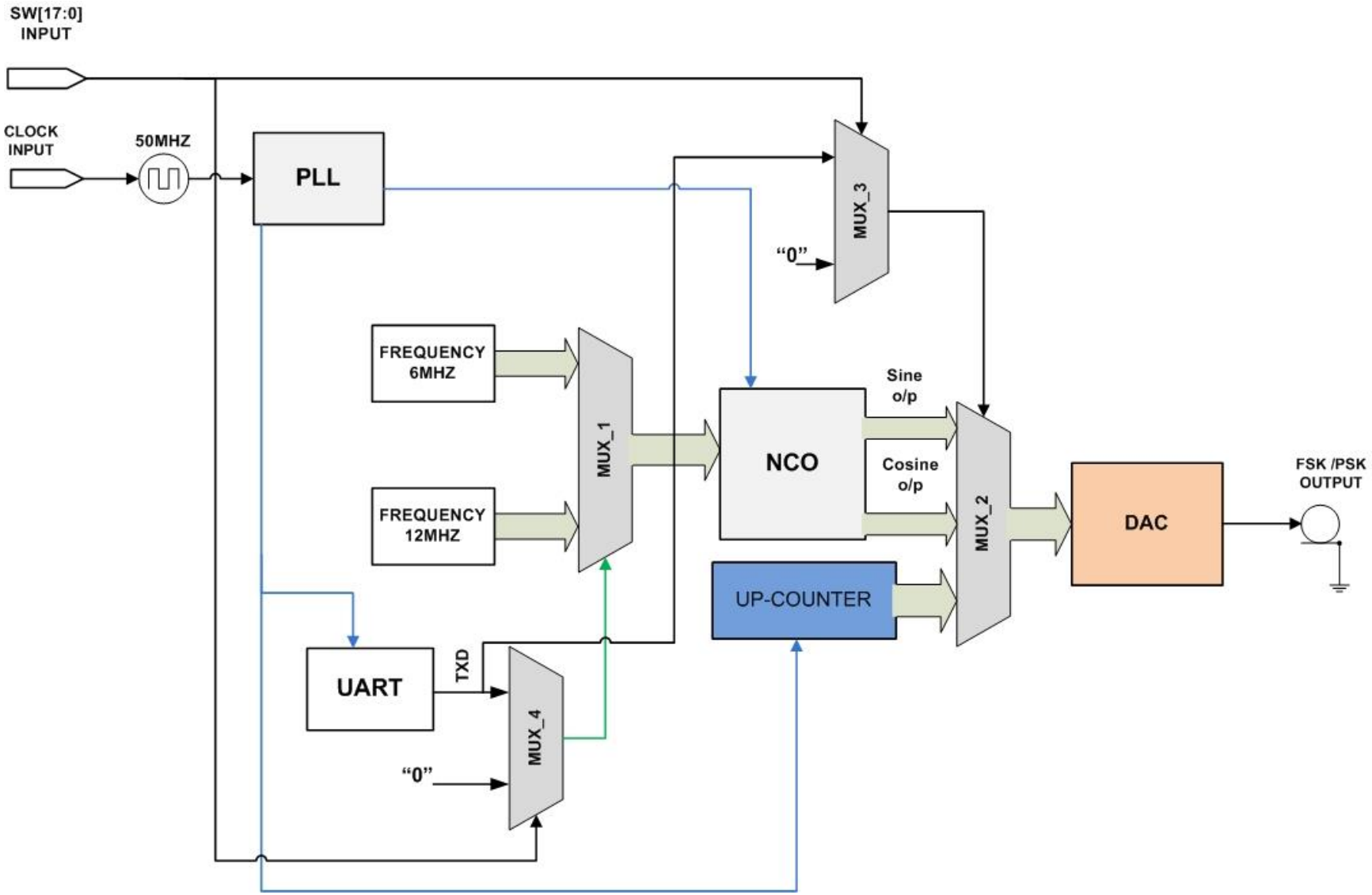

SW[17:0]
INPUT
CLOCK
INPUT
50MHZ
PLL
MUX_3
"0"
FREQUENCY
6MHZ
FREQUENCY
12MHZ
MUX_1
NCO
Sine
o/p
Cosine
o/p
MUX_2
DAC
FSK /PSK
OUTPUT
UP-COUNTER
UART
TXD
MUX_4
"0"

Figure 3

Binary FSK Modulation VHDL Description

As shown in figure 3 within this FPGA implementation PLL is being used to supply clean synchronized clock to UART, NCO and UP-Counter, whereas MUX_1 selects which frequency is set for the LUT-based NCO , the select signal is toggled by a MUX_4 output . User SW[17:0] selects whether to enable FSK Modulation by setting select lines to logic “0” of MUX_4 or enable Binary PSK Modulation by setting select lines to logic “1”.Binary Data is transmitted by a UART which is transmitting a pre-defined binary value in continuous loop at the rate of 9600 baud rate .When TX output drops to “0” NCO is set to generate 6Mhz sinewave and when it pulls up to logic “1” NCO is generating 12Mhz sinewave. The output of the NCO and the UP-Counter are selected by user via switches SW[17:0] using MUX_2 , here UP-Counter is being used for testing purpose only. When a UP-Counter is selected the output of DAC will be a saw-tooth wave , whereas NCO’s output will result in a sine wave. The output of MUX_2 is fed to a high-speed DAC populated on a HSMC card. There are two 14-bit DAC’s available on HSMC card one on channel A and the other on channel B , each has the conversion rate of 250Ms/ps. The analogue output of both DAC’s terminates on a surface mount SMA connector.

Binary PSK Modulation VHDL Description

Refer to figure 3 When user sets SW[17:0] to enable PSK Modulation MUX_4 output is logic “0” which sets NCO to generate 6Mhz of Sine and Cosine wave . UART serial output TX is routed via MUX_3 to MUX_2. The DAC “A” analogue output shifts its frequency synchronously with the bit toggling from the output of MUX_3 , which in turn change the channel selection of MUX_2.

VHDL Description

The implementation of blocks shown in figure 3 are described in VHDL in the following list of files :

1. Hsmc_card_interface_Top.vhd
2. UART_TX_CTRL.vhd
3. GPIO_demo.vhd
4. gh_nsincos_rom_14_4.vhd
5. gh_nco_lut_14p.vhd

Top VHDL File

The top vhdl file combines all components and contains interface description and signals for the DE2-115 board to HSMC card .

Below is the entity declaration of “Hsmc_card_interface_Top.vhd” file.

```
-- Top level module
ENTITY Hsmc_card_interface_Top  IS

        PORT (
            OSC_50  : IN STD_LOGIC_VECTOR(2 DOWNTO 0);
            -- HSMC Connector Signals
            -- ADC Signals
            AD_SCLK : INOUT STD_LOGIC;
            AD_SDIO : INOUT STD_LOGIC;
           -- U1 ADC Chip Channel A
            ADA_D   : IN STD_LOGIC_VECTOR(13 DOWNTO 0);
            ADA_DCO : IN STD_LOGIC;
            ADA_OE  : OUT STD_LOGIC;
            ADA_OR  : IN STD_LOGIC;
            ADA_SPI_CS : OUT STD_LOGIC;
            -- U2 ADC Chip Channel B
            ADB_D   : IN STD_LOGIC_VECTOR(13 DOWNTO 0);
            ADB_DCO : IN STD_LOGIC;
            ADB_OE  : OUT STD_LOGIC;
```

```
 ADB_OR  : IN STD_LOGIC;
 ADB_SPI_CS :  OUT STD_LOGIC;
--------------------
-- AIC23 AUDIO CODEC
                AIC_BCLK   : INOUT STD_LOGIC;
                AIC_DIN    : OUT STD_LOGIC;
                  AIC_DOUT   : INOUT STD_LOGIC;
                  AIC_LRCIN  : INOUT STD_LOGIC;
                  AIC_LRCOUT : INOUT STD_LOGIC;
                  AIC_SPI_CS : OUT STD_LOGIC;
                  AIC_XCLK   : OUT STD_LOGIC;
                  --
                  CLKIN1  : IN STD_LOGIC; -- Going to TP1
                  CLKOUT0 : OUT STD_LOGIC; -- Going to TP2
                  -- DAC Channel A
                  DA : OUT STD_LOGIC_VECTOR(13 DOWNTO 0);
                  -- DAC Channel B
                  DB : OUT STD_LOGIC_VECTOR(13 DOWNTO 0);
                  -- Both ADC and DAC Clocks are slected by Jumper settings
                  -- i.e J15, J17, J3 and J7
                  -- Differental Clock Output
                  FPGA_CLK_A_N : INOUT STD_LOGIC;
                  FPGA_CLK_A_P : INOUT STD_LOGIC;
                  --
```

```
                        FPGA_CLK_B_N : INOUT STD_LOGIC;
                        FPGA_CLK_B_P : INOUT STD_LOGIC;
                        --
                        J1_152 : INOUT STD_LOGIC; -- Going to TP5
                        -- External Clock Differental input
                        XT_IN_N : IN STD_LOGIC;
                        XT_IN_P : IN STD_LOGIC;
                        -- Switches
                        SW      :  IN STD_LOGIC_VECTOR(17 DOWNTO 0);
                        -- Push Buttons
                        KEY     :  IN STD_LOGIC_VECTOR(3 DOWNTO 0);
                        -- UART Signals
                        UART_CTS: OUT STD_LOGIC; -- Going to PIN_G14
                        UART_TXD: OUT STD_LOGIC; -- Going to PIN_G9
                        UART_RTS:  IN STD_LOGIC; -- Going to PIN_J13
                        UART_RXD:  IN STD_LOGIC; -- Going to PIN_G12
                        -- LEDs
                        LEDG    :  OUT STD_LOGIC_VECTOR(8 DOWNTO 0);
                        LEDR    :  OUT STD_LOGIC_VECTOR(17 DOWNTO 0)
                            ) ;
END Hsmc_card_interface_Top ;
```

Architecture description

```
ARCHITECTURE Data_acquisition OF Hsmc_card_interface_Top IS
------------------------------------------------------------
--Convert integer to unsigned std_logic vector function
function int2ustd(value : integer; width : integer) return std_logic_vector is
-- convert integer to unsigned std_logicvector
variable temp :   std_logic_vector(width-1 downto 0);
begin
        if (width>0) then
                temp:=conv_std_logic_vector(conv_unsigned(value, width ), width);
        end if ;
        return temp;
end int2ustd;
-------------
-- PLL_locked
Component PLL IS
        PORT
        (
                areset   : IN STD_LOGIC  := '0';
                inclk0   : IN STD_LOGIC  := '0';
                c0       : OUT STD_LOGIC ;
                c1       : OUT STD_LOGIC ;
                c2       : OUT STD_LOGIC ;
```

```
                c3          : OUT STD_LOGIC ;
                locked      : OUT STD_LOGIC
        );
End Component;
---------------------------------------
-- gh NCO
Component gh_nco_lut_14p is
        GENERIC (freq_word_size: INTEGER := 32);
        port(
                clk    : in STD_LOGIC;
                rst    : in STD_LOGIC;
                FREQ   : in STD_LOGIC_VECTOR(freq_word_size-1 downto 0);
                PHASE : in STD_LOGIC_VECTOR(13 downto 0):=(others => '0');
                nsin   : out STD_LOGIC_VECTOR(13 downto 0);
                cos    : out STD_LOGIC_VECTOR(13 downto 0)
                );
End Component;
-----------------------------------------------
Component GPIO_demo is
    Port ( SW       : in  STD_LOGIC_VECTOR (17 downto 0);
        CLOCK_50 : in  STD_LOGIC;
        LEDR0    : out  STD_LOGIC;
        UART_TXD : out  STD_LOGIC
                        );
```

```
End Component;
---------------
-- ADC data registers
SIGNAL ADC_data_A : STD_LOGIC_VECTOR(13 DOWNTO 0);
SIGNAL ADC_data_B : STD_LOGIC_VECTOR(13 DOWNTO 0);
--
-- DAC Data registers
SIGNAL DAC_A : STD_LOGIC_VECTOR(13 DOWNTO 0);
SIGNAL DAC_B : STD_LOGIC_VECTOR(13 DOWNTO 0);
--
SIGNAL freq_out     : STD_LOGIC_VECTOR(13 DOWNTO 0);
SIGNAL sin_out_reg  : STD_LOGIC_VECTOR(13 DOWNTO 0);
SIGNAL sin_out_adj  : STD_LOGIC_VECTOR(13 DOWNTO 0);
SIGNAL nco_sin_out  : STD_LOGIC_VECTOR(13 DOWNTO 0);
SIGNAL nco_cos_out  : STD_LOGIC_VECTOR(13 DOWNTO 0);
SIGNAL nco_squ_out  : STD_LOGIC_VECTOR(13 DOWNTO 0);
SIGNAL nco_saw_out  : STD_LOGIC_VECTOR(13 DOWNTO 0);
Constant OFF_set:STD_LOGIC_VECTOR(13 DOWNTO 0):="01111111111111";
Constant Phase_input : INTEGER := 180;
Constant Phase_mode  : INTEGER := 80;
Constant Freq_mode   : INTEGER := 20;
Constant Freqy_set   : INTEGER := 12;
-----------------------------------------------------------------------
Constant Freq_12Mhz  : INTEGER := 515396075; --<< to generate 12Mhz
```

```
Constant Freq_6Mhz   : INTEGER := 257698037; --<< to generate 6Mhz
SIGNAL   Freq_set    : STD_LOGIC_VECTOR(31 DOWNTO 0);
SIGNAL   Freq_set_12Mhz    : STD_LOGIC_VECTOR(31 DOWNTO 0);
SIGNAL   Freq_set_6Mhz     : STD_LOGIC_VECTOR(31 DOWNTO 0);
--
SIGNAL Count : STD_LOGIC_VECTOR(13 DOWNTO 0);
SIGNAL reset_n, reset_pll, NCO_valid, DAC_rdy, UART_Txout, UART_pll : STD_LOGIC ;
SIGNAL sys_clk, sys_clk_90deg, sys_clk_180deg, sys_clk_270deg, pll_locked : STD_LOGIC ;

Begin
----------------------
reset_n   <= KEY(3);
reset_pll  <= not(reset_n);
AD_SCLK  <= SW(0);                    -- (DFS)Data Format Select
AD_SDIO  <= SW(1);                    -- (DCS)Duty Cycle Stabilizer Select
ADA_OE  <= '0';                       -- enable ADA output
ADA_SPI_CS <= '1';                    -- disable ADA_SPI_CS (CSB)
ADB_OE  <= '0';                       -- enable ADB output
ADB_SPI_CS <= '1';                    -- disable ADB_SPI_CS (CSB)
-- Clock Sources
FPGA_CLK_A_P <=  sys_clk_180deg;
FPGA_CLK_A_N <= not(sys_clk_180deg);
FPGA_CLK_B_P <=  sys_clk_270deg;
FPGA_CLK_B_N <= not(sys_clk_270deg);
```

```
LEDG(2)  <= pll_locked;
Freq_set_12Mhz <= int2ustd(Freq_12Mhz,32);
Freq_set_6Mhz  <= int2ustd(Freq_6Mhz,32);
-----------------------------------------
-- Freq_set <= Freq_set_12Mhz when SW(4 downto 2) = "011" else Freq_set_6Mhz;
-------------------------------------------------------------------------------
-- PLL Port Mapping
Clock : PLL  port map (
                    reset_pll,
                    OSC_50(0),
                     sys_clk,
                     sys_clk_90deg,
                     sys_clk_180deg,
                     sys_clk_270deg,
                    pll_locked
                        );
----------------------
--gh NCO Port Mapping
Sine_Wave_Generator : gh_nco_lut_14p  port map (
                                                                  sys_clk,
                                                                reset_pll,
                                                                Freq_set,
                                                                int2ustd(Phase_input,14),
                                                                nco_sin_out,
```

```
                                                    nco_cos_out
                                                        );
-------------------------------------------------------
-- UART Port Mapping
UART_module : GPIO_demo port map (
                    SW,
                    OSC_50(0),
                     LEDR(0),
                     UART_Txout
                       );
---------------------------------
Generating_Ramp: PROCESS(sys_clk)
BEGIN
    IF reset_n = '0' THEN
        Count <= (others=>'0');
                    ELSE IF rising_edge(sys_clk) THEN
                      Count <= Count + "0000000000001" ;
        END IF;
     END IF;
END PROCESS;
-----------------------------------------------
Interfacing_ADC_DAC: PROCESS(sys_clk, reset_n)
BEGIN
```

```
IF reset_n = '0' THEN
                    ADC_data_A <= (others=>'0');
                     ADC_data_B <= (others=>'0');
                     --
                     DA <= (others=>'0');
                     DB <= (others=>'0');
                     --
                     LEDR(17 downto 15) <= (others=>'0');
                     DAC_rdy <= '0';
                  -- Defualt Frquency settings
                    Freq_set <= Freq_set_12Mhz;
                    -- 1st Condition --------------------
                      ELSIF SW(17 downto 15) = "000" THEN
                     LEDR(17 downto 15) <= "000";
                     ADC_data_A <= ADA_D;
                    ADC_data_B <= ADB_D;
                    ----------------
                    -- DAC's A & B are connected to an Up-Counter
                  DA <= Count;
                  DB <= not(Count);

          -- 2nd Condition --------------------
                      ELSIF SW(17 downto 15)  = "001" THEN
                                LEDR(17 downto 15) <= "001";
```

```
                    -- ADC output is connected directly to DAC
                   DA <= ADA_D;
                   DB <= ADB_D;

 -- 3rd Condition --------------------
    ELSIF SW(17 downto 15)  = "010" THEN
                  LEDR(17 downto 15) <= "010";
                       -- DAC is connected to a NOC output
                    DA <= nco_sin_out + OFF_set;
                   DB <= nco_sin_out + OFF_set;

 -- 4th Condition --------------------
 -- Connect loop back cable i.e
 -- DAC_A o/p -> ADC_A i/p
-- ADC_A digital o/p --> DAC_B
    ELSIF SW(17 downto 15)  = "100" THEN
                   LEDR(17 downto 15) <= "100";
                  -- DAC_A is connected to a NOC output
                  DA <= nco_sin_out + OFF_set;
                DB <= ADA_D;
-------------------------------------
 -- 5th Condition --------------------
    ELSIF SW(17 downto 15)  = "101" THEN
                   LEDR(17 downto 15) <= "101";
```

```
                -- FSK Modulation of UART output
                        if UART_Txout = '1' then
                           Freq_set <= Freq_set_12Mhz;
                            DA <= nco_sin_out + OFF_set;
                        else
                           Freq_set <= Freq_set_6Mhz;
                           DA <= nco_sin_out + OFF_set;
                        end
--------------------------------
  -- 6th Condition --------------------
    ELSIF SW(17 downto 15)  = "110" THEN
                  LEDR(17 downto 15) <= "110";
                  -- PSK Modulation of UART output
                 if UART_Txout = '1' then
                  DA <= nco_sin_out + OFF_set;
                  else
                  DA <= nco_cos_out + OFF_set;
                  end if;
    -- 7th Condition --------------------
    ELSIF SW(17 downto 15)  = "111" THEN
                  LEDR(17 downto 15) <= "111";
                        -- FSK Modulation @6.1Khz Square wave
                       if Countrg(18) = '1' then
                                  Freq_set <= Freq_set_12Mhz;
```

```
                DA <= nco_sin_out + OFF_set;
                DB <= nco_cos_out + OFF_set;
            else
                Freq_set <= Freq_set_6Mhz;
               DA <= nco_sin_out + OFF_set;
                DB <= nco_cos_out + OFF_set;
           end if;


        END IF;
END PROCESS;
END Data_acquisition;
```

Main Processing

The process labeled “Interfacing_ADC_DAC” in the top entity as shown above has been divided into 6 conditional functions . It starts with resetting output ports DA and DB that are connected to DAC_A and DAC_B , similarly ADC registers ADC_data_A and ADC_data_B .

1st Condition

First condition detects user input from switches SW[17:15] and connects ADC ‘A’ and ‘B’ 14-bit inputs to internal 14-bit registers ADC_data_A and ADC_data_B and DAC ‘A’ 14-bit output to UP-Counter whereas DAC ‘B’ output to inverted bits of UP-Counter if SW[17:15] = “000” .

2nd Condition

Second condition detects user input from switches SW[17:15] and connects ADC ‘A’ 14-bit input port to DAC ‘A’ output port and whereas DAC ‘B’ output port to ADC ‘B’ input port if SW[17:15] position is set to “001” .

3rd Condition

Third condition detects user input from switches SW[17:15] if these are set to "010" it connects 14-bit NCO’s sine wave output added with a offset value to out port of DAC ‘A’ and DAC ‘B’ . NCO is generating sine wave set by default value i.e 12Mhz initially set during reset when detecting reset_n = '0' .

4th Condition

Fourth condition detects user input from switches SW[17:15] if these are set to "100" it connects 14-bit NCO’s sine wave output added with a offset value to out port of DAC ‘A’ . Externally DAC ‘A’ analogue output from SMA connector is connected by a RF cable to analogue input of ADC ‘A’ , the converted 14-bit output of ADC ‘A’ is fed to DAC ‘B’. This condition can be used to evaluate the performance of DAC and ADC .

5th Condition

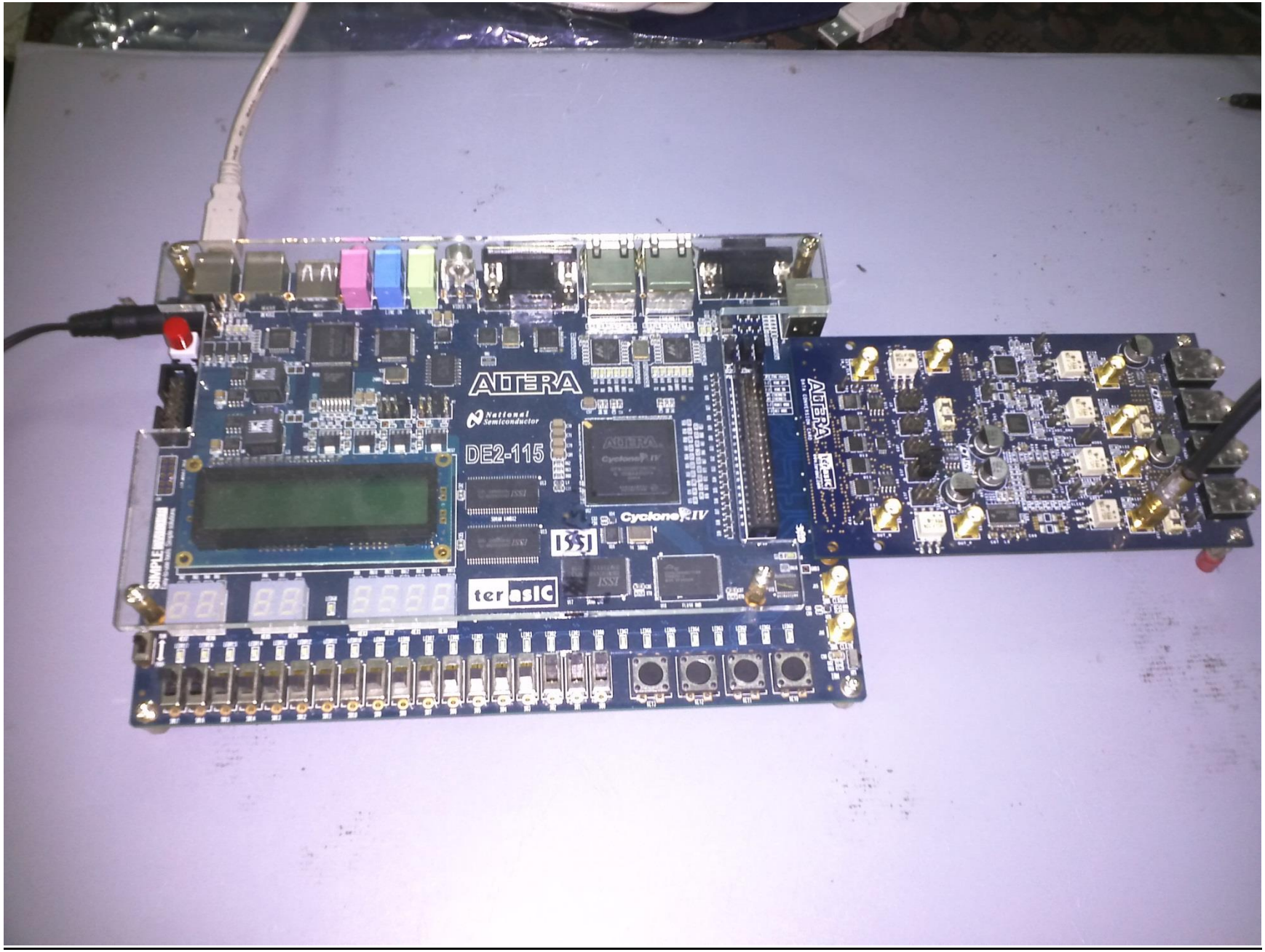
Fifth condition detects user input from switches SW[17:15] if these are set to "101" , it assigns internal Freq_set register to 12Mhz when UART TX output is at logic “1” and 6Mhz when TX output is at “0” . Simultaneously it connects NCO output to DAC ‘A’ , the analogue output of DAC ‘A’ which terminates at SMA connector can be observed on the scope to monitor FSK modulation .

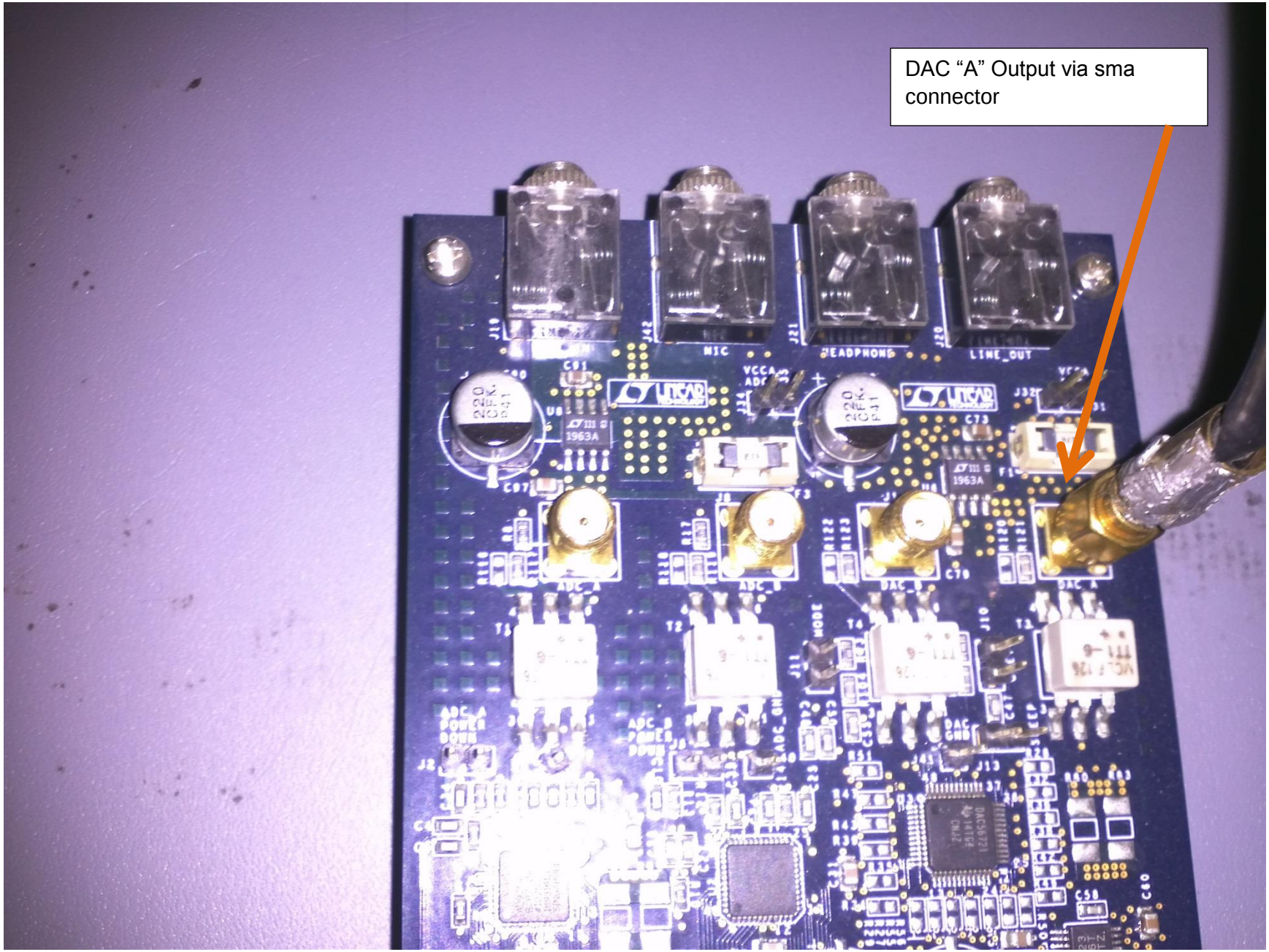
6th Condition

If switches SW[17:15] are set to "110" , DAC ‘A’ ’ output port is connected to Sine wave NCO output when UART_Txout is at logic “1” on UART_Txout toggling to logic “0” DAC ‘A’ ’ output port is connected to Cosine wave output of NCO , thus generating Phase Shift Keying modulated wave . UART TX output is also routed to UART_TXD pin of UART on DE2-115 for testing purposes

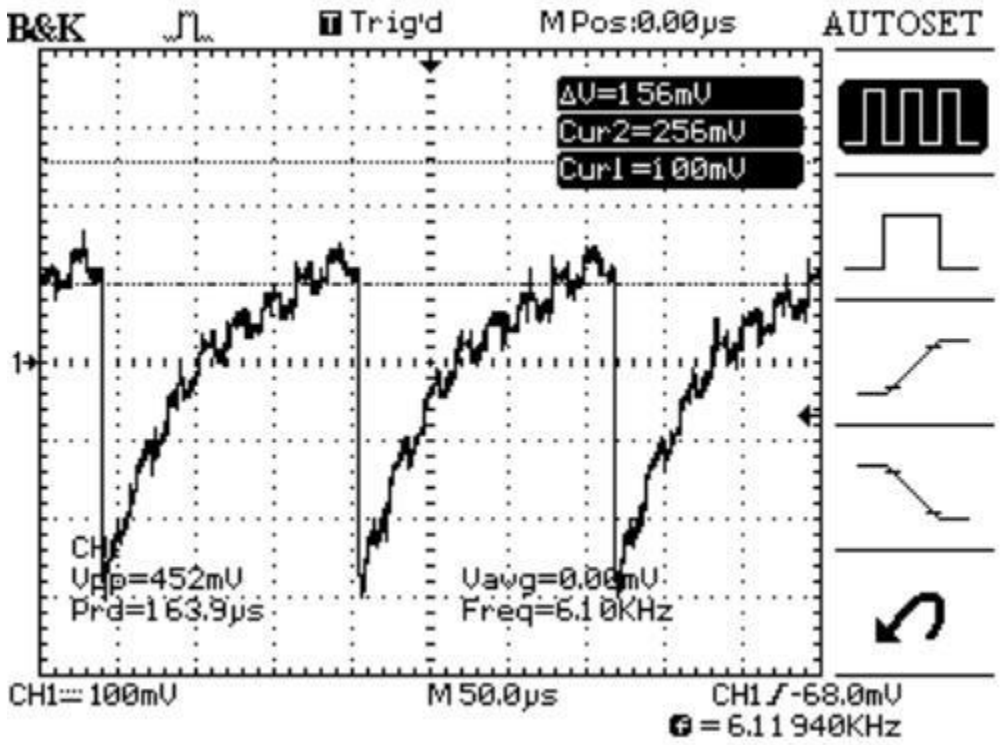
Test Set-up

Pictures below show DE2-115 based test setup for testing Binary FSK and PSK modulation testing described in the top entity “Hsmc_card_interface_Top”.Hsmc Card is connected to DE2-115 board via Hsmc port with 160 pins , refer to DE2-115 Schematic from Terasic for more details. Picture shows a RF cable connected to surface mount SMA connector of DAC “A” analogue output .

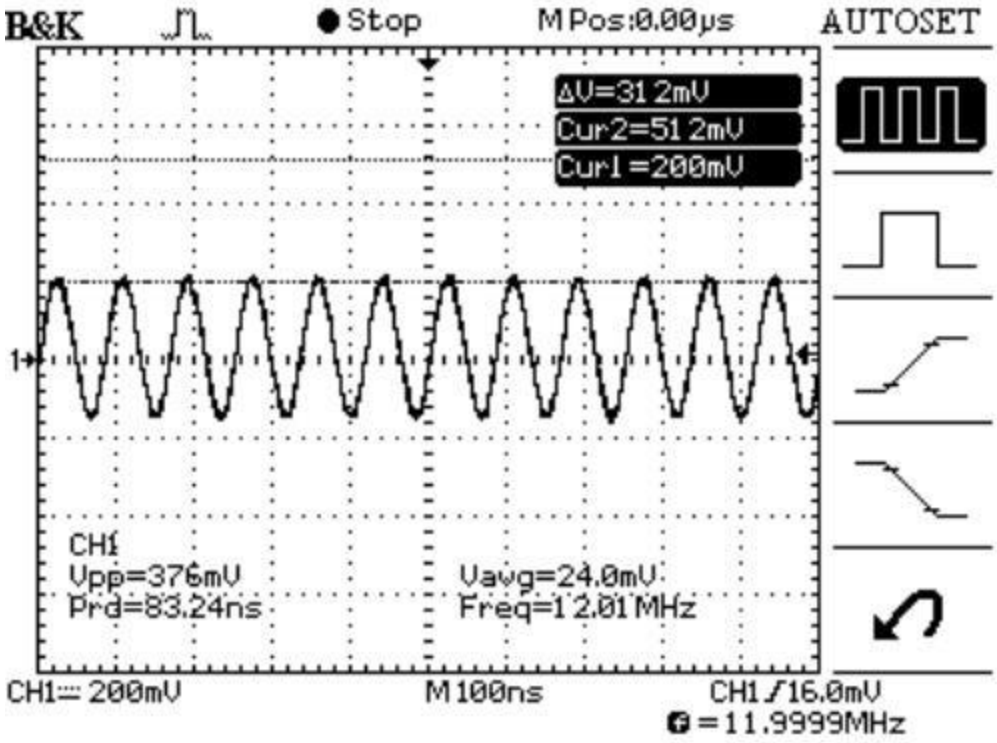

ALTERA
National Semiconductor
DE2-115
Cyclone IV
terasIC


DAC “A” Output via sma connector
MIC
HEADPHONE
LINE_OUT

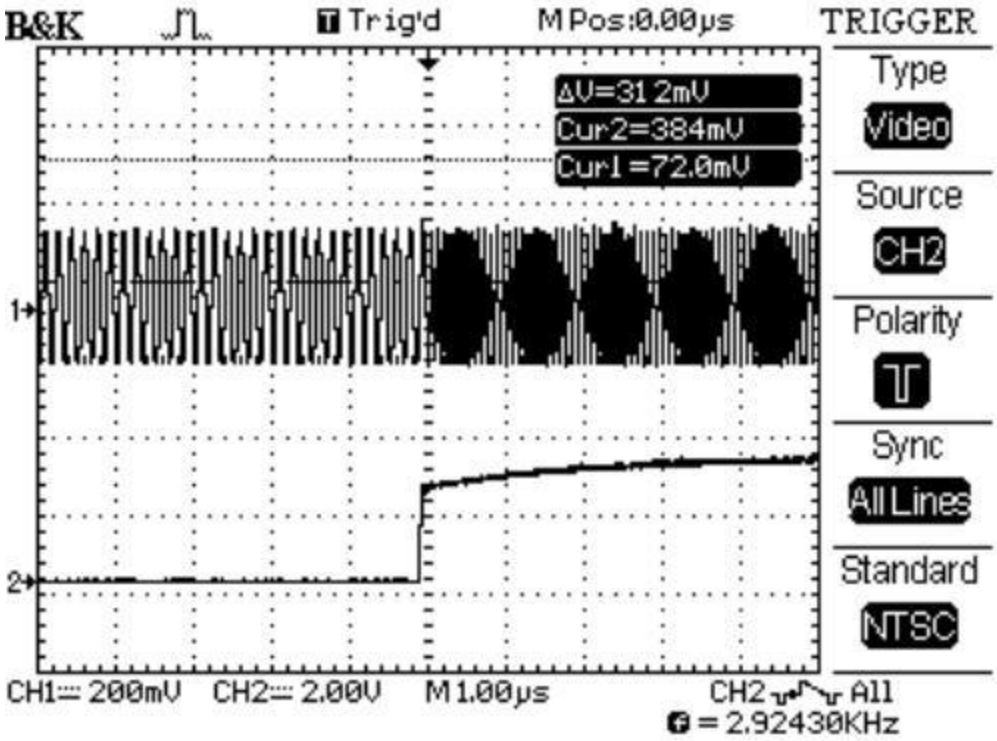
Testing 1st condition: The picture below shows oscilloscope screen capture of 1st condition , its shows output of DAC “A”. It shows analogue output of the DAC translating 14-bit Up-Counter values.



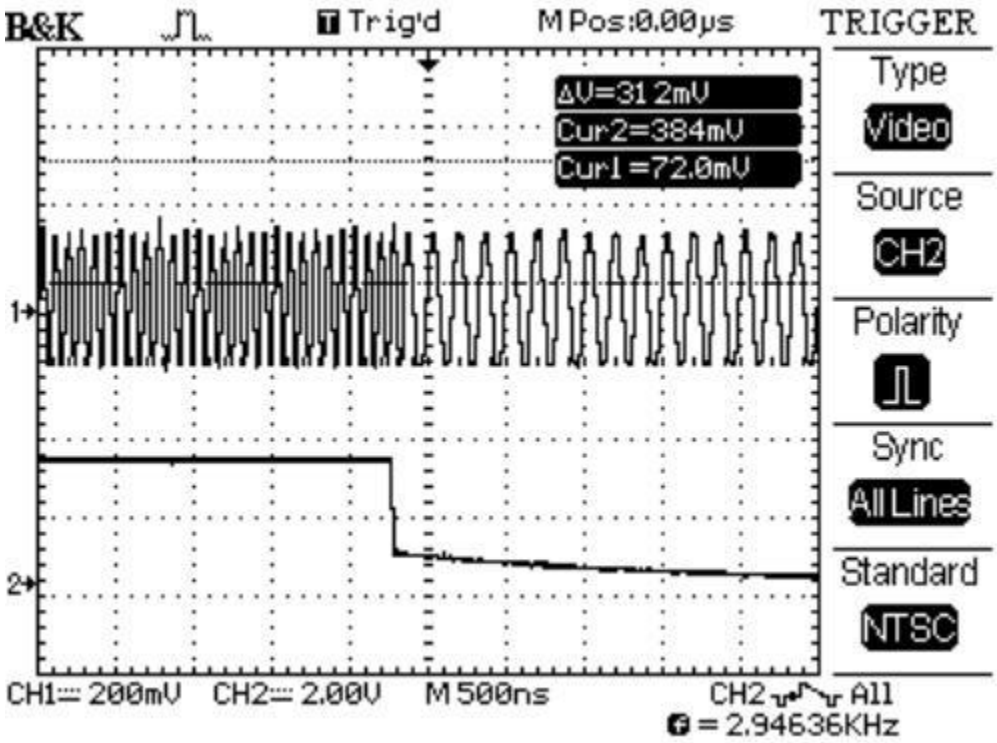
Testing 3rd condition: The picture below shows oscilloscope screen capture of 3rd condition , its shows output of DAC “A” which is a pure sine wave of 12Mhz.



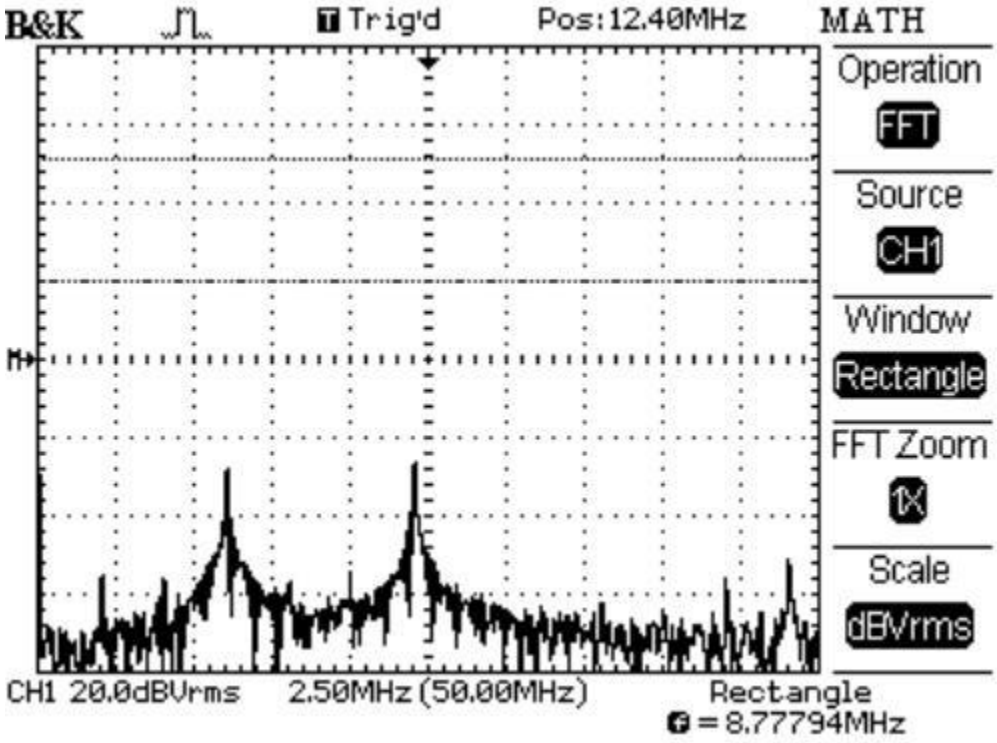
Testing 5th condition: The pictures below shows oscilloscope screen captures of 5th condition, its shows Binary FSK modulated output of DAC “A”. Channel 2 trace shows the rising edge of the UART TX output whereas Channel 1 trace shows change in frequency from 6Mhz to 12Mhz. Second picture shows shift in frequency from 12Mhz to 6Mhz in reference to the falling edge of the TX output. Third screen shot shows FFT trace of the oscilloscope, clearly shows peaks at 6Mhz and 12Mhz.



Binary FSK Modulation on rising edge of UART TX

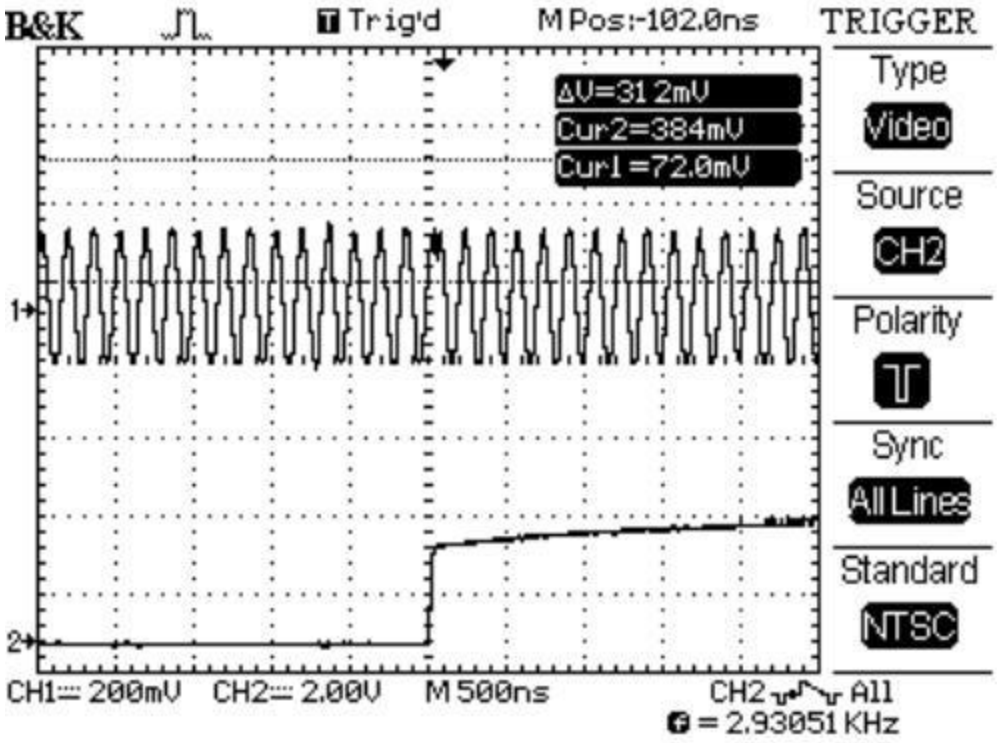


Binary FSK Modulation on falling edge of UART TX

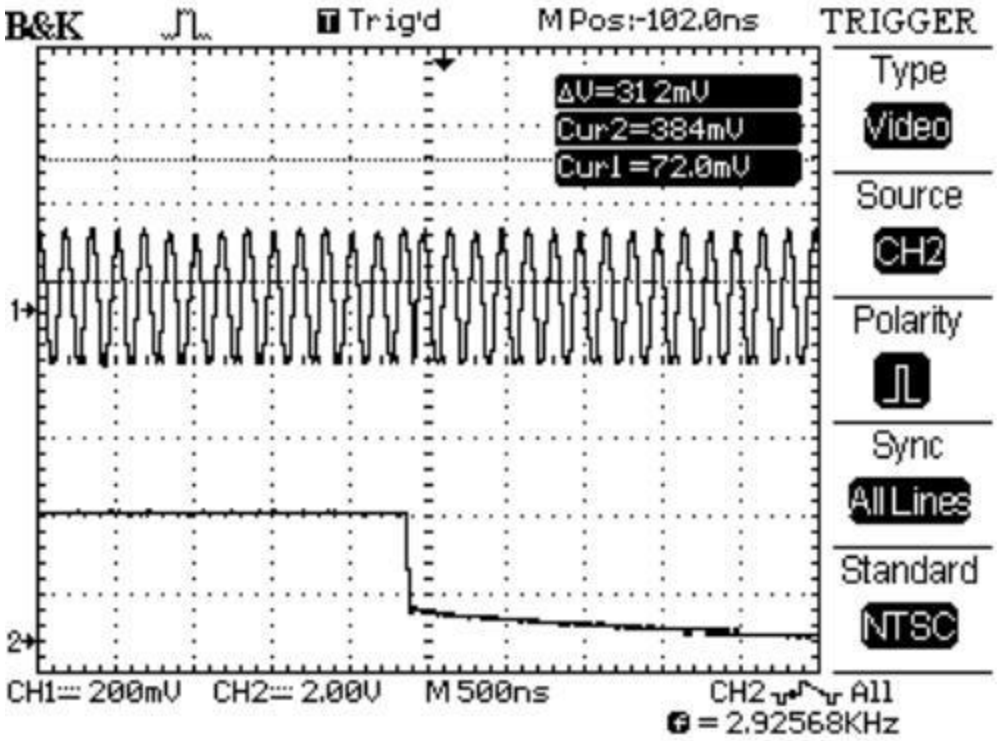


FFT of Binary FSK Modulation

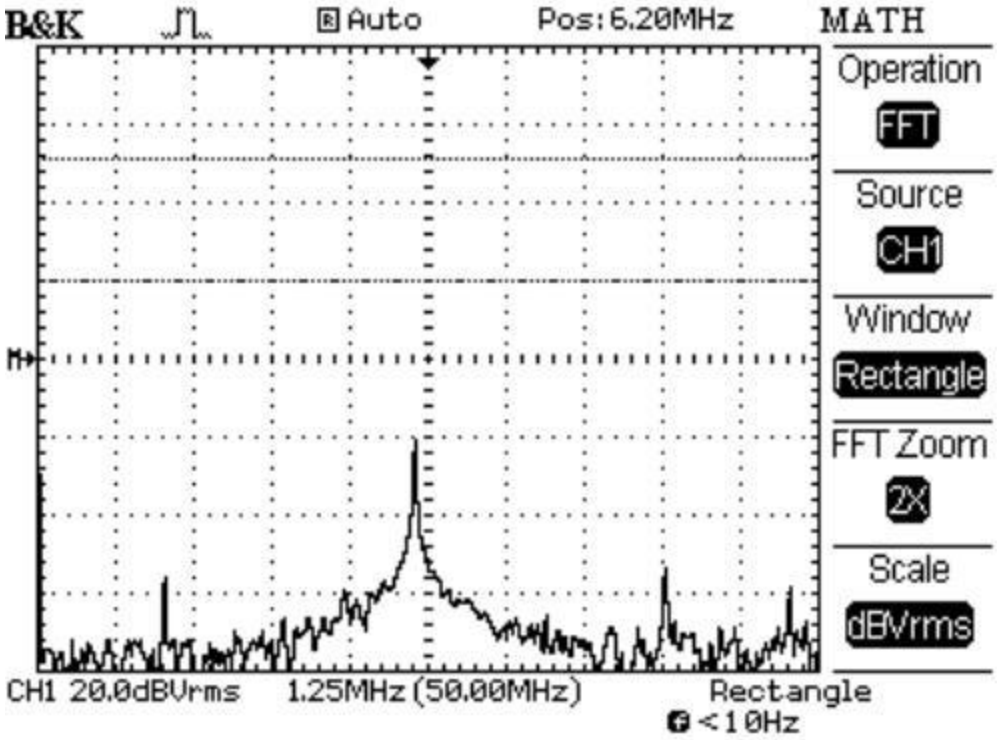
Testing 6th condition: The pictures below shows oscilloscope screen captures of 6th condition, its shows Binary PSK modulated output of DAC “A”. Channel 2 trace shows the rising edge of the UART TX output whereas Channel 1 trace shows change in Phase of 6Mhz sine wave. Second picture shows Phase reversal in reference to the falling edge of the TX output. Third screen shot shows FFT trace of the oscilloscope, clearly shows peak at 6Mhz only.



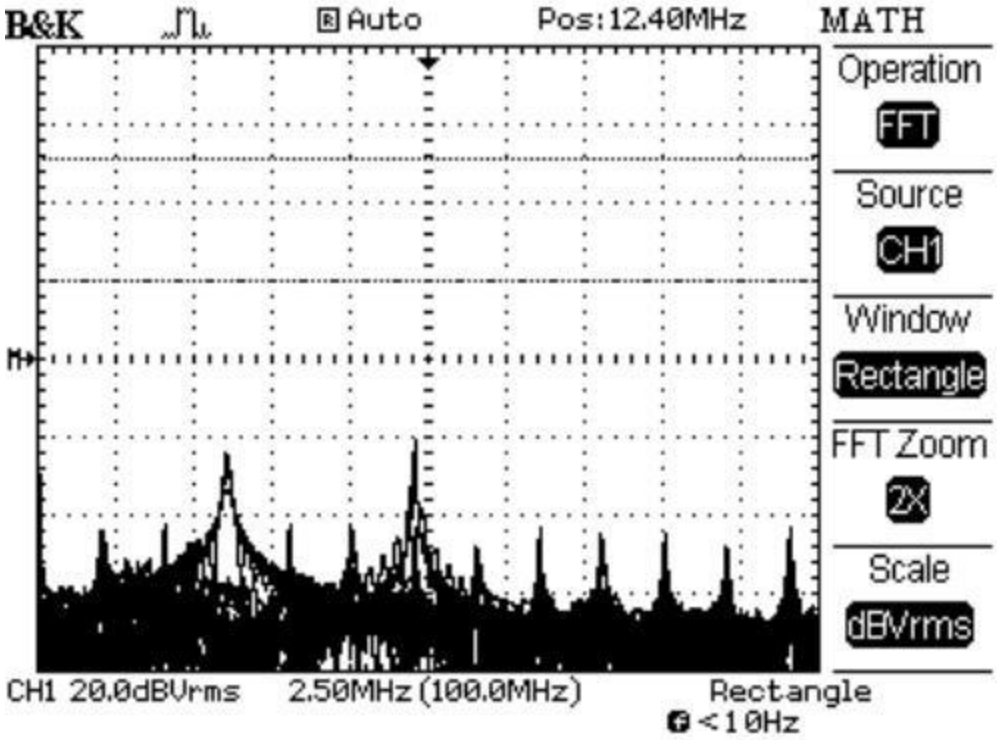
Binary PSK Modulation at Rising Edge of TX output



Binary PSK Modulation at Falling Edge of TX output



FFT trace of Binary PSK Modulation at 6Mhz



FFT Trace of FSK Modulation on 7th condition , modulating Countreg(18) i.e 6.1Khz Square Wave

Embedded Strings inc is a well-established company which specializes in the design, development, testing and manufacturing of complex embedded systems/Hardware. Our target markets are healthcare, telecommunication, aerospace and remote monitoring.

We specialize in Field Programmable Gate Array design , we have developed our own VHDL/Verilog ip cores that have been integrated in high performance embedded systems, Nios SOC , ARM SOC and Digital Signal Processing.

Please feel free to contact us in case you need to utilize any of the services that our company provides. We guarantee you complete security and confidentiality of information that you will share with us.

Author of this article can be reached at : asimghr@gmail.com

Download complete Quartus project from here : https://drive.google.com/file/d/0BwDURmHIJMZzUk9pSlZ2U2lkOXc/view?usp=sharing

Thank you ,

Ahmed Asim Ghouri
Embedded Strings inc
Website : www.emstrings.com
Email : support@emstrings.com