Politecnico di Torino

Academic Year 2014-2015

Specification and simulation of digital system:

FM radio transmitter

Group 10:

Andrea Ferri
Daniele Dallaglio
Luca Cucci
Marco Lucarella
Marina Zafiri
Matteo Bricchi

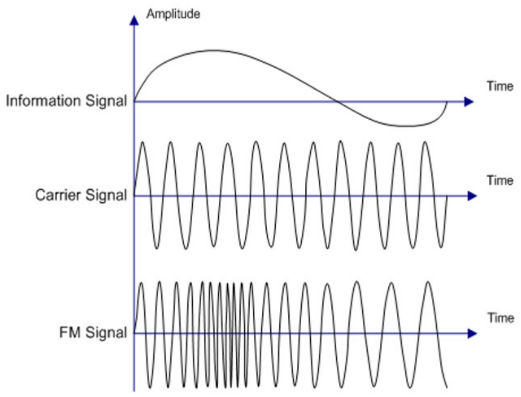
Introduction and objectives

The aim is develop with an evaluation board with a FPGA and VHDL a FM radio transmitter capable of sending at different radio frequencies some musical notes.

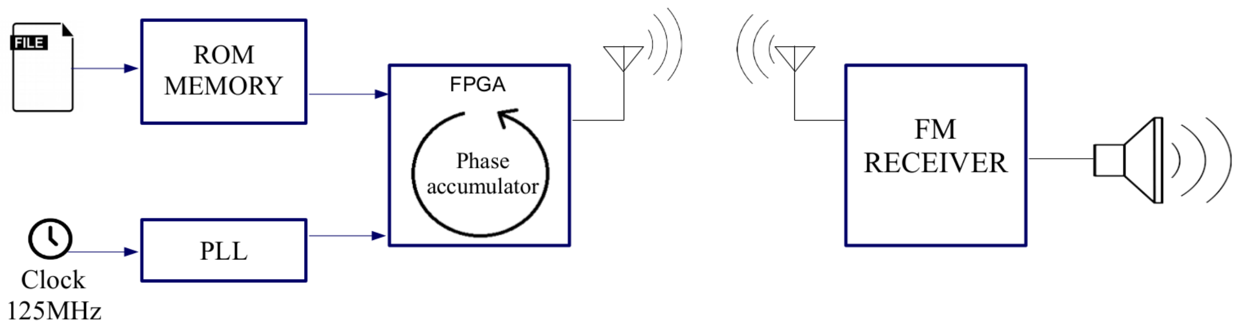
FM modulation

In telecommunications and signal processing, frequency modulation (FM) is the encoding of information in a carrier wave by varying the instantaneous frequency of the wave.

Digital data can be encoded and transmitted via a carrier wave by shifting the carrier's frequency among a predefined set of frequencies, a technique known as frequency shift keying (FSK). FSK is widely used in modems and fax modems, and can also be used to send Morse code.



Architecture



The architecture of the FM transmitter is composed by:

- **ROM Memory:** this component, using data from a text file, creates an image of memory that is loaded into the ROM memory;
- **Phase Locked Loop:** this component, created using a wizard tools of the suite, is used to multiply the clock signal of the board oscillator;
- **FPGA:** implements the VHDL code for the FM modulation;
- **FM Receiver:** standard FM receiver to test the transmission.

Implementation

The used **FPGA board** is the Digilent Zybo Zynq-7000 development board with an internal clock signal of 125MHz, several GPIO pins and LEDs, as an antenna to transmit has been used a simple wire and as a FM receiver a smartphone.

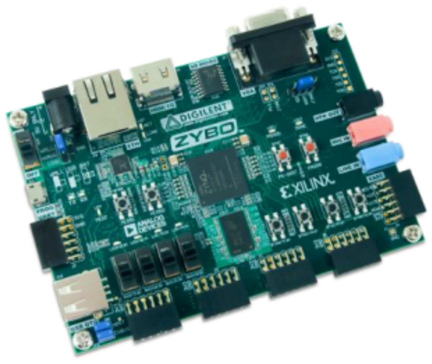


Figure 1. ZYBO Zynq-7000 development board.

The **ROM Memory** is a 256x8 bit with as input a 8bit std_logic signal to address the memory cells and the clock signal, as output a 8bit std_logic signal to read the data pointed by the address input.

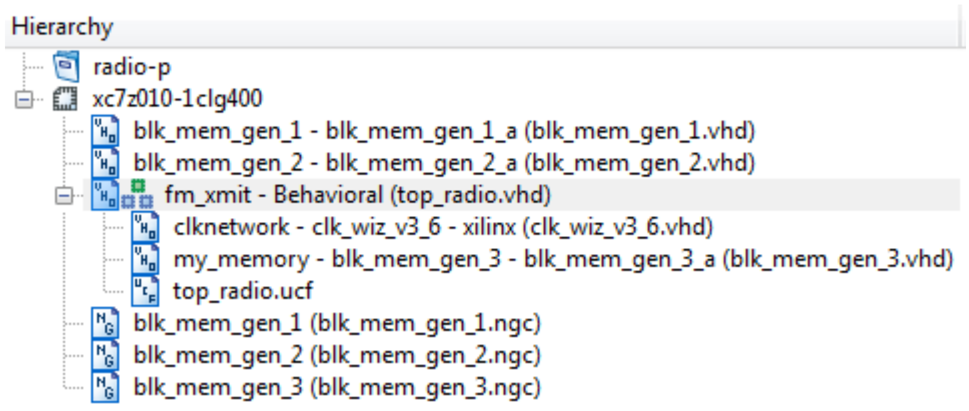
```
component  blk_mem_gen_3 IS
   PORT (
      clka  : IN STD_LOGIC;
      addra : IN STD_LOGIC_VECTOR(8 DOWNTO 0);
      douta : OUT STD_LOGIC_VECTOR(7 DOWNTO 0)
   );
END component;
```

This memory is initialized using an image memory created from a .coe file, where

- "memory_initialization_radix" field that is used to set the data format (hexadecimal);
- "memory_initialization_vector" contains the coded musical notes.

```
memory_initialization_radix=16;
memory_initialization_vector=
00,00,00,00,
07,07,07,07, 10,
07,07,07,07, 10,
08,08,08,08, 10,
09,09,09,09, 10,
09,09,09,09, 10,
08,08,08,08, 10,
07,07,07,07, 10,
06,06,06,06, 10,
05,05,05,05, 10,
05,05,05,05, 10,
06,06,06,06, 10,
07,07,07,07, 10,
07,07,07,07,07,07, 10,
06,06, 10,
```

In order to test and transmit different signals three different memories have been created with different contents. All the memories have been added to the project in order to easily switch from a memory to another.



```
Hierarchy
  radio-p
  xc7z010-1clg400
      blk_mem_gen_1 - blk_mem_gen_1_a (blk_mem_gen_1.vhd)
      blk_mem_gen_2 - blk_mem_gen_2_a (blk_mem_gen_2.vhd)
      fm_xmit - Behavioral (top_radio.vhd)
          clknetwork - clk_wiz_v3_6 - xilinx (clk_wiz_v3_6.vhd)
          my_memory - blk_mem_gen_3 - blk_mem_gen_3_a (blk_mem_gen_3.vhd)
          top_radio.ucf
      blk_mem_gen_1 (blk_mem_gen_1.ngc)
      blk_mem_gen_2 (blk_mem_gen_2.ngc)
      blk_mem_gen_3 (blk_mem_gen_3.ngc)
```

The **Phase Locked Loop** created by the wizard generates a 320MHz output clock using as input a 125MHz clock provided by the on board oscillator.

```
-- PLL component from 125MHz to 320MHz
component clk_wiz_v3_6 is
   port (
      CLK_IN1           : in     std_logic;  -- Clock in ports
      CLK_OUT1          : out    std_logic;  -- Clock out ports
      RESET             : in     std_logic;  -- Status and control signals
      LOCKED            : out    std_logic
   );
end component;
```

The **VHDL code** for the FM modulation and transmission is structured in several blocks of code.

Variables declaration defines some initial values that are used in the main process to modulate the wave:

- *tx_freq* and *tx_band* are constant values used to define the carrier features and the transmission band frequency found out through this formula (desired frequency)/320MHz*2^32;
- *sample_rate_max* is the parameter for the notes duration, this parameter is computed using this formula 320MHz*desired duration/4

```
-- The constants are calculated as (desired freq)/320Mhz*2^32
constant tx_freq           : integer := 91000 *134217728/10000;   --KHz of the carrier wave
constant tx_band           : integer := 75    *134217728/10000;   --KHz of the transmitted band

signal beep_tone           : integer := 0;
signal beep_tone_hi_lo     : std_logic := '0';
signal clk320              : std_logic;
signal rst_in              : std_logic;
signal locked_int          : std_logic;
signal phase_accumulator   : unsigned (31 DOWNTO 0) := (others => '0');
signal next_sample_cnt     : integer := 0;
signal sample_rate_max     : integer := 40000000;  -- 320Mhz*0.5s/4  (tone duration = 0.5s) (the base note is written 4 time in the file)
signal beep_tone_max       : integer := 0;
signal mem_address         : std_logic_vector(8 DOWNTO 0) := (others => '0');
signal mem_output          : std_logic_vector(7 DOWNTO 0);
```

The antenna is a wire connected to a digital pin and mapped on the most significant bit of the *phase_accumulator*.

```vhdl
begin
    -- CONSTANT MAPPING
    antenna <= std_logic(phase_accumulator(31));
    rst_in <= '0';
```

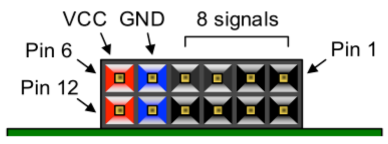
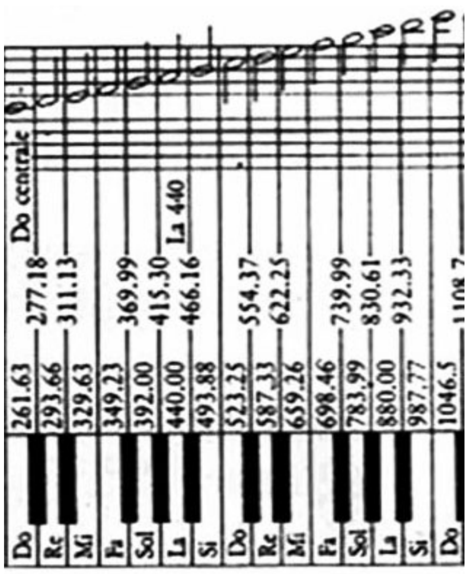


Figure 16. Pmod diagram.

The process is clock driven, in the first block reads from the memory the current note code and decodes it using a case statement.

```vhdl
-- MAIN PROCESS
process(clk320)
begin
   if rising_edge(clk320) then
      -- Read sample from ROM
      if next_sample_cnt >= sample_rate_max then
         mem_address <= std_logic_vector(unsigned(mem_address) + 1);
         -- Set tone sound 302MHz/(2*f_nota)
         case mem_output(3 downto 0) is
            when x"0" =>
               beep_tone_max <= 0;      -- No sound
            when x"1" =>
               beep_tone_max <= 613026; -- 261 Hz Do
            when x"2" =>
               beep_tone_max <= 546075; -- 293 Hz Re
            when x"3" =>
               beep_tone_max <= 484848; -- 330 Hz Mi
            when x"4" =>
               beep_tone_max <= 458452; -- 349 Hz Fa
            when x"5" =>
               beep_tone_max <= 408163; -- 392 Hz Sol
            when x"6" =>
               beep_tone_max <= 363636; -- 440 Hz La
            when x"7" =>
               beep_tone_max <= 323886; -- 494 Hz Si
            when x"8" =>
               beep_tone_max <= 305927; -- 523 Hz Do1
            when x"9" =>
               beep_tone_max <= 272572; -- 587 Hz Re1
            when x"A" =>
               beep_tone_max <= 242792; -- 659 Hz Mi1
            when x"B" =>
               beep_tone_max <= 229226; -- 698 Hz Fa1
            when x"C" =>
               beep_tone_max <= 204081; -- 784 Hz Sol1
            when x"D" =>
               beep_tone_max <= 181818; -- 880 Hz La1
            when x"E" =>
               beep_tone_max <= 162107; -- 987 Hz Si1
            when x"F" =>
               beep_tone_max <= 152963; -- 1046 Hz Do2
            when others =>
               beep_tone_max <= 0;      -- No sound
         end case;
```



The core of the modulation is the increment of the *phase_accumulator* in order to change its MSB and then the antenna signal. If the value read from the memory is zero, the carrier is transmitted, otherwise a wave that changes frequency periodically in order to generate the tone, is transmitted.

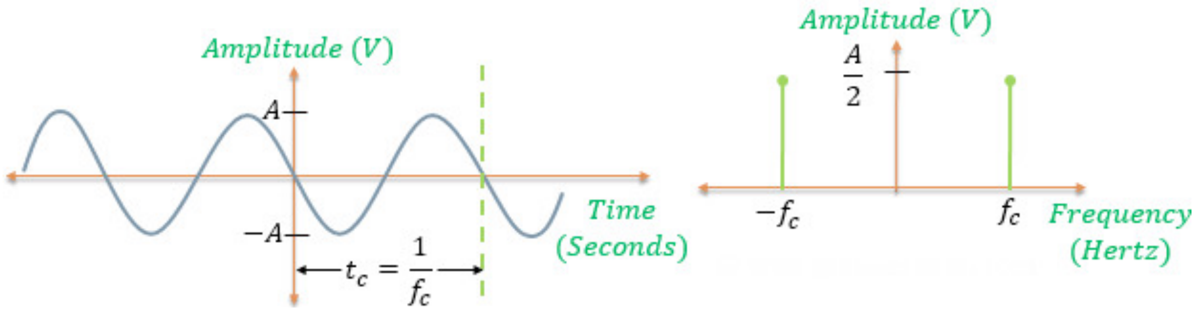
```vhdl
--Waveform generation
if mem_output(3 downto 0) = x"0" then
    phase_accumulator <= phase_accumulator + tx_freq;   -- No suond
else
    if beep_tone_hi_lo = '0' then
        phase_accumulator <= phase_accumulator + tx_freq + tx_band;
    else
        phase_accumulator <= phase_accumulator + tx_freq - tx_band;
    end if;
end if;

        --Tone waveform control
        beep_tone <= beep_tone + 1;
        if beep_tone >= beep_tone_max then
            beep_tone_hi_lo <= not beep_tone_hi_lo;
            beep_tone <= 0;
        end if;
```

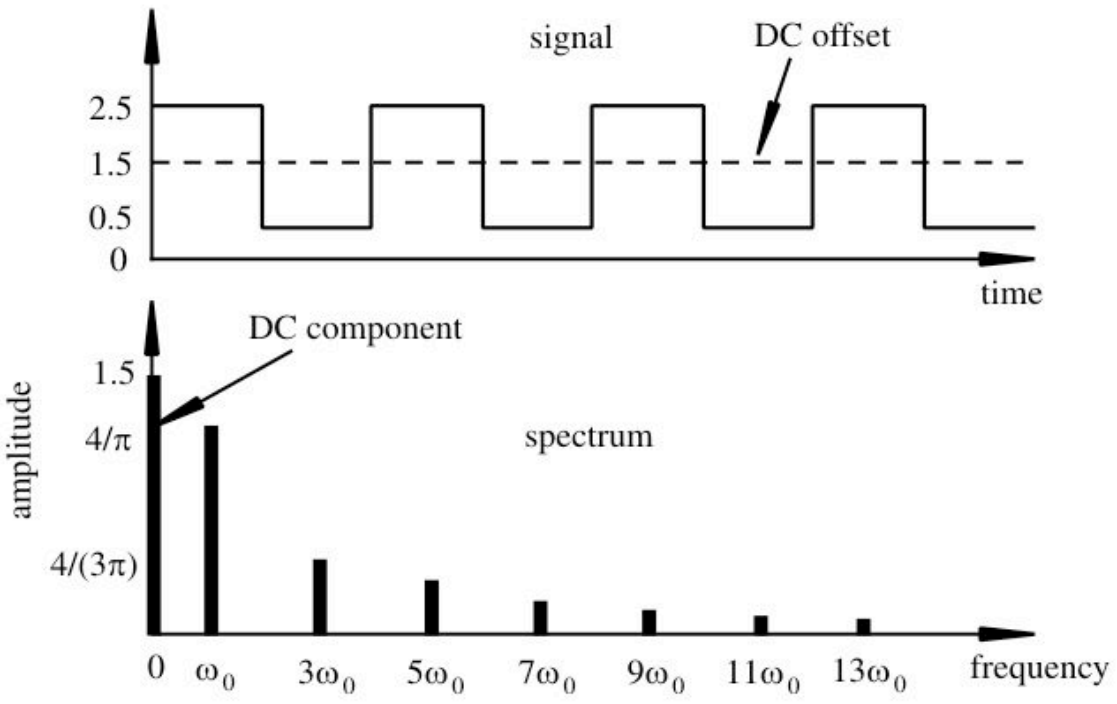
Conclusions

The waves sent into the ether are square waves and they have some issues:

- with an high sampling as audio track, the waves received are so much distorted that, as the result, we don't hear anything. The ideal situation would be to use sine waves generated by DAC, but the board doesn't mount a fast enough one to get 91MHz.
- transmission is not only heard at 91MHz, but also to the frequencies of the secondary harmonics because we don't implement band-pass filter.

However, we were able to send the "Ode to Joy" because its notes are very long.


Amplitude (V)
A
−A
$t_c = \frac{1}{f_c}$
Time (Seconds)
Amplitude (V)
$\frac{A}{2}$
$-f_c$
f_c
Frequency (Hertz)


signal
DC offset
2.5
1.5
0.5
0
time
DC component
amplitude
1.5
4/π
4/(3π)
spectrum
0
ω_0
$3\omega_0$
$5\omega_0$
$7\omega_0$
$9\omega_0$
$11\omega_0$
$13\omega_0$
frequency