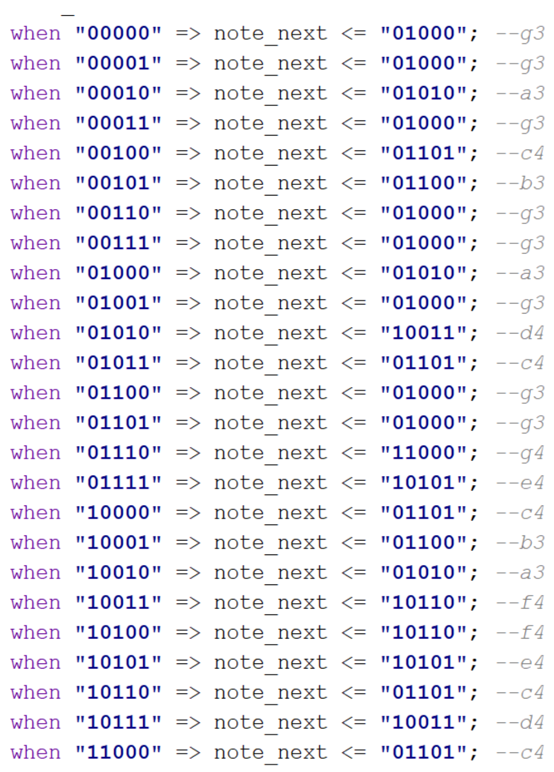
Lab 6: Electric piano design using Xilinx FPGAs, Part II
Lab Completed: February 24th
Wednesday Lab
Jason Jin

The video of this electric piano design is attached in a separate file uploaded on courseworks.

Name of the song: Happy Birthday song



```
when "00000" => note_next <= "01000"; --g3
when "00001" => note_next <= "01000"; --g3
when "00010" => note_next <= "01010"; --a3
when "00011" => note_next <= "01000"; --g3
when "00100" => note_next <= "01101"; --c4
when "00101" => note_next <= "01100"; --b3
when "00110" => note_next <= "01000"; --g3
when "00111" => note_next <= "01000"; --g3
when "01000" => note_next <= "01010"; --a3
when "01001" => note_next <= "01000"; --g3
when "01010" => note_next <= "10011"; --d4
when "01011" => note_next <= "01101"; --c4
when "01100" => note_next <= "01000"; --g3
when "01101" => note_next <= "01000"; --g3
when "01110" => note_next <= "11000"; --g4
when "01111" => note_next <= "10101"; --e4
when "10000" => note_next <= "01101"; --c4
when "10001" => note_next <= "01100"; --b3
when "10010" => note_next <= "01010"; --a3
when "10011" => note_next <= "10110"; --f4
when "10100" => note_next <= "10110"; --f4
when "10101" => note_next <= "10101"; --e4
when "10110" => note_next <= "01101"; --c4
when "10111" => note_next <= "10011"; --d4
when "11000" => note_next <= "01101"; --c4
```

List of notes played

The comment section on the right hand side of each line represents the notes played sequentially from the FPGA board. The codes on the left will be explained later so this can be ignored for now.

Code (only the sections edited attached):

```
 -- Signals
 signal CLK         : std_logic;                       -- 50MHz clock after DCM and BUFG
 signal CLK0        : std_logic;                       -- 50MHz clock from pad
 signal CLK_BUF     : std_logic;                       -- 50MHz clock after IBUF
 signal GND         : std_logic;
 signal RST         : std_logic;
 signal PB          : std_logic_vector(3 downto 0);    -- Pushbuttons after ibufs
 signal digit_l     : std_logic_vector(3 downto 0);    -- 7-seg digit MUX before obuf
 signal switch      : std_logic_vector(7 downto 0);    -- Toggle switches after ibufs
 signal led         : std_logic_vector(7 downto 0);    -- LEDs after ibufs
 signal seg_l       : std_logic_vector(7 downto 0);    -- 7-seg segment select before obuf.
 signal one_mhz     : std_logic;                       -- 1MHz Clock
 signal one_mhz_1   : std_logic;                       -- pulse with f=1 MHz created by divider
 signal two_hz      : std_logic;                       -- 2Hz clock()
 signal clk_10k_1   : std_logic;                       -- pulse with f=10kHz created by divider
 signal div         : std_logic_vector(15 downto 0);   -- variable clock divider for loadable counter
 signal note_in     : std_logic_vector(4 downto 0);    -- output of user interface. Current Note
 signal note_next   : std_logic_vector(4 downto 0);    -- Buffer holding current Note
 signal note_sel    : std_logic_vector(3 downto 0);    -- Encoding of switches.
 signal div_1       : std_logic;                       -- 1MHz pulse
 signal sound       : std_logic;                       -- Output of Loadable Clock Divider. Sent to Speaker if note is playing.
 signal SPK         : std_logic;                       -- Output for Speaker fed to OBUF
 signal cap_Index   : std_logic_vector(4 downto 0);    -- index()
```

Added signals: two_hz & cap_Index

```
-- Divide 50Mhz to 1Mhz clock
DIV_1M : clk_dvd
    port map ( CLK      => CLK,
               RST      => RST,
               DIV      => x"0032",   -- 50
               EN       => '1',
               CLK_OUT  => one_mhz,
               ONE_SHOT => one_mhz_1
             );

DIV_two_hz : clk_dvd
    port map ( CLK      => CLK,
               RST      => RST,
               DIV      => x"09c4",
               EN       => clk_10k_1,
               CLK_OUT  => open,
               ONE_SHOT => two_hz
             );
```

```vhdl
-- User Interface
note_in <= note_next;

process (CLK,RST) begin
    if (RST = '1') then
        note_next <= (others => '0');
        cap_Index <= (others => '0');
    elsif (CLK'event and CLK = '1') then
        if (switch(0) = '1') then
            if (two_hz = '1') then
                case cap_Index is
                    when "00000" => note_next <= "01000"; --g3
                    when "00001" => note_next <= "01000"; --g3
                    when "00010" => note_next <= "01010"; --a3
                    when "00011" => note_next <= "01000"; --g3
                    when "00100" => note_next <= "01101"; --c4
                    when "00101" => note_next <= "01100"; --b3
                    when "00110" => note_next <= "01000"; --g3
                    when "00111" => note_next <= "01000"; --g3
                    when "01000" => note_next <= "01010"; --a3
                    when "01001" => note_next <= "01000"; --g3
                    when "01010" => note_next <= "10011"; --d4
                    when "01011" => note_next <= "01101"; --c4
                    when "01100" => note_next <= "01000"; --g3
                    when "01101" => note_next <= "01000"; --g3
                    when "01110" => note_next <= "11000"; --g4
                    when "01111" => note_next <= "10101"; --e4
                    when "10000" => note_next <= "01101"; --c4
                    when "10001" => note_next <= "01100"; --b3
                    when "10010" => note_next <= "01010"; --a3
                    when "10011" => note_next <= "10110"; --f4
                    when "10100" => note_next <= "10110"; --f4
                    when "10101" => note_next <= "10101"; --e4
                    when "10110" => note_next <= "01101"; --c4
                    when "10111" => note_next <= "10011"; --d4
                    when "11000" => note_next <= "01101"; --c4
                    when others  => note_next <= "00000";
                end case;
                if (cap_Index = "11000") then
                    cap_Index <= "00000";
                else
                    cap_Index <= cap_Index + 1;
                end if;
            end if;
            end if;
        else
            case switch is
                when "10000000" => note_sel <= "0001"; -- C
                when "01000000" => note_sel <= "0011"; -- D
                when "00100000" => note_sel <= "0101"; -- E
                when "00010000" => note_sel <= "0110"; -- F
                when "00001000" => note_sel <= "1000"; -- G
                when "00000100" => note_sel <= "1010"; -- A
                when "00000010" => note_sel <= "1100"; -- B
                when others     => note_sel <= "0000";
            end case;

             -- Sharp -- Add one.  PB(3) is the octave key.
            if (PB(2) = '1') then
                note_next <= PB(3) & note_sel + 1;
               -- Flat --  Minus one.
            elsif (PB(1) = '1') then
                note_next <= PB(3) & note_sel - 1;
            else
                note_next <= PB(3) & note_sel;
            end if;
            cap_Index <= "00000";
        end if;
    end if;
end process;
```

Mechanism / Notes:

Since each switch represents a certain note in piano.vhd such as (Switch 7 -> C, Switch 6 -> D … Switch 2 -> A, Switch 1 -> B), Switch 0 can be used in another 'if' function to act as a trigger playing the notes. And another signal variable named "cap_Index" with std_logic_vector(4 down to 0) is declared to set as a pointer to the first note of the song so that when the number of the notes (in this case 24 in decimal) becomes "11000" then the next note to be played points back to the first note of the song. The notes are generated by referring to note_gen.vhd and how they are modified is explained in more detail in the seven-segment display section below.

Timing of the notes:

For the clock divider, I created more divs so that there are more counts until clk_out toggles. And the reason why another two-hz clock created above has a DIV set to 2500 is using the equation from the clock divider equation under clk_dvd.vhd. Since all the notes are in the range of roughly 200 hz from the piano notes figure in lab 5 manual, the div is set to the right value. Lastly, from 1MHz to 10k, the div is increased to 150 to lengthen one_shot. In other words, this increases the one cycle period of the input clock that results in playing longer notes.

Seven-segment display:

When 5 bits are first passed in as in note_gen.vhd then the least significant 4 bits represent the note and the last one most significant bit represents the octave. So this helps understand how the notes are played in octaves in the song. In seven_seg.vhd, it takes in 5 bits and decodes it in 9 bits. Inside 9 bits, 4 bits represent the first segment display(A~G) notes and another 4 bits represent octaves (3 or 4) and the last 1 bit is whether an extra dot is added to the display or not.

User interface walkthrough:

In this happy birthday song, note_next and cap_Index are initialized to 0. And in the next clock cycle, the 'if' statement is processed to see if the switch 0 is turned on. So when this particular switch on an FPGA board (the switch toggled in the video) is not turned on, it would just normally act as just an electric piano playing different notes when a different switch is turned on each time. Then as soon as I toggle the switch the 'if' statement is processed and another 'if' statement of two_hz that checks the range of the note frequencies is processed. If all of these statements are true, then the note_next is input with the first note of the song (G3). And continuously as the clock cycle is on, the note_next is updated every time with the next song of the note. Lastly, cap_Index works like a count where it repeats back to the case statement above when the last note is reached. So unless the switch 0 is toggled back to 0 the song will be played over and over.