Lab 5 Report

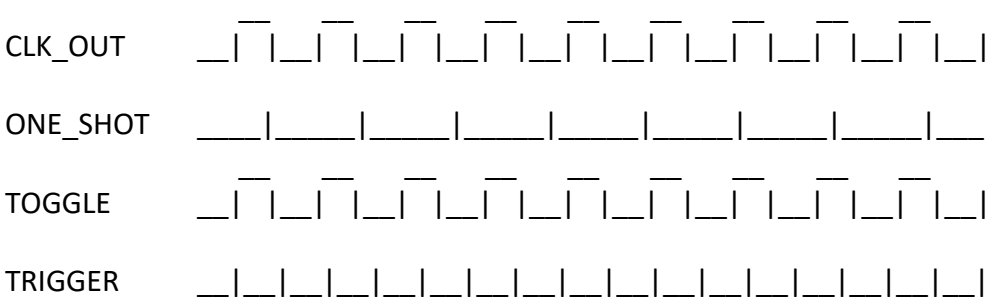
2.1

```
---------------------------------------------------------------------------------------------------
            note_in <= note_next;
            process (CLK,RST) begin
               if (RST = '1') then
                  note_next <= (others => '0');
               elsif (CLK'event and CLK = '1') then
                  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;

               end if;
            end process;
---------------------------------------------------------------------------------------------------
```

Each note corresponds to a single switch flipped on the board of switches. Each flipped switch corresponds to a single bit turned on in the 8bit logic vector "switch". The notes are encoded from the first/MSB switch (signal = '10000000' -> first switch is flipped --> note C selected -> encoded as 4bit_logic_vector "0001"), second switch flipped (signal = '01000000' --> note D selected -> "0011"), up to switch signal = '00000010' --> note B selects note value 1100. All switches, except the LSB switch in the array, are orthogonal in time. For all other combinations of input switch signals, the note_selected is none, "0000".

Once a note is selected, other parameters are checked (e.g. bushbuttons) to sharpen, flatten, or set the note's octave. If PB(2)='1' (pushbutton 2 is pressed), note_next <= PB(3) & note_sel + 1 (assign the next note the value of pb(3), pushbutton 3, the octave, and the selected note + 1, to sharpen the note). If PB(1)='1' (pushbutton 1 is pressed), note_next <= PB(3) & note_sel - 1 (assign the next note the value of the octave and note selected - 1, to flatten the note)
The notes are encoded by flipping the switches (one at a time) to select the notes,
and the different pushbuttons are used to sharpen, flatten, and select the note's octave.
The encoding of all notes (encoded with the switches and pushbuttons), is represented as a 5 bit log vector (2^5=32 > 20+ total notes)

2.2



2.3

FPGA piano output frequency
freq(div=10^0) = 5*10^5
freq(div) = (5*10^5) (div)^-1

Note	Next_div	FPGA Piano Output Freq	True note frequency	Error(Hz)
C3	x0EEE=3822	130.82	130.81	0.01
C3#	x0E18=3608	138.58	138.59	0.01
D3	x0D4E=3406	146.80	146.83	0.03
D3#	x0C8E=3214	155.57	155.56	0.01
E3	x0BDA=3034	164.80	164.81	0.01
F3	x0B30=2864	174.58	174.61	0.03
F3#	x0A8E=2702	185.05	185.00	0.05
G3	x09F7=2551	196.00	196.00	0.00
G3#	x0968=2408	207.64	207.65	0.01
A3	x08E1=2273	219.97	220.00	0.03
A3#	x0861=2145	233.10	233.08	0.02
B3	x07E9=2025	246.91	246.94	0.03

2.4
Study and explain implementation of seven_seg.vhd

Seven_seg.vhd controls the four-digit seven segment display.
Seven_seg outputs a binary encoded number, seg, that represents the transistor values required to display the current digit. Only a single digit of the binary number/note can be scanned/changed/output at a time.

Seven_seg takes in a complete note (encoded with note, octave, sharp, flat, etc.) and assigns the segment_buffer a value depending on the note. All possible notes are encoded as a 5bit logic vector and stored as a seg_buf (segment buffer of size 9, 8 down to 0).

Seven_seg checks the current digit being displayed from the note in cur_digit, out of four possible digit displays. Cur_digit is incremented on clk'event and clk='1' and scan_en='1'. Depending on the value of cur_digit (current digit), Digit_now and point_now are assigned values from the seven-segment buffer that encodes the note's current digit. Digit_now and point_now are finally used to output the seven-segment (8 bits, 7 down to 0) representation of the current digit displayed.

3.3

```
--------------------------------------------------------------------------------------------
//System Reset
        pb_in(0) <= '1';
        wait for 50 ns;
        pb_in <= "0000";
        wait for 100 ns;

//Set first octave with pushbutton 3
//Select 2nd switch, in hex x"40":
// hex(x40) = binary(0100 0000) -> Note D -> clock_div=x6a7 -> selected note = D4
// reset system for next note.
        pb_in(3) <= '1';
        switch_in <= x"40";
        wait for 4ms;
         pb_in(0) <= '1';
        wait for 50 ns;
        pb_in <= "0000";
        wait for 100 ns;


//Set first octave
```

```
//Select 6th switch, in hex x”04”:
// hex(x04) = binary(0000 0100) -> Note A -> clock_div=x470 -> selected note=A4
// reset system for next note.
        pb_in(3) <= '1';
        switch_in <= x"04";
        wait for 4ms;
        pb_in(0) <= '1';
        wait for 50 ns;
        pb_in <= "0000";
        wait for 100 ns;

//Set first octave
//Select 3rd switch, in hex x”20”:
// hex(x20) = binary(0010 0000) -> Note E -> clock_div=x5ed -> selected note=E4
// reset system for next note.
        pb_in(3) <= '1';
        switch_in <= x"20";
        wait for 4ms;
        pb_in(0) <= '1';
        wait for 50 ns;
        pb_in <= "0000";
        wait for 100 ns;

//Set first octave
//Select 5th switch, in hex x”08”:
// hex(x08) = binary(0000 1000) -> Note G -> clock_div=x4fb -> selected note=G4
// reset system for next note.
        pb_in(3) <= '1';
        switch_in <= x"08";
        wait for 4ms;
        pb_in(0) <= '1';
        wait for 50 ns;
        pb_in <= "0000";
        wait for 100 ns;

//Set first octave
//Sharpen Note with pushbutton 2
//Select 6th switch, in hex x”40”:
// hex(x40) = binary(0100 0000) -> Note D -> clock_div=x647-> selected note=D4#
// reset system for next note.
        pb_in(3) <= '1';
        pb_in(2) <= '1';
        switch_in <= x"40";
        wait for 4ms;
```

```
        pb_in(0) <= '1';
        wait for 50 ns;
        pb_in <= "0000";
        wait for 100 ns;

//Set first octave
//Select 4th switch, in hex x"10":
// hex(x10) = binary(0001 0000) -> Note  B -> clock_div=x3f4-> selected note=B4
// reset system.
        pb_in(3) <= '1';
        switch_in <= x"10";
        wait for 5ms;
        pb_in(0) <= '1';
        wait for 50 ns;
        pb_in <= "0000";
        wait for 100 ns;
```

Each note is played for 4ms. The input switch value can be seen in switch_in.
The value of switch_in corresponds to a note being played on the array of 8 switches (7 of which are used to encode a note). Each note requires the speaker to oscillate at a certain frequency to play the note. In order to play the note, the clock is divided by a factor (div), which drives the speaker at the required frequency. Each note has a different value of clock div[15:0]. Smaller input switch/hex values, e.g. X"04", correspond to smaller div values (clock is divided by smaller values), which create higher frequency notes. X"04" -> Note A is selected -> High frequency. Conversely, larger input switch/hex values, e.g. X"40", correspond to larger clock_divs, lower frequencies. The contrast in small and large frequency notes, and their clock divs, can be seen in SPK_N and div[15:0]. Sharps and flats are encoded in pb_in. D4# is played, which can be seen with pb_in= c (pushbutton 2 and 3 are pressed to encode octave and sharp) and div=647 (smaller div value in contrast to D4's div=6a7, corresponds to slightly higher frequency).