Digital Electronics Lab 6 Report

Ming Gong (mg4264), Xuanyi Wu (xw3036)

Testing

Custom Song

We implemented the first 16 notes of Jay Chou's "Dao Xiang". Below are the list of notes:
C3 -- C3 D3# D3 F3 E3 D3 -- C3 D3# E3 E3 E3 C3 --

Interface

Below are our edits to `piano.vhd` to automatically play a song.
Here's a link to the full code on GitHub:

1. Variables And Constants

We declare a few extra `signal`s and constants to store the state information, and the actual notes. These signals are declared under `acrhitecture Behavioral of piano`
There are 3 new signals:

- `song_index`: index of the current note in the note array
- `song_done`: indicates playing is done
- `song_tick`: indicates a new note needs to be played

and 2 new constants:

- `SONG_LENGTH`: number of notes in the song
- `song_notes`: the actual note mapping

```
signal song_index  : integer := 0;          -- mote index
signal song_done   : std_logic := '0';      -- done playing
signal song_tick   : std_logic;             -- signal for new note

-- hard-coded note array
type note_array_t is array (0 to 15) of std_logic_vector(4 downto 0);
constant SONG_LENGTH : integer := 16;
constant song_notes : note_array_t := (
    0  => "00001",  -- C
    1  => "00000",  -- C (cont)
    2  => "00001",  -- C
    3  => "00100",  -- D#
```

```
    4  => "00011",  -- D
    5  => "00110",  -- F
    6  => "00101",  -- E
    7  => "00011",  -- D
    8  => "00000",  -- D (cont)
    9  => "00001",  -- C
    10 => "00100",  -- D#
    11 => "00101",  -- E
    12 => "00101",  -- E
    13 => "00101",  -- E
    14 => "00001",  -- C
    15 => "00000"   -- C (cont)
);
```

2. User Interface Process

Now we modify the user interface part
We first define a clock divider `tempo_div`, that plays notes at a designated tempo:

- The 10 kHz `clk_10k_1` is divided to 2 Hz `song_tick`

```
-- Modified User Interface , with auto-play
-- Clock divider for song tempo (~0.5s)
tempo_div : clk_dvd
    port map (
        CLK      => CLK,
        RST      => RST,
        DIV      => x"09C4",  -- Tempo control
        EN       => clk_10k_1,
        CLK_OUT  => open,
        ONE_SHOT => song_tick -- Tempo output
    );
```

Next, the process sets `note_next` to the values hard-coded in `song_notes`. It is a state machine that keeps track of the current note

- `note_in`, which is sent to other blocks, is the output of the user interface.
- For every 0.5 seconds (`song_tick` = 1), it advances a note.
- If `RST` = 1, all relevant variables are reset to 0.
- If the song finishes, the index is reset, and `song_done` = 1. The player stops and waits for reset.

```
-- Assign note to note generator
note_in <= note_next;
process (CLK, RST) begin
    if RST = '1' then
        song_index <= 0;
        note_next  <= "00000";
        song_done  <= '0';
    elsif rising_edge(CLK) then
        if song_done = '0' and song_tick = '1' then
            if song_index < SONG_LENGTH then
                note_next <= song_notes(song_index);
                song_index <= song_index + 1;
            else
                note_next <= "00000";  -- rest at end
                song_done <= '1';
            end if;
        end if;
    end if;
end process;
```