ECEN302 : Integrated Digital Electronics
Lab 7 Submission : TCL scripts and Packaging IP

Daniel Eisen : 300447549

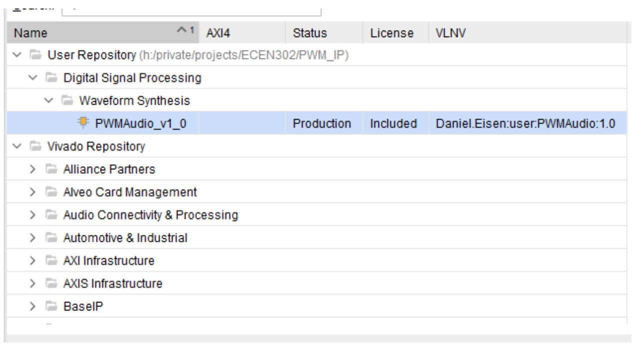
October 6, 2020

1 Overview

The use of pre-existing IP blocks in the in HDL design is extremely time saving and in many to most cases is necessary when creating a more complex design with more external dependencies or just utilises a lot of basic reused design blocks. Likewise the IP block system can be made use of to package a preexisting design into a library that can be loaded into the user repository and dropped into block designs or directly instanced without having to redo and grunt work.

2 Methodology

To do this, we initially setup the basic project. Vivado allows for a project to be exported not only as an archive are other larger containers but as set TCL script file and dependency files that can setup project from launch. THis allows for more efficient storage space, can bypass PATH and directory issues upon transfer and just generally can allow for greater flexibility. This is what was used to launch and setup a PWM audio driver project.
The HDL design worked by varying the time step that read through a sine wave LUT to be fed into a PWD generator. This is what effectively changes the frequency and thus pitch of the outputted (filter smoothed) signal/sound.
This project and its 2 HDL modules can be packaged into it own IP block, and exported into a directory that we then imported, add to the user repository and were able to then utilise the generated block as a part of larger design.
Approaching the design this way enables this relatively simple design to grow easily into a larger and more complex form. This for example is a setup for a larger embedded softcore processor project where the PWM audio IP will be utilised as a peripheral out module to a MicroBlaze uC and driven to play much more complex sequences that can be generated with DIP switches.



Name	AXI4	Status	License	VLNV
User Repository (h:/private/projects/ECEN302/PWM_IP)				
Digital Signal Processing				
Waveform Synthesis				
PWMAudio_v1_0		Production	Included	Daniel.Eisen:user:PWMAudio:1.0
Vivado Repository				
Alliance Partners				
Alveo Card Management				
Audio Connectivity & Processing				
Automotive & Industrial				
AXI Infrastructure				
AXIS Infrastructure				
BaseIP				

Appendix

PWMAudio

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity PWMAudio is
    Port ( CLK100MHZ : in STD_LOGIC;
           AUD_PWM : out STD_LOGIC;
           AUD_SD : out STD_LOGIC;
           SW : in STD_LOGIC_VECTOR(8 downto 0));
end PWMAudio;

architecture Behavioral of PWMAudio is

component PWMDriver is
    Port ( clk_100 : in STD_LOGIC;
           pwm_level : in STD_LOGIC_VECTOR (9 downto 0);
           pwm_out : out STD_LOGIC);
end component;

COMPONENT dist_mem_gen_0
  PORT (
    a : IN STD_LOGIC_VECTOR(9 DOWNTO 0);
    spo : OUT STD_LOGIC_VECTOR(9 DOWNTO 0)
  );
END COMPONENT;

signal sine_count : unsigned(16 downto 0) := (others => '0');
signal sine_freq : unsigned(8 downto 0) := (others => '0');

signal lut_addr : unsigned(9 downto 0) := (others => '0');
signal lut_out : std_logic_vector(9 downto 0) := (others => '0');

signal pwm_int : std_logic := '0';

begin

pd0 : PWMDriver
port map(
    clk_100 => CLK100MHZ,
    pwm_level => lut_out,
    pwm_out => pwm_int
);

dmg0 : dist_mem_gen_0
  PORT MAP (
    a => std_logic_vector(lut_addr),
    spo => lut_out
);

AUD_SD <= '1';
AUD_PWM <= '0' when pwm_int = '0' else 'Z';

sine_freq <= unsigned(SW);

process(CLK100MHZ)
begin
    if rising_edge(CLK100MHZ) then
        if sine_count = sine_freq then
            lut_addr <= lut_addr + 1;
            sine_count <= (others => '0');
        else
            sine_count <= sine_count + 1;
        end if;
    end if;
end process;

end Behavioral;
```

PWMDriver

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity PWMDriver is
    Port ( clk_100 : in STD_LOGIC;
           pwm_level : in STD_LOGIC_VECTOR (9 downto 0);
           pwm_out : out STD_LOGIC);
end PWMDriver;

architecture Behavioral of PWMDriver is

signal count : unsigned(9 downto 0) := (others => '0');
signal compare : unsigned(9 downto 0) := (others => '0');

begin

process(clk_100)
begin
    if rising_edge(clk_100) then
        if count = 0 then
            compare <= unsigned(pwm_level);
            count <= count + 1;
        elsif count = 1023 then
            count <= (others => '0');
        else
            count <= count + 1;
            if count < compare then
                pwm_out <= '1';
            else
                pwm_out <= '0';
            end if;
        end if;
    end if;
end process;

end Behavioral;
```

HDL Wrapper

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
library UNISIM;
use UNISIM.VCOMPONENTS.ALL;

entity design_1_wrapper is
  port (
    AUD_PWM : out STD_LOGIC;
    AUD_SD : out STD_LOGIC;
    SW : in STD_LOGIC_VECTOR ( 8 downto 0 );
    reset : in STD_LOGIC;
    sys_clock : in STD_LOGIC
  );
end design_1_wrapper;

architecture STRUCTURE of design_1_wrapper is
  component design_1 is
  port (
    sys_clock : in STD_LOGIC;
    reset : in STD_LOGIC;
    AUD_PWM : out STD_LOGIC;
    AUD_SD : out STD_LOGIC;
    SW : in STD_LOGIC_VECTOR ( 8 downto 0 )
  );
  end component design_1;
begin
design_1_i: component design_1
     port map (
      AUD_PWM => AUD_PWM,
      AUD_SD => AUD_SD,
      SW(8 downto 0) => SW(8 downto 0),
      reset => reset,
      sys_clock => sys_clock
    );
end STRUCTURE;
```