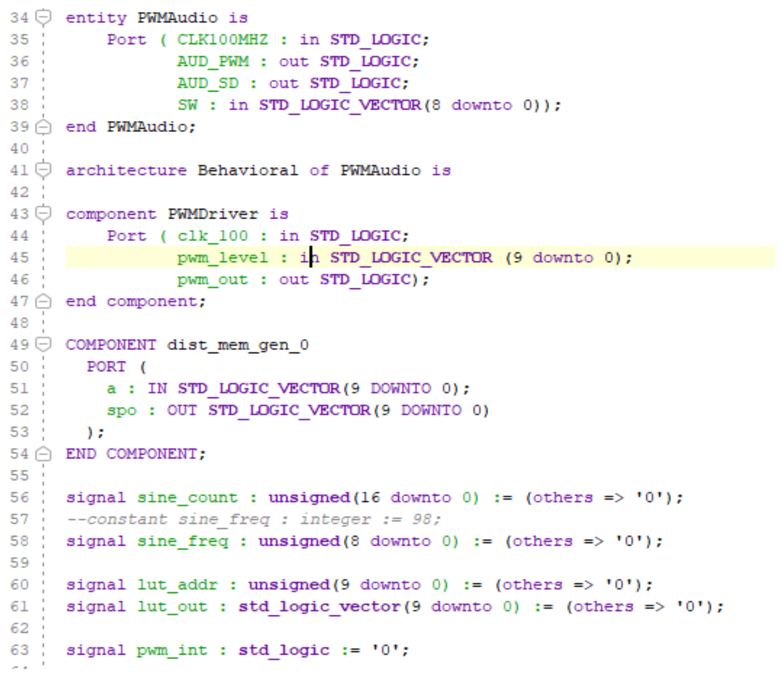
1. Introduction

Using the IP integrator (IPI) of Vivado to design a softcore Microblaze is limited to using it as a microprocessor. This can be very limiting to complex SoC designs and could be further upgraded. This lab aims to create an IP which produces a PWM audio tone generator so that future IP integrated circuits such as one with the Microblaze softcore processor can interact with the PWM output capabilities of the FPGA.

2. IP integrator

The digital circuit for the PWM audio was provided as part of the lab. To make future iterations easier, this digital circuit was saved as a Tool Command Language file (Tcl) so that Vivado could import it without having to deal with the previous versions. When this script is run, Vivado can upgrade the IP to fit the current version as shown in the VHDL code in Figure 2.1. Once upgraded, a bit stream can then be generated and uploaded on the FPGA.



```
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');
--constant sine_freq : integer := 98;
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';
```

Figure 2.1: Digital circuit of PWM generator, created by the Tcl script

Based on input switch combinations on the FPGA board, the digital circuit outputs a PWM. By averaging a PWM duty cycle into a DC voltage, the desired frequency can be produced when the DC voltage is output. As a result, a combination of input switches can be correlated with the desired frequency. As this Tcl script only has 9-bit switches, only the lower 9 switches may set the tone. This PWM generator consists of 2 parts, a sine wave generator which uses a counter to access a look-up table called a direct digital synthesiser (DDS), This sinewave then generates a PWM output and is smoothed out using analogue filters.

To prime the circuit for IP packaging, the constraints file must be tailored to allow the generated PWM to be routed to its IO port. For versatile routing when imported as a component of a larger digital circuit, the **constraints file** must be deleted since this digital circuit is going to be packaged into an IP. Consequently, other FPGAs with different IO mapping such as Basys 3 can use this IP block. However, critical information such as I/O pins and voltage levels (3.3 Vcc) must be checked before packaging.

3. IP packaging

This digital circuit can now be packaged as an IP and will appear in the designated folder during the compilation process. This IP block may be added to the IP catalogue for importation as part of another circuit. As illustrated in Figure 3.1, the PWM IP was imported through the IPI interface along with the clocking wizard.

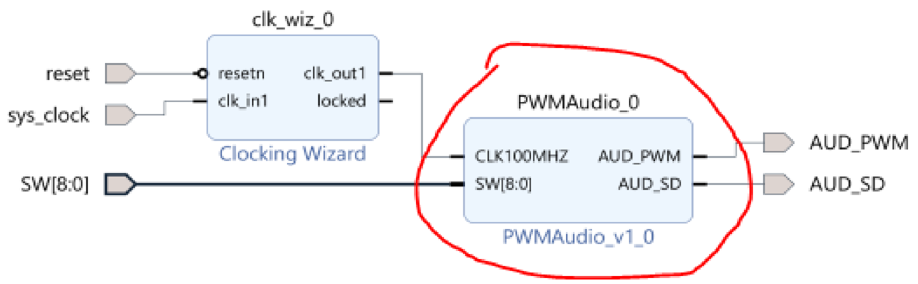


Figure 3.1: Imported PWMAudio_0 in an IP integrated circuit.

The 100 Mhz clock input on the PWM_Audio block was connected to a clock module, this newly instantiated clock will provide the PWM generator with sufficient clock cycle to function. Additional pins for the clock were created, such as “reset” and “sys_clock”. In addition to the output pins for the PWM_Audio block, input ports with 8 to 0 bits were made under the names “SW”, “AUD_SD'' and “AUD_PWM”. Although some of the routings can be achieved through auto-routing block design. This IPI would then need to be validated to ensure the correct routing of the digital circuit.

In order to generate this block design, an HDL wrapper must be created, this will allow the block design to the block design into a source file for Vivado. To synthesise the new IP and HDL, this IPI circuit will need to be synthesised globally, thereby removing the previous error message. In the new IPI project designed for Nexys 4 DDR, the **constraints file** can be added back into the project so that it matches the IP block requirements for the PWM IP block. As a final step, the bitstream can be generated and uploaded to the FPGA. In a similar manner to Section 2, the FPGA will output a PWM based on the current configuration of switches. However, this time it was achieved through the use of a custom-packaged IP.

4. Conclusion

In conclusion, packaging IP for the PWMAudio allows for scalability by providing design blocks for different FPGAs instead of being limited to the one on which it was developed. As a result of IP packaging capabilities, developers can protect their intellectual properties by preventing future users from accessing the content of the block. By importing this IP in the future, the MicroBlaze softcore processor will be able to communicate with the FPGA’s PWM output.