library ieee;

use ieee.std\_logic\_1164.all;

use ieee.numeric\_std.all;

entity ClockGeneration is

port(

clk : in std\_logic; -- 50 MHz frequency (20 ns period)

adc : in std\_logic\_vector(7 downto 0);

reset : in std\_logic;

clock\_out : out std\_logic

);

end ClockGeneration;

architecture arch of ClockGeneration is

signal freq\_out : integer; -- Frequency of clk\_out

signal divisor : integer;

signal total : INTEGER;

signal two : integer;

constant freq\_in : integer := 50000000; -- Frequency of input clk

signal counter : integer range 0 to 50000000;

type st is (clock\_out\_1, clock\_out\_0); --, capture

signal states : st;

begin

-- Clock enable with variable frequency output

process(clk,states, adc, reset)

begin

divisor <= freq\_in / freq\_out;

total <= freq\_out + divisor;

two <= total / 2;

if adc = X"FF" then

freq\_out <= 1574;

else

freq\_out <= (((1750-500)/255)\*to\_integer(unsigned(adc))) + 503;

end if;

if (reset = '1') then

clock\_out <= '0';

counter <= 0;

states <= clock\_out\_0;

elsif rising\_edge(clk) then

counter <= counter + 1;

-- adc2 <= adc;

if (counter = 50000000) then

counter <= 0;

states <= clock\_out\_0;

end if;

case states is

when clock\_out\_0 =>

clock\_out <= '0';

if (two <= counter) then

clock\_out <= '1';

counter <= 0;

states <= clock\_out\_1;

end if;

when clock\_out\_1 =>

clock\_out <= '1';

if (total <= counter) then

clock\_out <= '0';

counter <= 0;

states <= clock\_out\_0;

end if;

end case;

end if;

end process;

end arch;