--------------------------------------------------------------------------------

--

-- FileName: ps2\_keyboard.vhd

-- Dependencies: debounce.vhd

-- Design Software: Quartus II 32-bit Version 12.1 Build 177 SJ Full Version

--

-- HDL CODE IS PROVIDED "AS IS." DIGI-KEY EXPRESSLY DISCLAIMS ANY

-- WARRANTY OF ANY KIND, WHETHER EXPRESS OR IMPLIED, INCLUDING BUT NOT

-- LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A

-- PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL DIGI-KEY

-- BE LIABLE FOR ANY INCIDENTAL, SPECIAL, INDIRECT OR CONSEQUENTIAL

-- DAMAGES, LOST PROFITS OR LOST DATA, HARM TO YOUR EQUIPMENT, COST OF

-- PROCUREMENT OF SUBSTITUTE GOODS, TECHNOLOGY OR SERVICES, ANY CLAIMS

-- BY THIRD PARTIES (INCLUDING BUT NOT LIMITED TO ANY DEFENSE THEREOF),

-- ANY CLAIMS FOR INDEMNITY OR CONTRIBUTION, OR OTHER SIMILAR COSTS.

--

-- Version History

-- Version 1.0 11/25/2013 Scott Larson

-- Initial Public Release

--

--------------------------------------------------------------------------------

LIBRARY ieee;

USE ieee.std\_logic\_1164.all;

ENTITY ps2\_keyboard IS

GENERIC(

clk\_freq : INTEGER := 50\_000\_000; --system clock frequency in Hz

debounce\_counter\_size : INTEGER := 8); --set such that (2^size)/clk\_freq = 5us (size = 8 for 50MHz)

PORT(

clk : IN STD\_LOGIC; --system clock

ps2\_clk : IN STD\_LOGIC; --clock signal from PS/2 keyboard

ps2\_data : IN STD\_LOGIC; --data signal from PS/2 keyboard

ps2\_code\_new : OUT STD\_LOGIC; --flag that new PS/2 code is available on ps2\_code bus

ps2\_code : OUT STD\_LOGIC\_VECTOR(7 DOWNTO 0)); --code received from PS/2

END ps2\_keyboard;

ARCHITECTURE logic OF ps2\_keyboard IS

SIGNAL sync\_ffs : STD\_LOGIC\_VECTOR(1 DOWNTO 0); --synchronizer flip-flops for PS/2 signals

SIGNAL ps2\_clk\_int : STD\_LOGIC; --debounced clock signal from PS/2 keyboard

SIGNAL ps2\_data\_int : STD\_LOGIC; --debounced data signal from PS/2 keyboard

SIGNAL ps2\_word : STD\_LOGIC\_VECTOR(10 DOWNTO 0); --stores the ps2 data word

SIGNAL error : STD\_LOGIC; --validate parity, start, and stop bits

SIGNAL count\_idle : INTEGER RANGE 0 TO clk\_freq/18\_000; --counter to determine PS/2 is idle

--declare debounce component for debouncing PS2 input signals

COMPONENT debounce IS

GENERIC(

counter\_size : INTEGER); --debounce period (in seconds) = 2^counter\_size/(clk freq in Hz)

PORT(

clk : IN STD\_LOGIC; --input clock

button : IN STD\_LOGIC; --input signal to be debounced

result : OUT STD\_LOGIC); --debounced signal

END COMPONENT;

BEGIN

--synchronizer flip-flops

PROCESS(clk)

BEGIN

IF(clk'EVENT AND clk = '1') THEN --rising edge of system clock

sync\_ffs(0) <= ps2\_clk; --synchronize PS/2 clock signal

sync\_ffs(1) <= ps2\_data; --synchronize PS/2 data signal

END IF;

END PROCESS;

--debounce PS2 input signals

debounce\_ps2\_clk: debounce

GENERIC MAP(counter\_size => debounce\_counter\_size)

PORT MAP(clk => clk, button => sync\_ffs(0), result => ps2\_clk\_int);

debounce\_ps2\_data: debounce

GENERIC MAP(counter\_size => debounce\_counter\_size)

PORT MAP(clk => clk, button => sync\_ffs(1), result => ps2\_data\_int);

--input PS2 data

PROCESS(ps2\_clk\_int)

BEGIN

IF(ps2\_clk\_int'EVENT AND ps2\_clk\_int = '0') THEN --falling edge of PS2 clock

ps2\_word <= ps2\_data\_int & ps2\_word(10 DOWNTO 1); --shift in PS2 data bit

END IF;

END PROCESS;

--verify that parity, start, and stop bits are all correct

error <= NOT (NOT ps2\_word(0) AND ps2\_word(10) AND (ps2\_word(9) XOR ps2\_word(8) XOR

ps2\_word(7) XOR ps2\_word(6) XOR ps2\_word(5) XOR ps2\_word(4) XOR ps2\_word(3) XOR

ps2\_word(2) XOR ps2\_word(1)));

--determine if PS2 port is idle (i.e. last transaction is finished) and output result

PROCESS(clk)

BEGIN

IF(clk'EVENT AND clk = '1') THEN --rising edge of system clock

IF(ps2\_clk\_int = '0') THEN --low PS2 clock, PS/2 is active

count\_idle <= 0; --reset idle counter

ELSIF(count\_idle /= clk\_freq/18\_000) THEN --PS2 clock has been high less than a half clock period (<55us)

count\_idle <= count\_idle + 1; --continue counting

END IF;

IF(count\_idle = clk\_freq/18\_000 AND error = '0') THEN --idle threshold reached and no errors detected

ps2\_code\_new <= '1'; --set flag that new PS/2 code is available

ps2\_code <= ps2\_word(8 DOWNTO 1); --output new PS/2 code

ELSE --PS/2 port active or error detected

ps2\_code\_new <= '0'; --set flag that PS/2 transaction is in progress

END IF;

END IF;

END PROCESS;

END logic;