



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

-- Company: IAEA

-- Engineer: M.B.

--

-- Create Date: 10.12.2017 15:27:29

-- Design Name:

-- Module Name: bram\_incr - Behavioral

-- Project Name: MCA

-- Target Devices: Zynq

-- Tool Versions:

-- Description:

--

-- Dependencies:

--

-- Revision:

-- Revision 0.01 - File Created

-- Additional Comments:

--

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

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.NUMERIC\_STD.ALL;

use IEEE.std\_logic\_unsigned.all;

-- Uncomment the following library declaration if using

-- arithmetic functions with Signed or Unsigned values

--use IEEE.NUMERIC\_STD.ALL;

-- Uncomment the following library declaration if instantiating

-- any Xilinx leaf cells in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

entity bram\_incr is

Generic( BRAM\_ADDRESS\_SIZE: INTEGER := 11);

port

(

--bram address is byte address

bram\_addr : out STD\_LOGIC\_VECTOR ( BRAM\_ADDRESS\_SIZE - 1 downto 0 );

bram\_dout : in STD\_LOGIC\_VECTOR ( 31 downto 0 );

bram\_we : out STD\_LOGIC;

bram\_din : out STD\_LOGIC\_VECTOR ( 31 downto 0 );

peak\_amp : in STD\_LOGIC\_VECTOR ( 15 downto 0 );

peak\_amp\_rdy : in STD\_LOGIC;

rstn : STD\_LOGIC;

clk : STD\_LOGIC

);

end bram\_incr;

architecture Behavioral of bram\_incr is

signal doutb : std\_logic\_vector(31 downto 0) := (others => '0');

signal wea : std\_logic := '0';

signal uData, uSum : unsigned(31 downto 0); -- := (others => '0');

signal Data, Sum : std\_logic\_vector(31 downto 0);

--state machine

type state\_type is (st1\_idle, st2\_read, st3\_increment, st4\_store);

signal state, next\_state : state\_type;

signal web :std\_logic := '0';

signal inc :std\_logic := '0';

begin

--bram address is byte address; lowest two bits must be zero

bram\_addr <= peak\_amp(14 downto 15 - BRAM\_ADDRESS\_SIZE);

Data <= bram\_dout;

bram\_we <= web;

bram\_din <= Sum;

--

-- MOORE State-Machine for 'Add 1' cycle to write on port B

--

SYNC\_PROC: process ( clk)

begin

if ( clk'event and clk = '1') then

if (rstn = '0') then

state <= st1\_idle;

else

state <= next\_state;

end if;

end if;

end process;

OUTPUT\_DECODE: process (state)

begin

--decode internal output signals

if state = st1\_idle then

web <= '0';

inc <= '0';

elsif state = st2\_read then

web <= '0';

inc <= '0';

elsif state = st3\_increment then

web <= '0';

inc <= '1';

elsif state = st4\_store then

web <= '1';

inc <= '0';

else

web <= '0';

inc <= '0';

end if;

end process;

NEXT\_STATE\_DECODE: process (state, peak\_amp\_rdy)

begin

--declare default state for next\_state to avoid latches

next\_state <= state; --default is to stay in current state

--insert statements to decode next\_state

--below is a simple example

case (state) is

when st1\_idle =>

if peak\_amp\_rdy = '1' then

next\_state <= st2\_read;

else

next\_state <= st1\_idle;

end if;

when st2\_read =>

next\_state <= st3\_increment;

when st3\_increment =>

next\_state <= st4\_store;

when st4\_store =>

next\_state <= st1\_idle;

when others =>

next\_state <= st1\_idle;

end case;

end process;

--

--increment and latch

--

regadder: process(inc,uData)

begin

if inc = '1' then

uSum <= uData + 1;

end if;

end process;

Sum <= std\_logic\_vector(uSum);

uData <= unsigned(Data);

end Behavioral;