```
----------------------------------------------------------------------------------
-- Engineer: Taggart Bonham and Will Chisholm
--
-- Create Date: 05/23/2017 09:05:13 PM
-- Design Name:
-- Module Name: KnobDecoder- Behavioral
-- Project Name: Etch-a-Sketch final project
-- Target Devices: Digilent Basys3 Board
-- Tool Versions: Vivado 2016.1
-- Description: Translates debounced A,B signals from a rotary encoder
--and puts out CW/CCW control signals based on the rotation. This state machine
--is the underlying logic behind our controller.

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

library IEEE;
use IEEE.STD_LOGIC_1164.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 knobdecoder is
    Port ( a : in STD_LOGIC;  --knob signal a
           b : in STD_LOGIC;  --knob signal b
           cw : out STD_LOGIC;  --1 if knob is turning clockwise, 0 else
           ccw : out STD_LOGIC;  --1 if knob is turning counter-clockwise, 0 else
           clk : in STD_LOGIC);
end knobdecoder;

architecture Behavioral of knobdecoder is
type statetype is (alpha , beta, gamma, delta); --four states correspond to 4 possible
combinations of AB
--alpha = "00", beta = "10", gamma = "11", delta = "01"
signal currentstate, nextstate : statetype;

begin
process(clk, currentstate, a , b)
begin
    cw <= '0'; --default both directions to '0'
    ccw <= '0';
    nextstate <= currentstate; --default nextstate back to curstate

    if (rising_edge(clk)) then
      currentstate <= nextstate; --uepdate currentstate
    end if;
    case currentstate is
        when alpha => --"00"
            if (a = '1') then
                cw <= '1';
                nextstate <= beta;
            elsif (b = '1') then
                ccw <= '1';
                nextstate <= delta;
            end if;
        when beta => --"10"
            if (b = '1') then
                cw <= '1';
                nextstate <= gamma;
```

```
            elsif (a = '0') then
                ccw <= '1';
                nextstate <= alpha;
            end if;
        when gamma => --"11"
            if (a = '0') then
                cw <= '1';
                nextstate <= delta;
            elsif (b = '0') then
                ccw <= '1';
                nextstate <= beta;
            end if;
        when delta => --"01"
            if (b = '0') then
                cw <= '1';
                nextstate <= alpha;
            elsif (a = '1') then
                ccw <= '1';
                nextstate <= gamma;
            end if;
        end case;
end process;

end Behavioral;
```