```vhdl
library ieee;
USE ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity keys is
port (  key3:                   in std_logic;
        key2:                   in std_logic;
        rst:                    in std_logic;
        clk:                    in std_logic;
        cleanKey3: out std_logic;
        cleanKey2: out std_logic;
        roll_1:                 out std_logic_vector(2 downto 0);
        roll_2:                 out std_logic_vector(2 downto 0)
);
end keys;

-- when keys pushed they go to zero
architecture rtl of keys is -- call output logic
signal count, count_d: std_logic_vector(7 downto 0);
signal curr_in_roll1, curr_in_roll2, next_in_roll1, next_in_roll2,
        curr_out_roll1, curr_out_roll2, next_out_roll1, next_out_roll2: std_logic_vect
or(2 downto 0);
signal clean_key2, clean_key3, pressed_key3, pressed_key2, temp_key3, temp_key2: std_l
ogic;
begin
        debouncer_key3: process(key3, clk)
        begin -- edge detector, falling edge and then 'not' that to be zero
                if(clk='1' and clk'event) then
                        if(temp_key3='1' and key3='0') then
                                clean_key3<='0'; -- falling edge
                        elsif(temp_key3='0' and key3='1') then
                                clean_key3<='1'; -- rising edge
                        end if;
                end if;
        end process debouncer_key3;

        debouncer_key2: process(key2, clk)
        begin -- edge detector, falling edge and then 'not' that to be zero
                if(clk='1' and clk'event) then
                        if(temp_key2='1' and key2='0') then
                                clean_key2<='0'; -- falling edge
                        elsif(temp_key2='0' and key2='1') then
                                clean_key2<='1';
                        end if;
                end if;
        end process debouncer_key2;

        rst_roll: process(rst,clk)
        begin
                if rst='0' then
                        curr_in_roll1 <= "000";
                        curr_in_roll2 <= "000";
                        curr_out_roll1 <= "000";
                        curr_out_roll2 <= "000";
                        pressed_key3 <= '1';
                        pressed_key2 <= '1';
                        temp_key3 <= '1';
                        temp_key2 <= '1';
                elsif(clk='1' and clk'event) then
                        curr_in_roll1 <= next_in_roll1; -- update from temp next_in
                        curr_in_roll2 <= next_in_roll2;
                        curr_out_roll1 <= next_out_roll1;
                        curr_out_roll2 <= next_out_roll2;
                        pressed_key3 <= clean_key3;
                        pressed_key2 <= clean_key2;
                        temp_key3 <= key3; -- saves key state to compare with
                        temp_key2 <= key2;
                end if;
        end process rst_roll;

        rolling1: process(clk,pressed_key3, curr_in_roll1)
        begin
                if(pressed_key3 = '0') then
                        if(unsigned(curr_in_roll1) > 5) then
                                next_in_roll1 <= "001"; -- resets rolls if >5
                        else
                                next_in_roll1 <= std_logic_vector(unsigned(curr_in_rol
l1) + 1);
                        end if;
                else -- '1'
                        next_in_roll1 <= curr_in_roll1;
                end if;
        end process rolling1;

        rolling2:process(clk,pressed_key2, curr_in_roll2)
        begin
                if(pressed_key2 = '0') then
                        if(unsigned(curr_in_roll2) > 5) then
                                next_in_roll2 <= "001";
                        else
                                next_in_roll2 <= std_logic_vector(unsigned(curr_in_rol
l2) + 1);
                        end if;
                else -- '1'
                        next_in_roll2 <= curr_in_roll2;
                end if;
        end process rolling2;

        update_out_roll1: process(pressed_key3, curr_out_roll1, curr_in_roll1)
        begin
                if(pressed_key3 = '0') then
                        next_out_roll1 <= curr_out_roll1; -- assign to temp next_out
                else
                        next_out_roll1 <= curr_in_roll1;
                end if;
        end process update_out_roll1;


        update_out_roll2: process(pressed_key2, curr_out_roll2, curr_in_roll2)
        begin
                if(pressed_key2 = '0') then
                        next_out_roll2 <= curr_out_roll2;
                else
                        next_out_roll2 <= curr_in_roll2;
                end if;
        end process update_out_roll2;

        roll_1 <= curr_out_roll1; -- dummy assignment to actual roll1
        roll_2 <= curr_out_roll2;
        cleanKey3 <= pressed_key3; -- dummy assignment to actual cleanKey3
        cleanKey2 <= pressed_key2;
end rtl;
```