```
--------------------------------------------------------------------------------
------
-- State machine for multisegment envelope generator
--------------------------------------------------------------------------------
------

library ieee;
use ieee.std_logic_1164.all;

package StateType_pkg is
type StateType is (idle,attack,sustain,release);
end StateType_pkg;

library ieee;
use ieee.std_logic_1164.all;
use work.StateType_pkg.all;

package statemachine_pkg is
        component statemachine
        port (
                clk:in std_logic;
                reset: in std_logic;
                gate:in std_logic;
                settled:in std_logic;
                envend:in std_logic;
                hold:in std_logic;
                inc:out std_logic;
                clear:out std_logic;
                state:buffer StateType
                );
        end component;
end statemachine_pkg;

library ieee;
use ieee.std_logic_1164.all;

library cypress;
use cypress.std_arith.all;
use cypress.lpmpkg.all;
use work.statemachine_pkg.all;
use work.StateType_pkg.all;

entity statemachine is
        port (
                clk:in std_logic;
                reset: in std_logic;
                gate:in std_logic;
                settled:in std_logic;
                envend:in std_logic;
                hold:in std_logic;
                inc:out std_logic;
                clear:out std_logic;
                state:buffer StateType
                );
end;

architecture statemachine_arch of statemachine is
signal current_state,next_state: StateType;
begin
        state_clock:process(reset,clk)
        begin
                if(reset = '0') then
                        current_state <= idle;
```

```
        elsif (clk'event and clk = '1') then
                current_state <= next_state;
        end if;
    end process state_clock;
    state_comb:process(current_state,gate,settled,envend,hold)
    begin
        case current_state is
            when idle =>
                if(gate = '1') then
                    next_state <= attack;
                    inc <= '1';
                    clear <= '0';
                else
                    next_state <= idle;
                    inc <= '0';
                    clear <= '0';
                end if;
            when attack =>
                if(gate = '0') then
                    next_state <= idle;
                    inc <= '0';
                    clear <= '1';
                elsif (gate = '1') and (settled = '0') and (hold =
'0') then
                    next_state <= attack;
                    inc <= '0';
                    clear <= '0';
                elsif (gate = '1') and (settled = '1') and (hold =
'0') then
                    next_state <= attack;
                    inc <= '1';
                    clear <= '0';
                elsif (gate = '1') and (hold = '1') then
                    next_state <= sustain;
                    inc <= '0';
                    clear <= '0';
                else
                    next_state <= attack;
                    inc <= '0';
                    clear <= '0';
                end if;
            when sustain =>
                if(gate = '1') then
                    next_state <= sustain;
                    inc <= '0';
                    clear <= '0';
                elsif (gate = '0') and (envend = '1') then
                    next_state <= idle;
                    inc <= '0';
                    clear <= '1';
                elsif (gate = '0') and (envend = '0') then
                    next_state <= release;
                    inc <= '1';
                    clear <= '0';
                else
                    next_state <= sustain;
                    inc <= '0';
                    clear <= '0';
                end if;
            when release =>
                if(settled = '0') then
                    next_state <= release;
                    inc <= '0';
```

```
                clear <= '0';
            elsif(settled = '1') and (envend = '0') then
                next_state <= release;
                inc <= '1';
                clear <= '0';
            elsif(settled = '1') and (envend = '1') then
                next_state <= idle;
                inc <= '0';
                clear <= '1';
            else
                next_state <= idle;
                inc <= '0';
                clear <= '0';
            end if;
        end case;
    end process state_comb;
    state <= current_state;
end statemachine_arch;
```