library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_ARITH.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

use ieee.numeric\_std.all;

entity carParking is

port(

A : in std\_logic;

B : in std\_logic;

reset : in std\_logic;

clock : in std\_logic;

entranceA : out std\_logic;

exitB : out std\_logic;

count : out std\_logic\_vector(9 downto 0));

end carParking;

architecture Behavioral of carParking is

signal cou : std\_logic\_vector(9 downto 0);

begin

process(clock)

begin

if(rising\_edge(clock)) then

if(reset = '1') then

cou(0) <= '0';

cou(1) <= '0';

cou(2) <= '0';

cou(3) <= '0';

cou(4) <= '0';

cou(5) <= '0';

cou(6) <= '0';

cou(7) <= '0';

cou(8) <= '0';

cou(9) <= '0';

entranceA <= '0';

exitB <= '0';

end if;

if(A = '1' and B = '0' and cou < "1111111111" and reset = '0') then

cou <= cou+1;

entranceA <= '1';

exitB <= '0';

end if;

if(A = '0' and B = '1' and cou > "000000000" and reset = '0') then

cou <= cou-1;

entranceA <= '0';

exitB <= '1';

end if;

end if;

end process;

process(clock)

begin

if(rising\_edge(clock)) then

count<= cou;

end if;

end process;

end Behavioral;