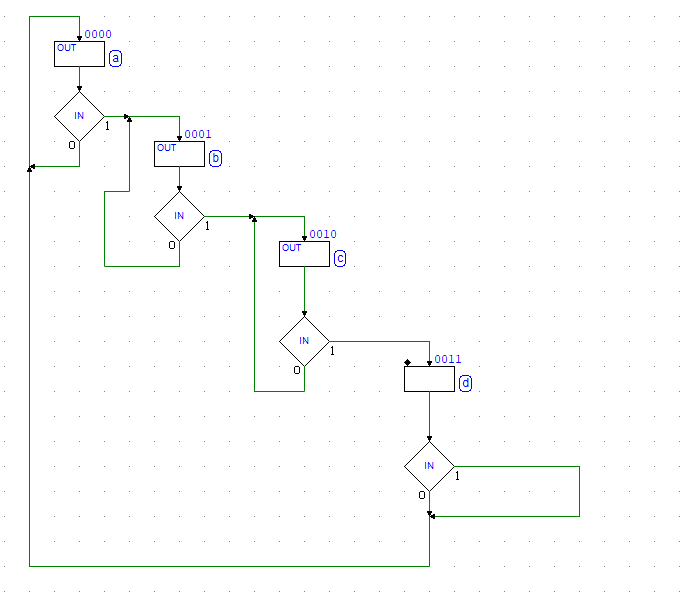
FSM



LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

ENTITY AAA IS

PORT( ----------------------------------->Clock & Reset:

Ck: IN std\_logic;

Reset: IN std\_logic;

----------------------------------->Inputs:

i\_IN: IN std\_logic;

----------------------------------->Outputs:

o\_OUT: OUT std\_logic

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

);

END AAA;

ARCHITECTURE behavioral OF AAA IS -- (Behavioral Description)

TYPE states is ( state\_a,

state\_b,

state\_c,

state\_d,

dummy\_0100,

dummy\_0101,

dummy\_0110,

dummy\_0111,

dummy\_1000,

dummy\_1001,

dummy\_1010,

dummy\_1011,

dummy\_1100,

dummy\_1101,

dummy\_1110,

dummy\_1111 );

SIGNAL State,

Next\_State: states;

BEGIN

-- Next State Combinational Logic ----------------------------------

FSM: process( State, i\_IN )

begin

CASE State IS

when state\_a =>

if (i\_IN = '1') then

Next\_State <= state\_b;

else

Next\_State <= state\_a;

end if;

when state\_b =>

if (i\_IN = '1') then

Next\_State <= state\_c;

else

Next\_State <= state\_b;

end if;

when state\_c =>

if (i\_IN = '1') then

Next\_State <= state\_d;

else

Next\_State <= state\_c;

end if;

when state\_d =>

Next\_State <= state\_a;

when OTHERS =>

Next\_State <= state\_d;

END case;

end process;

-- State Register --------------------------------------------------

REG: process( Ck, Reset )

begin

if (Reset = '0') then

State <= state\_d;

elsif rising\_edge(Ck) then

State <= Next\_State;

end if;

end process;

-- Outputs Combinational Logic -----------------------------------

OUTPUTS: process( State, i\_IN )

begin

-- Set output defaults:

o\_OUT <= '0';

-- Set output as function of current state and input:

CASE State IS

when state\_a =>

o\_OUT <= '1';

when state\_b =>

o\_OUT <= '1';

when state\_c =>

o\_OUT <= '1';

when OTHERS =>

o\_OUT <= '0';

END case;

end process;

END behavioral;