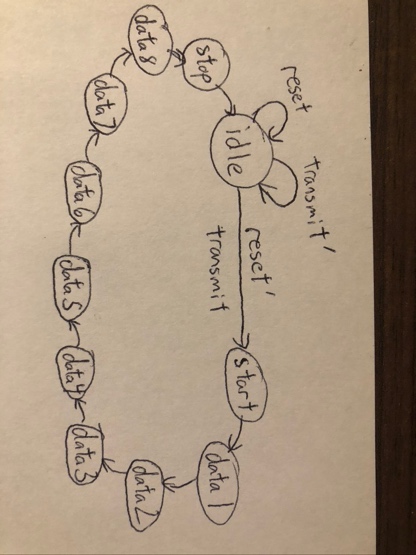
**FSM**



**ASM**

A picture containing text, whiteboard

Description automatically generated

**Manch\_encoder**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.NUMERIC\_STD.ALL;

entity manch\_encoder is

Port ( str\_input : in STD\_LOGIC\_VECTOR (7 downto 0);

clk : in STD\_LOGIC;

reset : in STD\_LOGIC;

transmit : in STD\_LOGIC;

str\_output : out STD\_LOGIC);

end manch\_encoder;

architecture Behavioral of manch\_encoder is

type statetype is (idle, start, data1, data2, data3, data4, data5, data6, data7, data8, stop);

signal state\_reg, state\_next : statetype;

signal str\_output\_next, str\_output\_reg : std\_logic := '0';

signal data\_reg : std\_logic\_vector(7 downto 0);

begin

--state\_reg

process(clk, reset, transmit) is

begin

if(transmit = '1') then

data\_reg <= str\_input;

end if;

if(reset = '1') then

state\_reg <= idle;

data\_reg <= "00000000";

elsif(clk'event and clk = '1') then

state\_reg <= state\_next;

end if;

end process;

--Next state logic

process(state\_reg, transmit)

begin

case(state\_reg) is

when idle =>

if(transmit = '1') then

state\_next <= start;

else

state\_next <= idle;

end if;

when start => state\_next <= data1;

when data1 => state\_next <= data2;

when data2 => state\_next <= data3;

when data3 => state\_next <= data4;

when data4 => state\_next <= data5;

when data5 => state\_next <= data6;

when data6 => state\_next <= data7;

when data7 => state\_next <= data8;

when data8 => state\_next <= stop;

when stop => state\_next <= idle;

end case;

end process;

process(state\_next)

begin

str\_output\_next <= '0';

case(state\_next) is

when idle => str\_output\_next <= '0';

when start => str\_output\_next <= '0';

when data1 => str\_output\_next <= data\_reg(7);

when data2 => str\_output\_next <= data\_reg(6);

when data3 => str\_output\_next <= data\_reg(5);

when data4 => str\_output\_next <= data\_reg(4);

when data5 => str\_output\_next <= data\_reg(3);

when data6 => str\_output\_next <= data\_reg(2);

when data7 => str\_output\_next <= data\_reg(1);

when data8 => str\_output\_next <= data\_reg(0);

when stop => str\_output\_next <= '0';

end case;

end process;

process(clk, reset, str\_output\_next)

begin

if(reset = '1') then

str\_output\_reg <= '0';

elsif (clk'event and clk = '1') then

str\_output\_reg <= str\_output\_next;

end if;

end process;

process(clk, state\_reg)

begin

if(state\_reg = idle or state\_reg = start or state\_reg = stop) then

str\_output <= '0';

else

str\_output <= str\_output\_reg xnor clk;

end if;

end process;

end Behavioral;

Manch\_encoder\_tb

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity manch\_encoder\_tb is

end manch\_encoder\_tb;

architecture Behavioral of manch\_encoder\_tb is

signal d\_in : std\_logic\_vector(7 downto 0);

signal clk, reset, d\_out : std\_logic;

signal t : std\_logic := '1';

begin

p0 : entity work.manch\_encoder(behavioral) port map(d\_in, clk, reset, t, d\_out);

d\_in <= "10011011";

--t <= '1';

process is

begin

reset <= '0';

clk <= '1';

wait for 50ns;

clk <= '0';

t <= '0';

wait for 50ns;

end process;

end Behavioral;

A screenshot of a computer

Description automatically generated with medium confidence