**ĐỒ ÁN MÔN HỌC**

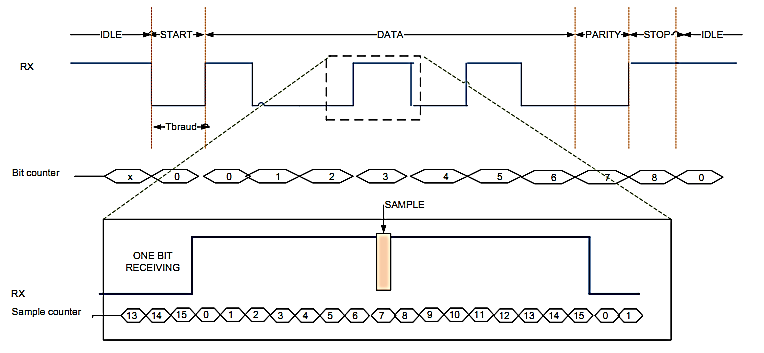
THIẾT KẾ LOGIC SỐ

**Đề bài**: Thiết kế khối truyền nhận giao thức UART. Các kí tự nhận được sẽ hiển thị lên khối LCD 16x2, đồng thời gửi lại máy tính các ký tự đã nhận được.

1. **Cơ sở lý thuyết**

UART là khối thực hiện giao thức truyền tin dị bộ nối tiếp(Universal Asynchronous Receiver/Transmitter), đặc điểm giao thức này là đơn giản, phổ biến, nhược điểm là tốc độ trao đổi thông tin hạn chế.

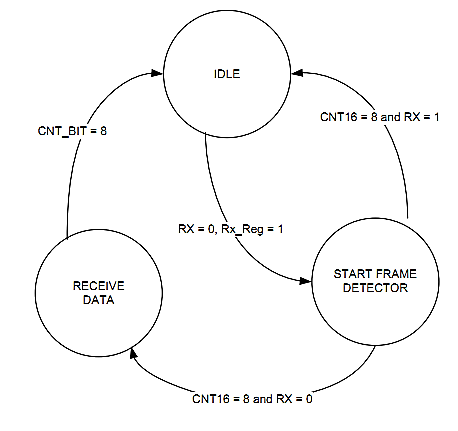
Thông tin nối tiếp được truyền theo một dây dẫn duy nhất và các bit thông tin được mã hóa theo mức điện áp trên dây dẫn.



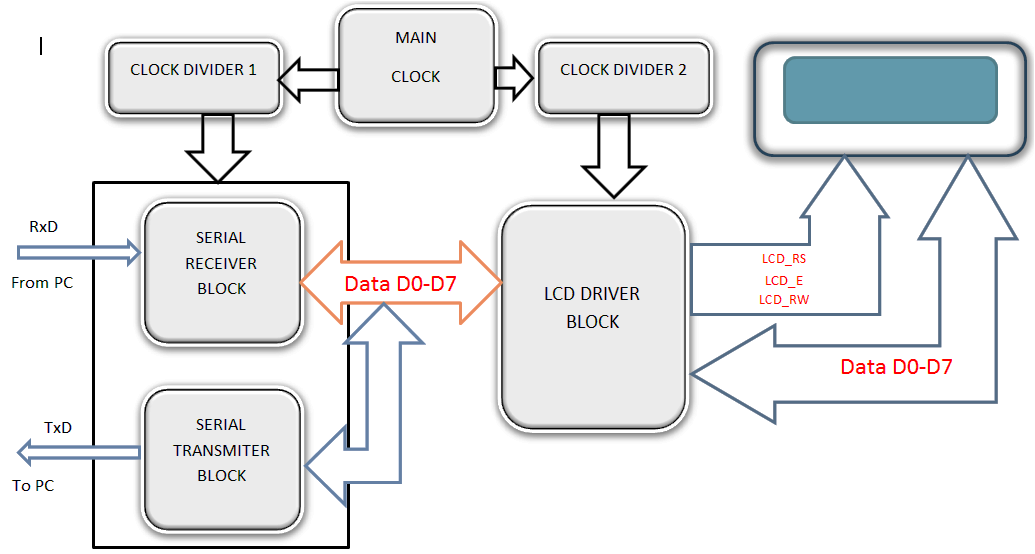
Hình trên mô tả tín hiệu đầu vào cho một khối nhận tín hiệu từ đường truyền nối tiếp, ở trạng thái nghỉ (IDLE), không có dữ liệu thì tín hiệu được giữ ở mức cao, để bắt đầu truyền thông tin tín hiệu Rx sẽ chuyển về mức thấp trong một khoảng thời gian đủ lớn, thời gian này bằng thời gian nhận 1 bit thông tin tương ứng với tốc độ truyền của cổng gọi là Tbraud. = 1/Fbraud. Sau bit START này thì các bit dữ liệu được truyền nối tiếp trên Rx, tương ứng trạng thái DATA trên hình vẽ, cổng COM có thể được cấu hình để truyền nhận 6, 7 hay 8 bit thông tin. Sau khi kết thúc truyền các tin này có thể có thêm một bit kiểm tra chẵn lẻ của khối tin PARITY, một bit STOP (mức logic 1) được giữ với thời gian bằng 1, 1.5 hay 2 Tbraud. Khi kết thúc quá trình truyền tin thì Rx trở về trạng thái nghỉ ở mức điện áp cao. Đường truyền nối tiếp đơn giản và tốc độ không cao, thông thường Fbraud thay đổi từ 1200 đến 115200 Bit/s. Đường truyền này sử dụng các thiết bị độc lập với nhau và bị ảnh hưởng của nhiễu không nhỏ, mặt khác bộ nhận và bộ chia hoạt động không đồng bộ (không cùng xung nhịp hệ thống) do đó cần phải có cơ chế thu nhận nhằm tránh lỗi phát sinh. Cơ chế đó cũng được minh họa ở trên hình 4.49. Một bit thông tin được chia thành 16 điểm lấy mẫu (Samples), các điểm này được xác định bằng một bộ đếm mẫu (sample counter). Vì xác suất lỗi ở các vị trí mẫu đầu và cuối là cao nhất còn xác suất lỗi ở vị trí giữa là thấp nhất do đó ta chọn điểm lấy mẫu ở giữa, nghĩa là bit thông tin đang nhận bằng Rx ở trạng thái counter = 8.

Với tốc độ cao nhất có thể là 115200 Bit/s thì tần số của bộ đếm lấy mẫu bằng 115200 x16 = 1 843 200 Hz < 2Mhz vẫn là một tốc độ không cao đối với các thiết kế số trên FPGA. Trên thực tế cũng không phải lúc nào cũng phải chia bit thông tin thành 16 mẫu mà có thể chia lớn hơn hoặc nhỏ hơn.

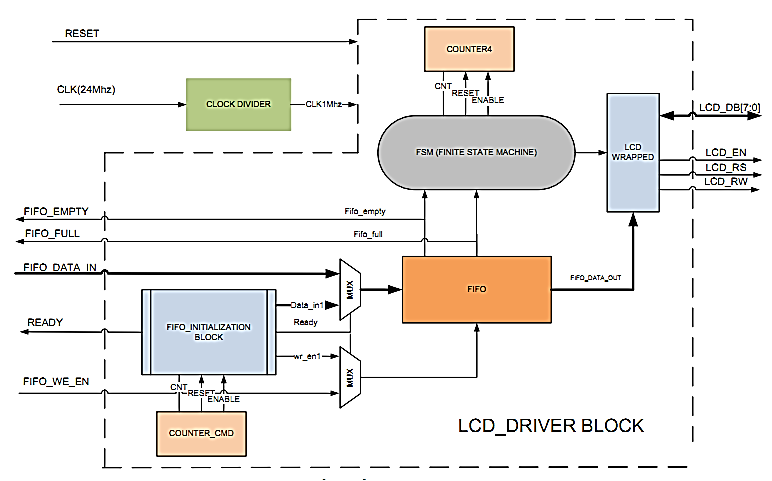
Hình dưới trình bày máy trạng thái hữu hạn của phần thu,máy trạng thái hữu hạn của phần phát cũng có dạng tương tự, chỉ khác là trạng thái receive data thay bằng transmit data.



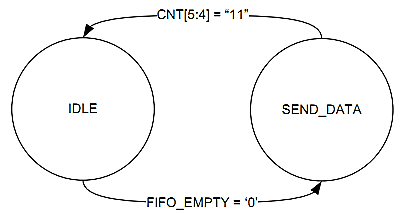
* So đồ khối của hệ thu phát dị bộ UART và hiển thị LCD cho như hình dưới



Sơ đồ khối của khối điều khiển LCD:



Máy trạng thái hữu hạn của khối:



1. **Mã VHDL cho các khối**
2. Khối chia tần :file nguồn clk\_div.vhd

***-----------clk\_div-------------------***

**library IEEE;**

**use IEEE.std\_logic\_1164.all;**

**use IEEE.std\_logic\_unsigned.all;**

**--------------------------------------**

**entity clk\_div is**

**generic(baud :std\_logic\_vector(11 downto 0):=x"058");**

**port ( clk :in std\_logic;**

**clk16 :out std\_logic**

**);**

**end entity;**

**---------------------------------------------------------**

**architecture behavioral of clk\_div is**

**signal cnt :std\_logic\_vector(11 downto 0):=x"000";**

**signal clk\_temp:std\_logic:='0';**

**BEGIN**

**process(clk,cnt)**

**begin**

**if rising\_edge(clk) then**

**if cnt = baud then**

**cnt <= x"000";**

**else**

**cnt <= cnt + 1;**

**end if;**

**end if;**

**end process;**

**process(clk,cnt,clk\_temp)**

**begin**

**if rising\_edge(clk) then**

**if cnt = baud then**

**clk\_temp <= not clk\_temp;**

**else**

**clk\_temp <= clk\_temp;**

**end if;**

**end if;**

**end process;**

**clk16<=clk\_temp;**

**END architecture;**

1. Khối UART:
2. Khối thu: file nguồn Serial\_Rx.vhd

**--**

**Library ieee;**

**use ieee.std\_logic\_1164.all;**

**------------------------------------------------**

**Entity rx\_uart is**

**port(**

**clk : in std\_logic;--xung clock**

**rst : in std\_logic;--reset input**

**rx\_en : in std\_logic;--enable input**

**rx : in std\_logic;**

**donerx : out std\_logic;-- bao nhan xong**

**dout : out std\_logic\_vector(7 downto 0));--byte nhan**

**end rx\_uart;**

**--------------------------------------------------**

**Architecture bhv of rx\_uart is**

**type state\_type is (idle,start\_rx,shift\_rx,stop\_rx);**

**signal rxfsm : state\_type;**

**signal rxbitcnt,div\_cnt : integer;**

**signal srx : std\_logic;**

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

**begin**

**rx\_fsm: process (rst,clk,rx\_en)**

**begin**

**if rst='1' then**

**rx\_reg <= (others => '0');**

**rxbitcnt <= 0;**

**div\_cnt <= 0;**

**rxfsm <= idle;**

**elsif rx\_en = '1' then**

**if rising\_edge(clk) then**

**srx <= rx;**

**case rxfsm is**

**when idle =>**

**donerx<='0';**

**div\_cnt <= 1;**

**if rx = '0' and srx = '1' then**

**rxfsm <= start\_rx;**

**end if;**

**when start\_rx =>**

**if div\_cnt = 8 then**

**if rx = '0' then**

**rxbitcnt <= 0;**

**div\_cnt <= 1;**

**rxfsm <= shift\_rx;**

**rx\_reg <= (others => '0');**

**else**

**rxfsm <= idle;**

**end if;**

**else**

**div\_cnt <= div\_cnt + 1;**

**end if;**

**when shift\_rx =>**

**if div\_cnt = 16 then**

**div\_cnt <= 1;**

**if rxbitcnt < 8 then**

**rx\_reg(rxbitcnt) <= rx;**

**elsif rxbitcnt = 8 then**

**rxfsm <= stop\_rx;**

**donerx<='1';**

**end if;**

**rxbitcnt <= rxbitcnt + 1;**

**else**

**div\_cnt <= div\_cnt + 1;**

**end if;**

**when stop\_rx =>**

**rxfsm <= idle;**

**when others =>**

**rxfsm <= idle;**

**end case;**

**end if;**

**end if;**

**end process rx\_fsm;**

**dout <= rx\_reg;**

**end bhv;**

1. Khối phát:

**Library ieee;**

**use ieee.std\_logic\_1164.all;**

**------------------------------------------------**

**Entity tx\_uart is**

**port(**

**clk : in std\_logic;--xung clock**

**rst : in std\_logic;--reset**

**tx\_en : in std\_logic;--cho phep**

**tx : out std\_logic;-- dau ra**

**donetx : out std\_logic;--truyen xong**

**din : in std\_logic\_vector(7 downto 0));--byte truyen**

**end tx\_uart;**

**--------------------------------------------------**

**Architecture bhv of tx\_uart is**

**type state\_type is (idle,load\_tx,shift\_tx,stop\_tx);**

**signal txfsm : state\_type;**

**signal regdin : std\_logic\_vector(7 downto 0);**

**signal txbitcnt : integer;**

**signal cnt\_div : integer;**

**signal tx\_reg : std\_logic\_vector(12 downto 0);**

**begin**

**tx <= tx\_reg(0);**

**tx\_fsm: process(rst,clk,tx\_en,din)**

**begin**

**if rst = '1' then**

**tx\_reg <= (others=>'1');**

**txbitcnt <= 0;**

**cnt\_div <= 0;**

**txfsm <= idle;**

**regdin <= (others=>'0');**

**donetx<='0';**

**elsif tx\_en = '1' then**

**if rising\_edge(clk) then**

**case txfsm is**

**when idle =>**

**donetx<='0';**

**regdin <= din;**

**txfsm <= load\_tx;**

**when load\_tx =>**

**txfsm <= shift\_tx;**

**txbitcnt <= 12;**

**cnt\_div <= 0;**

**tx\_reg <= "1111" & regdin & '0';**

**when shift\_tx =>**

**if cnt\_div = 15 then**

**txbitcnt <= txbitcnt - 1;**

**tx\_reg <= '1' & tx\_reg(tx\_reg'high downto 1);**

**cnt\_div <= 0;**

**if txbitcnt=1 then**

**tx\_reg <= '1' & tx\_reg(tx\_reg'high downto 1);**

**txfsm <= stop\_tx;**

**donetx<='1';**

**end if;**

**else**

**cnt\_div <= cnt\_div +1;**

**end if;**

**when stop\_tx =>**

**txfsm<=idle;**

**when others =>**

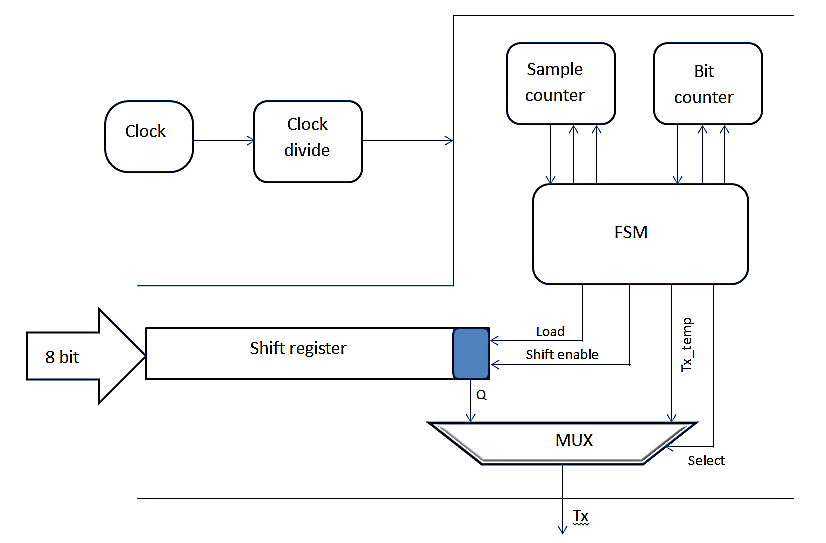
**txfsm <= idle;**

**end case;**

**end if;**

**end if;**

**end process tx\_fsm;**

**end bhv;** 

1. Khối tổng UART: theo yêu cầu đề bài ta thiết kế 1 khối UART có phần đầu ra song song của khối nhận sẽ nối trực tiếp với đầu vào song song của khối phát. Như vậy khi nhận được ký tự gì từ máy tính UART sẽ phát luôn trở lại PC.

**-------------------------------------------**

**Library ieee;**

**use ieee.std\_logic\_1164.all;**

**use ieee.std\_logic\_unsigned.all;**

**use IEEE.STD\_LOGIC\_ARITH.all;**

**-------------------------------------------------**

**Entity UART is**

**port (**

**reg : out std\_logic\_vector(7 downto 0);**

**Rx : in std\_logic;**

**Tx : out std\_logic;**

**clk : in std\_logic;**

**Reset : in std\_logic;**

**En : in std\_logic**

**);**

**end entity;**

**--------------------------------------------**

**Architecture atk of UART is**

**signal s8bit : std\_logic\_vector(7 downto 0);**

**signal clk16 : std\_logic;**

**signal nreset : std\_logic;**

**signal entx :std\_logic:='0';**

**signal donetx :std\_logic:='0';**

**signal donetx1 :std\_logic:='0';**

**signal donerx :std\_logic:='0';**

**signal cnt :integer:=0;**

**signal cnt1 :integer:=0;**

**--------------------------------------**

**component clk\_div is**

**generic(baud :std\_logic\_vector(7 downto 0):="01000001");**

**port ( clk :in std\_logic;**

**clk16 :out std\_logic**

**);**

**end component;**

**component Tx\_uart is**

**port(**

**clk : in std\_logic;--xung clock**

**rst : in std\_logic;--reset**

**tx\_en : in std\_logic;--chophep**

**tx : out std\_logic;-- dau ra**

**donetx : out std\_logic;-- truyen xong**

**din : in std\_logic\_vector(7 downto 0));--byte truyen**

**end component;**

**component Rx\_uart is**

**port(**

**clk : in std\_logic;--xung clock**

**rst : in std\_logic;--reset input**

**rx\_en : in std\_logic;--enable input**

**rx : in std\_logic;**

**donerx : out std\_logic;--nhan xong**

**dout : out std\_logic\_vector(7 downto 0));--byte nhan**

**end component;**

**begin**

**cTX\_uart : Tx\_uart port map(clk16,nreset,entx,tx,donetx,s8bit);**

**cRx\_uart : Rx\_uart port map(clk16,nreset,en,rx,donerx,s8bit);**

**U6 :component clk\_div**

**generic map(x"02")**

**port map(clk,clk16);**

**nreset<=(not reset) or donetx1;**

**c:process(clk16,donerx)**

**begin**

**if rising\_edge(clk16) then**

**if donerx='1' or cnt>0 then**

**cnt<=cnt+1;**

**if cnt<195 then**

**entx<='1';**

**else cnt<=0;**

**entx<='0';**

**end if;**

**end if;**

**end if;**

**end process;**

**d: process(clk16,donetx)**

**begin**

**if rising\_edge(clk16) then**

**if donetx='1' or cnt1>0 then**

**cnt1<=cnt1+1;**

**if cnt1<10 then**

**donetx1<='1';**

**else**

**donetx1<='0';**

**cnt1<=0;**

**end if;**

**end if;**

**end if;**

**end process;**

**q:process(reset,clk16,donerx)**

**begin**

**if reset='0' then**

**reg<=(others=>'0');**

**elsif rising\_edge(clk16) then**

**if donerx='1' then**

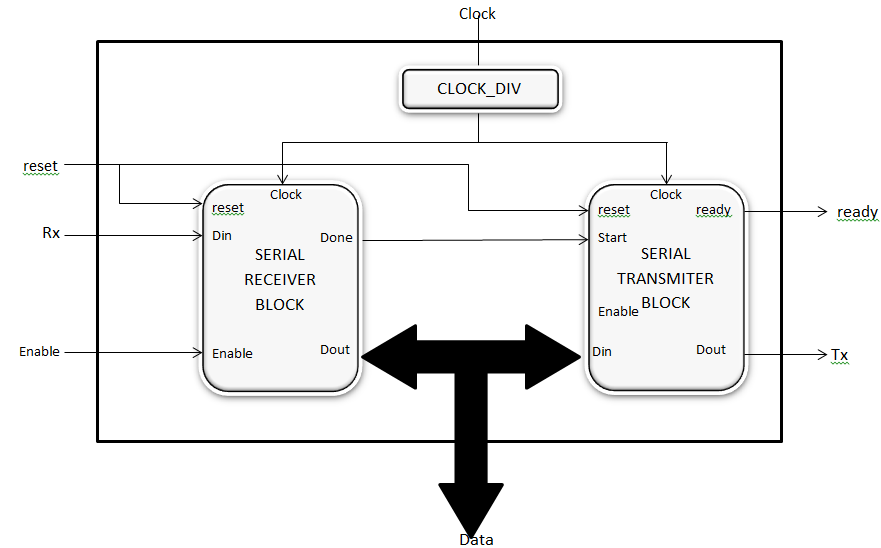
**reg <= s8bit;**

**end if;**

**end if;**

**end process;**

**end atk;**



Khối tổng:

**library IEEE;**

**use IEEE.STD\_LOGIC\_1164.ALL;**

**use IEEE.STD\_LOGIC\_ARITH.ALL;**

**use IEEE.STD\_LOGIC\_UNSIGNED.ALL;**

**library UNISIM;**

**use UNISIM.VComponents.all;**

**Library XilinxCoreLib;**

**Entity hethong is**

**port (**

**reg : out std\_logic\_vector(7 downto 0);**

**Rx : in std\_logic;**

**Tx : out std\_logic;**

**clk : in std\_logic;**

**Reset : in std\_logic;**

**En : in std\_logic;**

**wchar : in std\_logic;**

**lcd\_rs : out std\_logic;**

**lcd\_rw : out std\_logic;**

**lcd\_e : out std\_logic;**

**lcd\_data : inout std\_logic\_vector(7 downto 0)**

**);**

**end entity;**

**architecture at of hethong is**

**component lcd\_driver is**

**port (**

**clk1Mhz : in std\_logic;**

**reset : in std\_logic;**

**lcd\_rs : out std\_logic;**

**lcd\_rw : out std\_logic;**

**lcd\_e : out std\_logic;**

**data\_in : in std\_logic\_vector (9 downto 0); -- Data input**

**wr\_en : in std\_logic; -- Write Enable FIFO**

**full : out std\_logic; -- FIFO full**

**ready : out std\_logic; -- LCD ready**

**lcd\_data : out std\_logic\_vector (7 downto 0)**

**);**

**end component;**

**component clk\_div is**

**generic(baud :std\_logic\_vector(7 downto 0):="01000001");**

**port ( clk :in std\_logic;**

**clk16 :out std\_logic**

**);**

**end component;**

**component UART is**

**port (**

**reg : out std\_logic\_vector(7 downto 0);**

**Rx : in std\_logic;**

**Tx : out std\_logic;**

**clk : in std\_logic;**

**Reset : in std\_logic;**

**En : in std\_logic**

**);**

**end component ;**

**signal nreset : std\_logic;**

**signal clk1Mhz : std\_logic;**

**signal ready : std\_logic;**

**signal data\_in : std\_logic\_vector (9 downto 0); -- Data input**

**signal wr\_en : std\_logic; -- Write Enable**

**signal full : std\_logic; -- FIFO full**

**signal wchar\_reg : std\_logic;**

**signal wchar\_sig : std\_logic;**

**signal counter20 : std\_logic\_vector (19 downto 0);**

**signal second : std\_logic\_vector (3 downto 0);**

**signal lcd\_char : std\_logic\_vector (7 downto 0);**

**signal we\_sig : std\_logic;**

**signal we\_sig1 : std\_logic;**

**signal lcd\_char1 : std\_logic\_vector (7 downto 0);**

**signal reg1 : std\_logic\_vector (7 downto 0);**

**---------------------------------------**

**begin**

**u: component uart**

**port map (reg1,rx,tx,clk,reset,en);**

**cd : component clk\_div**

**generic map ("00011000") -- 48Mhz/24/2 = 1Mhz**

**port map (clk, clk1Mhz);**

**lcd\_x : component lcd\_driver**

**port map (clk1Mhz, nreset, lcd\_rs, lcd\_rw, lcd\_e, data\_in, wr\_en, full, ready, lcd\_data=> lcd\_data);**

**nreset <= not reset;**

**buttin:**

**process (clk1Mhz, en)**

**begin**

**if rising\_edge(clk1Mhz) then**

**wchar\_reg <= wchar;**

**counter20 <= counter20 + 1;**

**if (counter20 = x"fffff") then**

**we\_sig <= '1';**

**if second = x"9" then**

**second <= x"0";**

**else**

**second <= second + 1;**

**end if;**

**else**

**we\_sig <= '0';**

**second <= second;**

**end if;**

**we\_sig1 <= we\_sig;**

**-- tin hieu we\_sig1 tre so voi we 1 xung nhip de thuc hien 2 lenh tren LCD**

**end if;**

**end process buttin;**

**wr\_en <= (we\_sig1 or we\_sig or wchar\_sig) and ready and (not full);**

**wchar\_sig <= (not wchar) and wchar\_reg;**

**process (we\_sig1, we\_sig, lcd\_char)**

**begin**

**if we\_sig = '1' then**

**data\_in <= "00" & x"c0"; -- dua con tro ve dau dong 2**

**elsif we\_sig1 = '1' then**

**data\_in <= "10" & lcd\_char;**

**else**

**data\_in <= "01" & x"ff";**

**end if;**

**end process;**

**lcd\_char1<=reg1;**

**convert\_to\_lcd\_char: process (second,lcd\_char1)**

**begin**

**case (second) is**

**when x"0" => lcd\_char <= lcd\_char1;**

**when x"1" => lcd\_char <= lcd\_char1;**

**when x"2" => lcd\_char <= lcd\_char1;**

**when x"3" => lcd\_char <= lcd\_char1;**

**when x"4" => lcd\_char <= lcd\_char1;**

**when x"5" => lcd\_char <= lcd\_char1;**

**when x"6" => lcd\_char <= lcd\_char1;**

**when x"7" => lcd\_char <= lcd\_char1;**

**when x"8" => lcd\_char <= lcd\_char1;**

**when x"9" => lcd\_char <= lcd\_char1;**

**when others => lcd\_char <= lcd\_char1;**

**end case;**

**end process convert\_to\_lcd\_char;**

**reg<=lcd\_char1;**

**end at;**