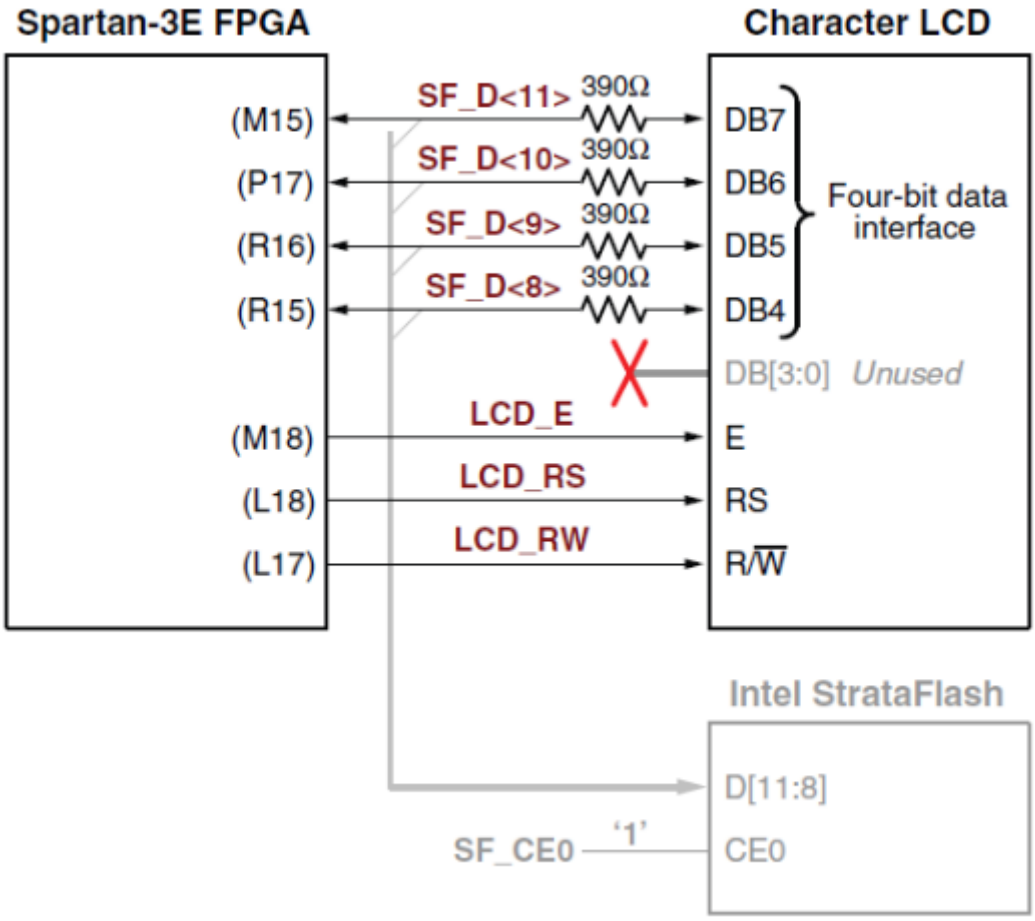
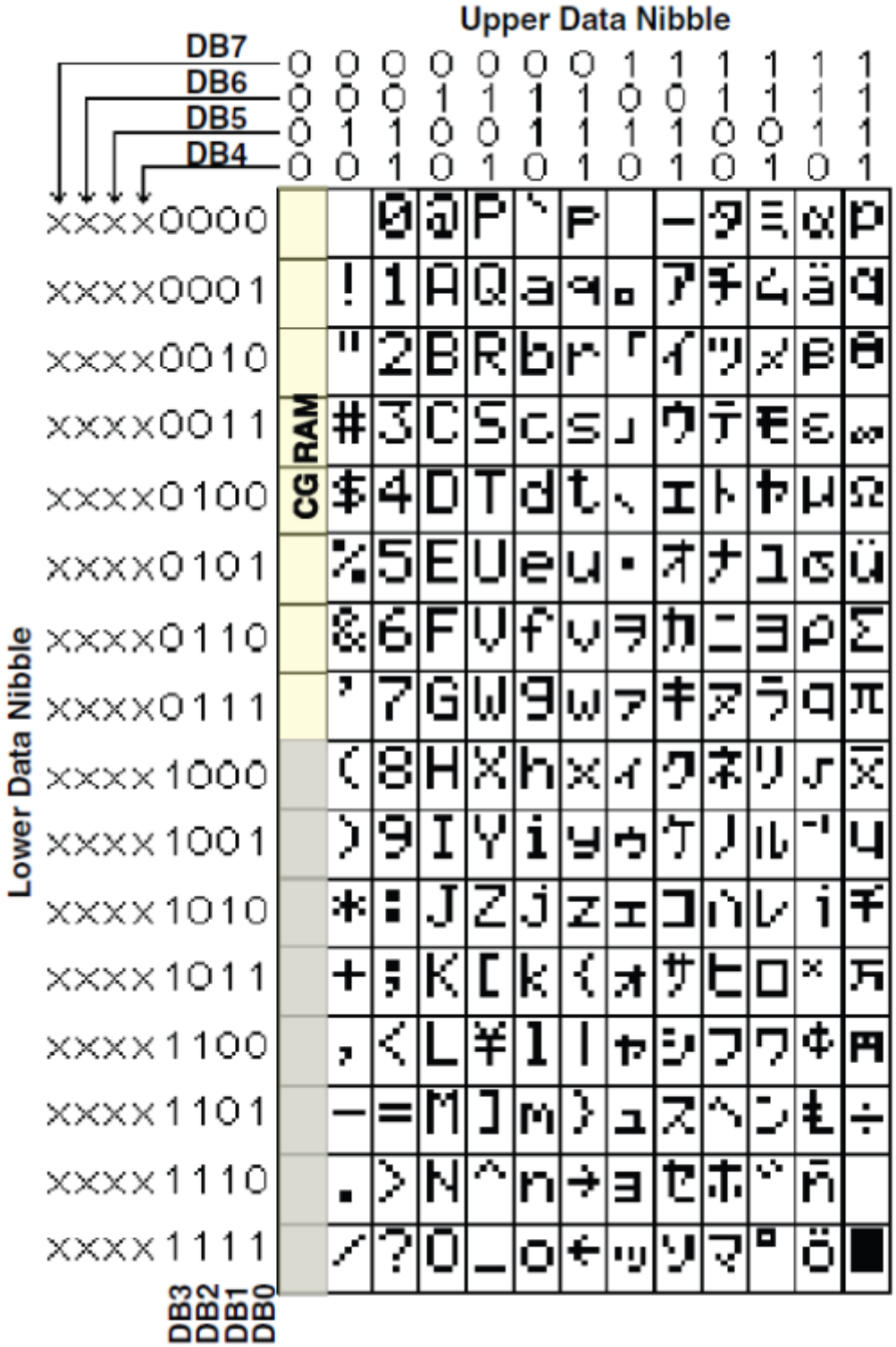
Mohamed Manessouri
SSE

LCD PROJECT

The Spartan-3E contains a Liquid Crystal Display (LCD) which displays 2 lines of 16 characters each. The FPGA controls the LCD through a 4-bit interface, as shown in Figure 4.1. The LCD actually supports an 8-bit interface, however the Spartan-3E uses a 4-bit interface to remain compatible with other Xilinx products, and to minimise pin count.




Upper Data Nibble
Lower Data Nibble
DB7
DB6
DB5
DB4
DB3
DB2
DB1
DB0
CG RAM
xxxx0000
xxxx0001
xxxx0010
xxxx0011
xxxx0100
xxxx0101
xxxx0110
xxxx0111
xxxx1000
xxxx1001
xxxx1010
xxxx1011
xxxx1100
xxxx1101
xxxx1110
xxxx1111

```
19  ----------------------------------------------------------
20  library IEEE;
21  use IEEE.STD_LOGIC_1164.ALL;
22  use IEEE.STD_LOGIC_1164.ALL;
23  use IEEE.STD_LOGIC_ARITH.ALL;
24  use IEEE.STD_LOGIC_UNSIGNED.ALL;
25
26  -- Uncomment the following library declaration if using
27  -- arithmetic functions with Signed or Unsigned values
28  --use IEEE.NUMERIC_STD.ALL;
29
30  -- Uncomment the following library declaration if insta
31  -- any Xilinx primitives in this code.
32  --library UNISIM;
33  --use UNISIM.VComponents.all;
34
35  entity LCD is
36  Port (
37          clk : in STD_LOGIC;  -- 50-MHz clock
38          sf_e : out STD_LOGIC; -- LCD access enable
39          e : out STD_LOGIC;   -- Enable
40          rs : out STD_LOGIC;  -- Register Select
41          rw : out STD_LOGIC;  -- Read/Write
42          db_4 : out STD_LOGIC;   -- Data bit 4
43          db_3 : out STD_LOGIC;   -- Data bit 3
44          db_2 : out STD_LOGIC;   -- Data bit 2
45          db_1 : out STD_LOGIC    -- Data bit 1
46      );
```

```
architecture Behavioral of LCD is
    signal count : STD_LOGIC_VECTOR(26 downto 0) := (others => '0');
    signal code : STD_LOGIC_VECTOR(5 downto 0) := (others => '0');
    signal refresh : STD_LOGIC := '0';

begin
process(clk)
    begin
       if rising_edge(clk) then
            -- Increment counter
            count <= count + 1;

            case count(26 downto 21) is
                when "000000" => code <= "000011"; -- Power-on init s
                when "000001" => code <= "000011"; -- Repeat power-on
                when "000010" => code <= "000011"; -- Ensure initial
                when "000011" => code <= "000010"; -- Transition to 4

                -- Function Set
                when "000100" => code <= "000010"; -- Upper nibble 00
                when "000101" => code <= "001000";

                when "000110" => code <= "000000"; -- Upper nibble 00
                when "000111" => code <= "000110"; -- Lower nibble 01

                -- Display On/Off Control
                when "001000" => code <= "000000"; -- Upper nibble 00
                when "001001" => code <= "001100";
```

```
    when "001010" => code <= "000000"; -
    when "001011" => code <= "000001";


    -- Write "MOHAMED"
    when "001100" => code <= "100100";
    when "001101" => code <= "101101";
    -- O
    when "001110" => code <= "100100";
    when "001111" => code <= "101111";
    -- H
    when "010000" => code <= "100100";
    when "010001" => code <= "101000";

   -- A
    when "010010" => code <= "100100";
    when "010011" => code <= "100001";
    -- M
    when "010100" => code <= "100100";
    when "010101" => code <= "101101";
    -- E
    when "010110" => code <= "100100";
    when "010111" => code <= "100101";


     -- D
    when "011000" => code <= "100100";
    when "011001" => code <= "100100";
```

```
-- Set Cursor to 2nd Line
when "011010" => code <= "001100";
when "011011" => code <= "000000";



 -- M
when "011100" => code <= "100100";
when "011101" => code <= "101101";
-- A
when "011110" => code <= "100100";
when "011111" => code <= "100001";
-- N
when "100000" => code <= "100100";
when "100001" => code <= "101110";

-- E
when "100010" => code <= "100100";
when "100011" => code <= "100101";
-- S
when "100100" => code <= "100101";
when "100101" => code <= "100011";
-- S
when "100110" => code <= "100101";
when "100111" => code <= "100011";
```

```
    when "101000" => code <= "100100";
    when "101001" => code <= "101111";
    -- U
    when "101010" => code <= "100101";
    when "101011" => code <= "100101";
    -- R
    when "101100" => code <= "100101";
    when "101101" => code <= "100010";


   -- I
    when "101110" => code <= "100100";
    when "101111" => code <= "101001";

    -- Default case: Idle state
    when others => code <= "010000"; -- Default idle
end case;
```

```
            sf_e <= '1'; -- Enable LCD access
            e <= refresh; -- Toggle enable signal
            rs <= code(5); -- RS is the MSB of `code`
            rw <= code(4); -- RW is the second MSB of `code`
            db_4 <= code(3); -- Data bit 4
            db_3 <= code(2); -- Data bit 3
            db_2 <= code(1); -- Data bit 2
            db_1 <= code(0); -- Data bit 1
        end if;
    end process;


end Behavioral;
```

```
1
2  NET "clk" loc = C9;
3  NET "db_1" loc = R15;
4  NET "db_2" loc = R16;
5  NET "db_3" loc = P17;
6  NET "db_4" loc = M15;
7
8  NET "e" loc = M18;
9  NET "rs" loc = L18;
10 NET "rw" loc = L17;
11 NET "sf_e" loc = D16;
12
13
```