The Abdus Salam International Centre for Theoretical Physics

2499-22

# International Training Workshop on FPGA Design for Scientific Instrumentation and Computing

*11 - 22 November 2013*

## Serial Communicaton with FPGA

Thulasiraman NANDHA KUMAR
*Department of Electrical & Electronic Engineering, Faculty of Engineering, Jalan Broga, 43500 Semenyih, Selangor Malaysia*

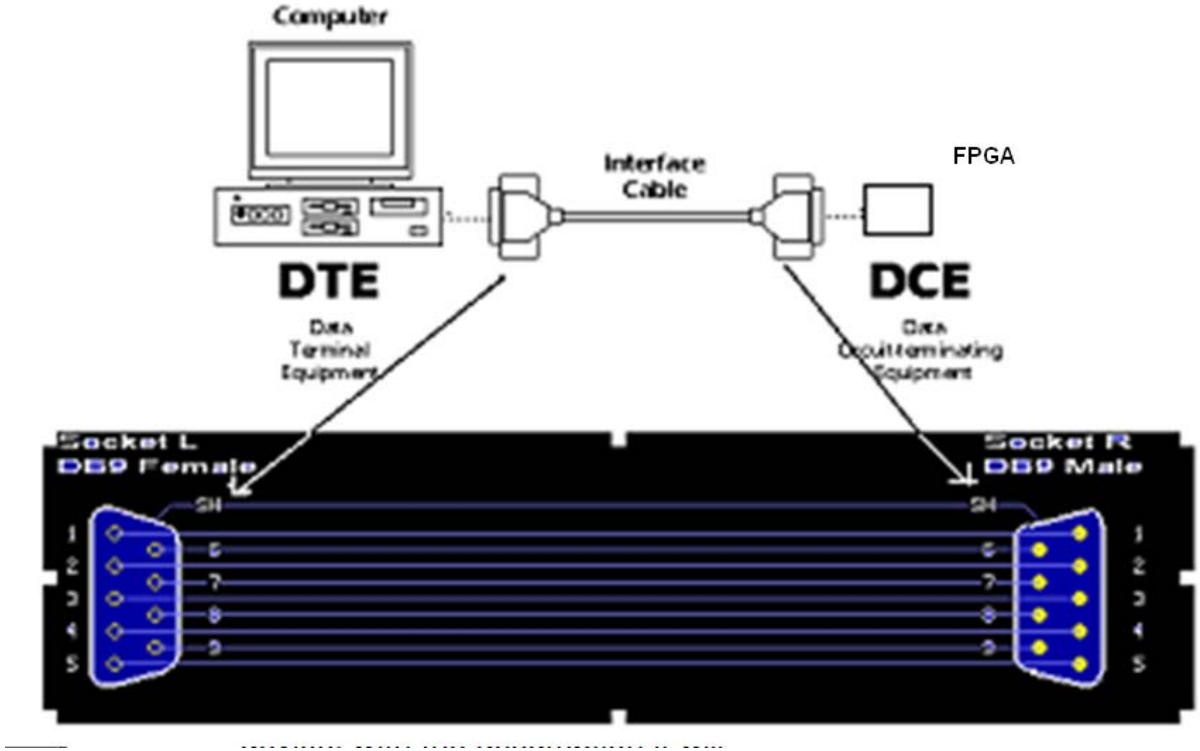
# Serial communication with FPGA

**Dr.T .Nandha Kumar**
**Senior Member IEEE, Fellow HEA(UK), C.Eng.(UK)**
**( nandhakumaar.t@nottingham.edu.my )**
**Associate Professor**
**The University of Nottingham**
**Malaysia**

# Objective

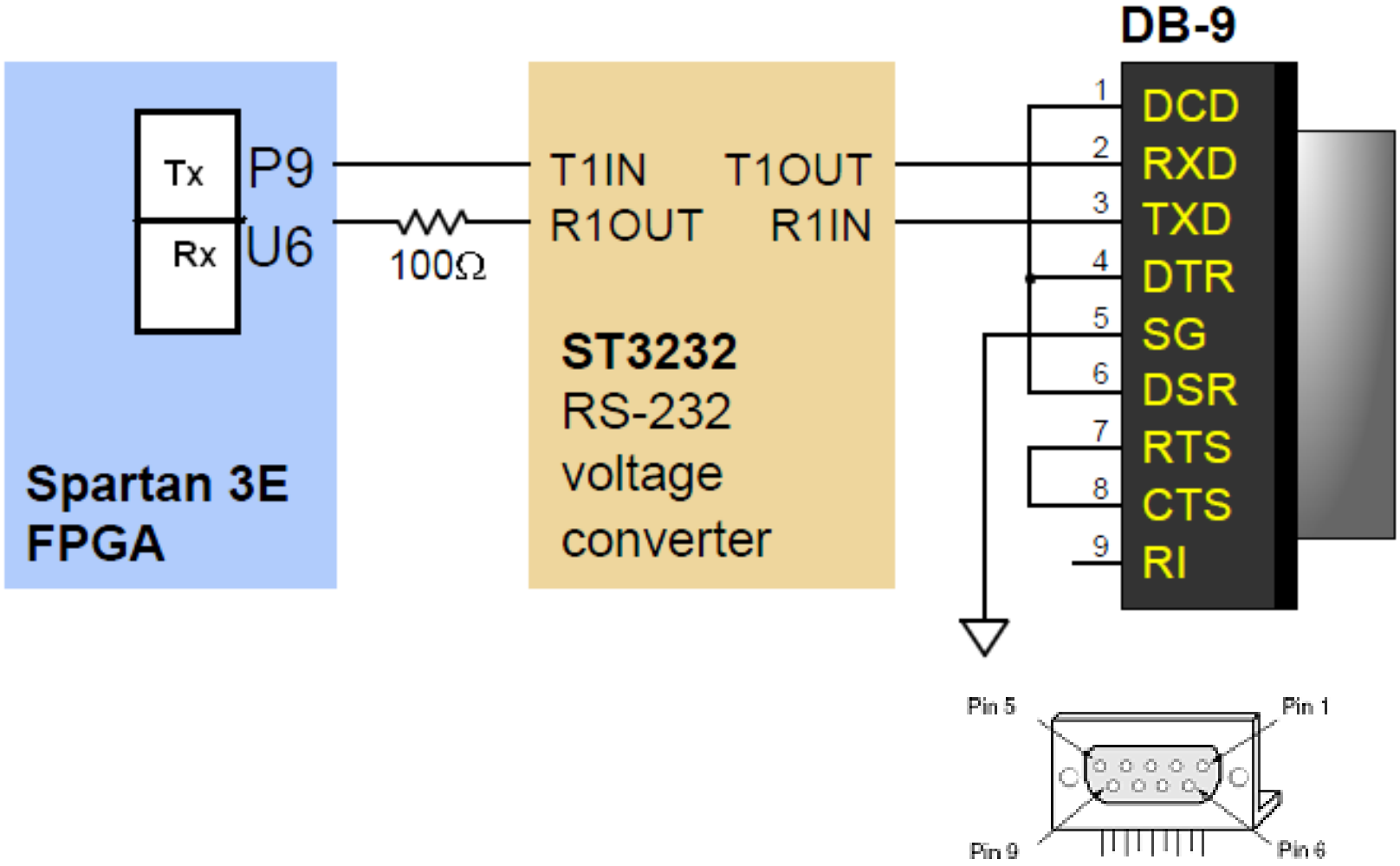
- Design & Implementation RS232 Transceiver on FPGA
- Verification
  - Simulation
  - Hardware

# RS232 Transceiver

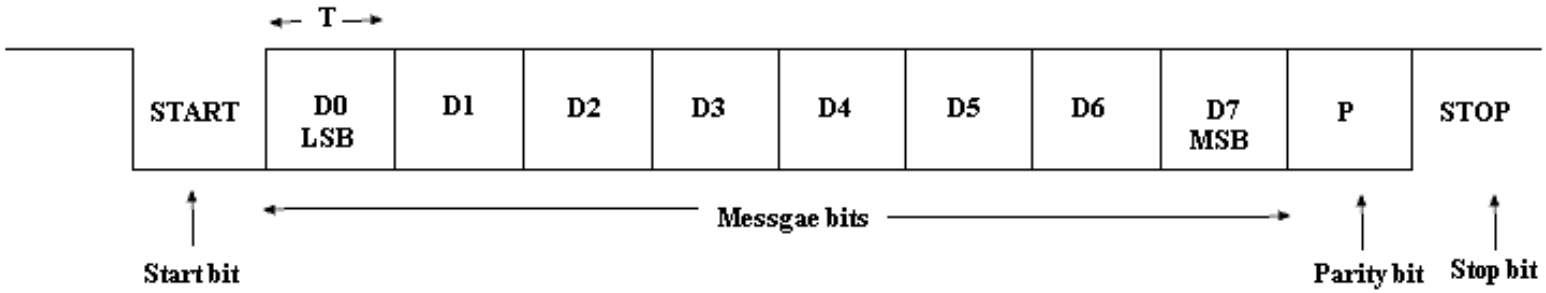



| DTE | | DCE |
|---|---|---|
| 2 Rxd | ← | 2 Txd |
| 3 Txd | → | 3 Rxd |

# Serial port in Nexys2



# RS232 Specification




| Baud Rate | T |
|---|---|
| 110 | 9.09 ms |
| 300 | 3.33 ms |
| 1200 | 833 us |
| 2400 | 417 us |
| 4800 | 208 us |
| 9600 | 104 us |
| 19200 | 52 us |
| 115200 | 8.6 us |

The parity bit is disabled. Therefore the efficient data rate be (115200*(8/10))/8 bits/sec that is equal to 11520 bits/sec.

# Receiver Design (Rxd)

```
entity Rs232Rxd is
    port( Reset, Clock16x, Rxd: in std_logic;
          DataOut1: out std_logic_vector (7 downto 0));
end Rs232Rxd;
```

# Rxd- Clock 16x

```
process (SystemClock)
begin

if SystemClock'event and SystemClock = '1' then
    if Reset = '1' then
        iCount9 <= (others=>'0');
    elsif
        iCount9 = "101000101" then -- the divider is 325, or "101000101"
        iCount9 <= (others=>'0');
else iCount9 <= iCount9 + '1';
end if;
end if;

end process;

iClock16x <= iCount9(8);
```

153846Hz, which is equal to 50MHz/325
16 * baudrate = 153600Hz

# Rxd -START Bit Detection

```
process (Clock16x)
begin
if rising_edge(Clock16x) then
    if Reset = '1' or iReset = '1' then
        iRxd1 <= '1';
        iRxd2 <= '1';
        iClock1xEnable <= '0';
        iClockDiv <= (others=>'0');
    else
        iRxd1 <= Rxd;
        iRxd2 <= iRxd1;
    end if;
    if iRxd1 = '0' and iRxd2 = '1' then
        iClock1xEnable <= '1';
    end if;
    if iClock1xEnable = '1' then
        iClockDiv <= iClockDiv + '1';
    end if;
end if;
end process;

iClock1x <= iClockDiv(3);
```

Detection of falling edge

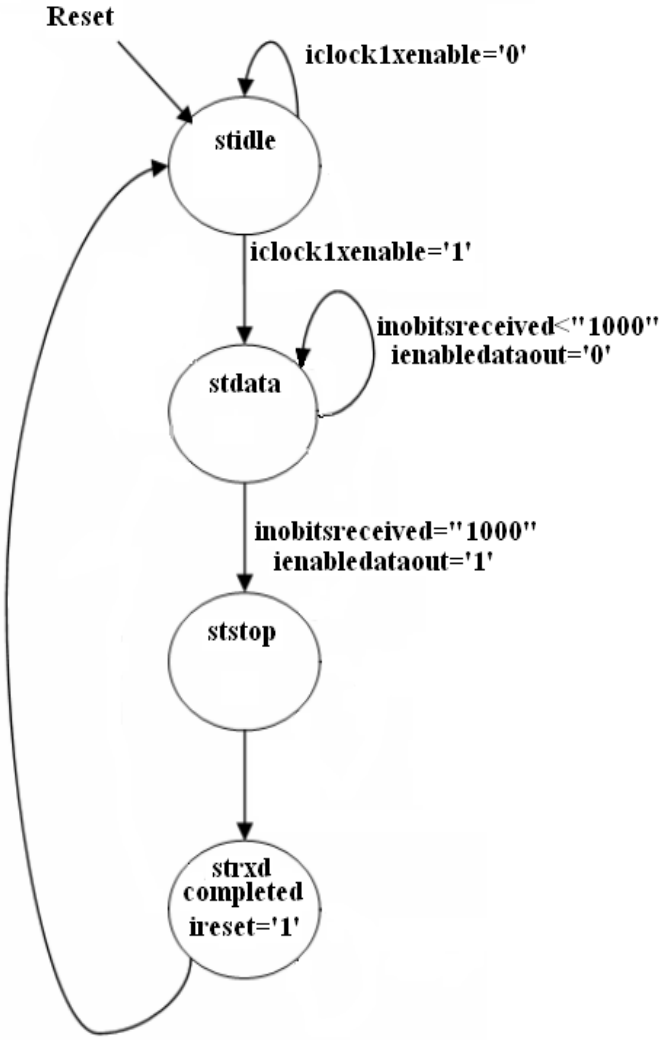
Generation of 9600Hz clock

# Rxd - Synchronous state machine

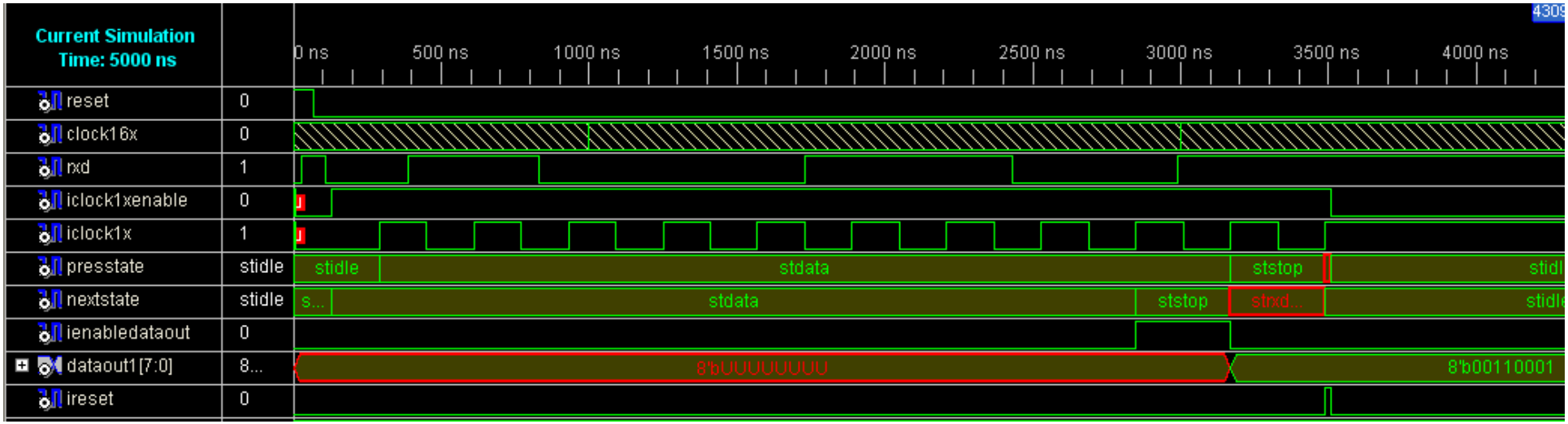
```
process (iClock1xEnable, iClock1x)
begin

if iClock1xEnable = '0' then
   iNoBitsReceived <= (others=>'0');
   presState <= stIdle;
elsif rising_edge(iClock1x) then
   iNoBitsReceived <= iNoBitsReceived + '1';
   presState <= nextState;
end if;

if rising_edge(iClock1x) then
   if iEnableDataOut = '1' then
      iDataOut1 <= iShiftRegister;
   else
      iShiftRegister <= Rxd & iShiftRegister(7 downto 1);
   end if;
end if;

end process;

DataOut1 <= iDataOut1;
```

# Rxd - Asynchronous state machine

```
process (presState, iClock1xEnable, iNoBitsReceived)
begin
iReset <= '0';
iEnableDataOut <= '0';
case presState is
   when stIdle =>
      if iClock1xEnable = '1' then
         nextState <= stData;
      else
         nextState <= stIdle;
      end if;
   when stData =>
      if iNoBitsReceived = "1001" then
         iEnableDataOut <= '1';
         nextState <= stStop;
      else
         iEnableDataOut <= '0';
         nextState <= stData;
      end if;
   when stStop =>
      nextState <= stRxdCompleted;
   when stRxdCompleted =>
      iReset <= '1';
      nextState <= stIdle;
   end case;

end process;
```



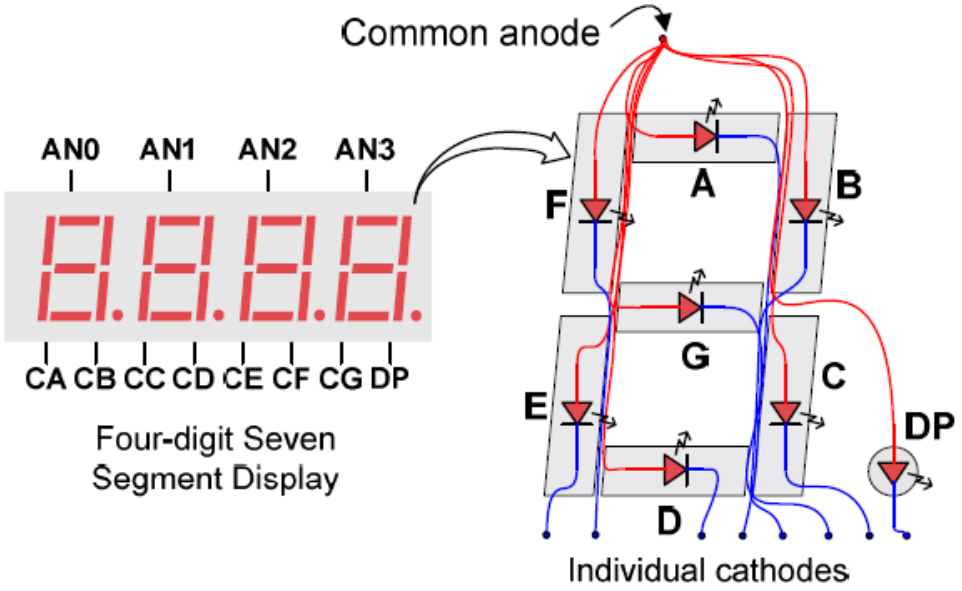
# Rxd - Simulation



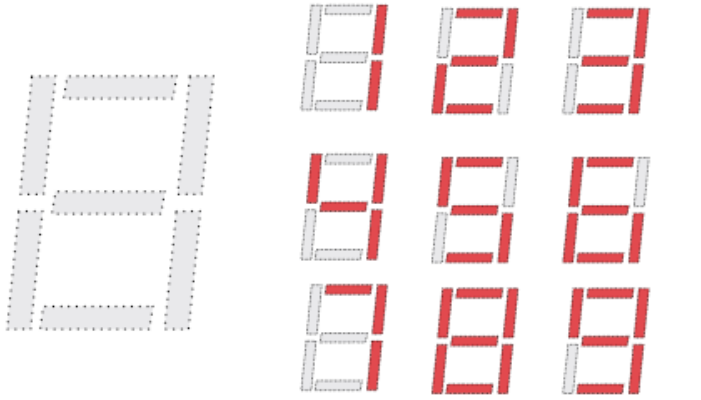
# Transmitter

- Read the data in parallel
  - Use slide switches
  - Start signal
- Transmit in Serial

# 7 Segment Display




Four-digit Seven Segment Display



An un-illuminated seven-segment display, and nine illumination patterns corresponding to decimal digits

# 7 Segment LED -HDL Design

```
entity decoder is
   Port(       Q : in std_logic_vector(3 downto 0);
         Seg : out std_logic_vector(6 downto 0));
end decoder;

architecture Behavioral of decoder is

begin
      Seg <="0000001" when q = "0000" else
            "1001111" when q = "0001" else
            "0010010" when q = "0010" else
            "0000110" when q = "0011" else
            "1001100" when q = "0100" else
            "0100100" when q = "0101" else
            "0100000" when q = "0110" else
            "0001111" when q = "0111" else
            "0000000" when q = "1000" else
            "0000100" when q = "1001" else
            "0001000" when q = "1010" else
            "1100000" when q = "1011" else
            "0110001" when q = "1100" else
            "1000010" when q = "1101" else
            "0110000" when q = "1110" else
            "0111000" when q = "1111" else
            "1111111";
end Behavioral;
```

-- Segment encoding

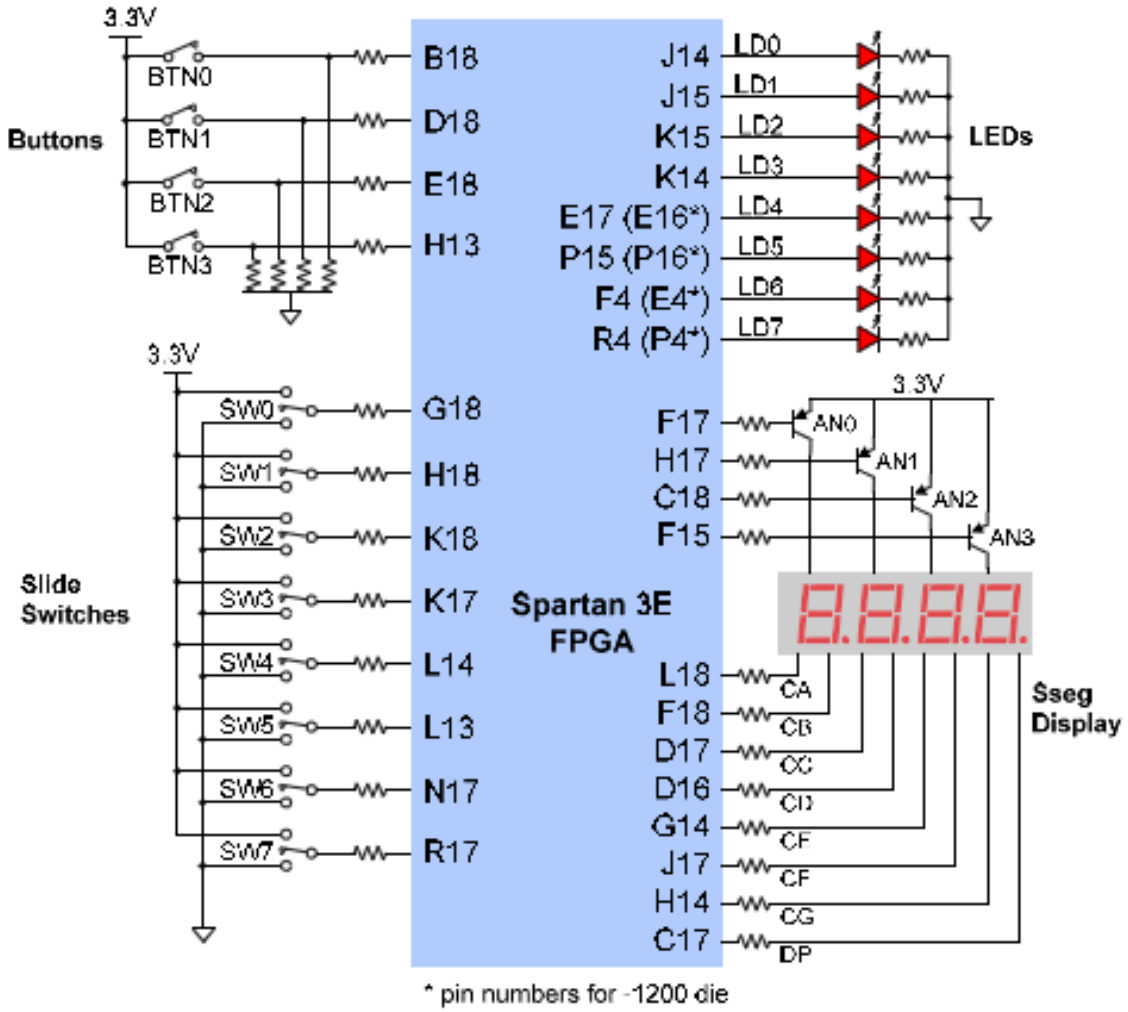
```
      a
     ---
  f |   | b
     --- <- g
  e |   | c
     ---
      d
```

LED seg order

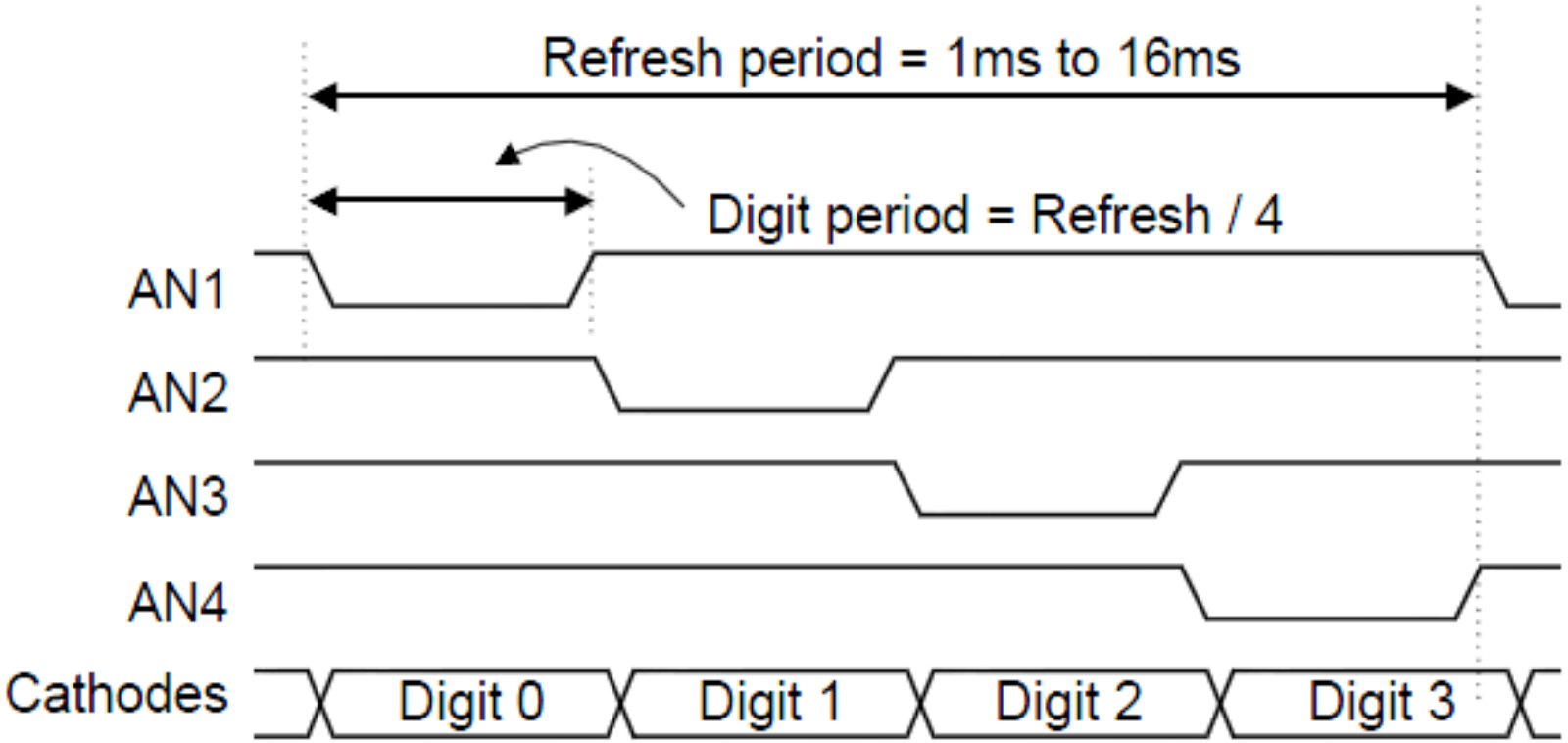
a, b, c, d, e, f, g

seg6, seg5, seg4, seg3, seg2, seg1, seg0

# 7 segment LED – Nexys2



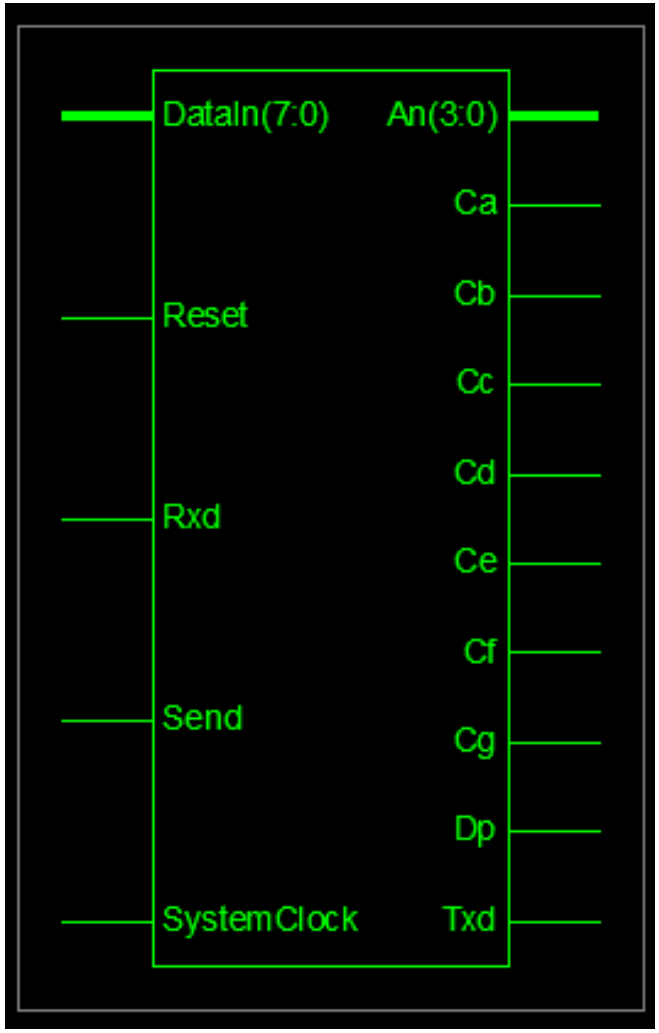
# Scanning display controller circuit



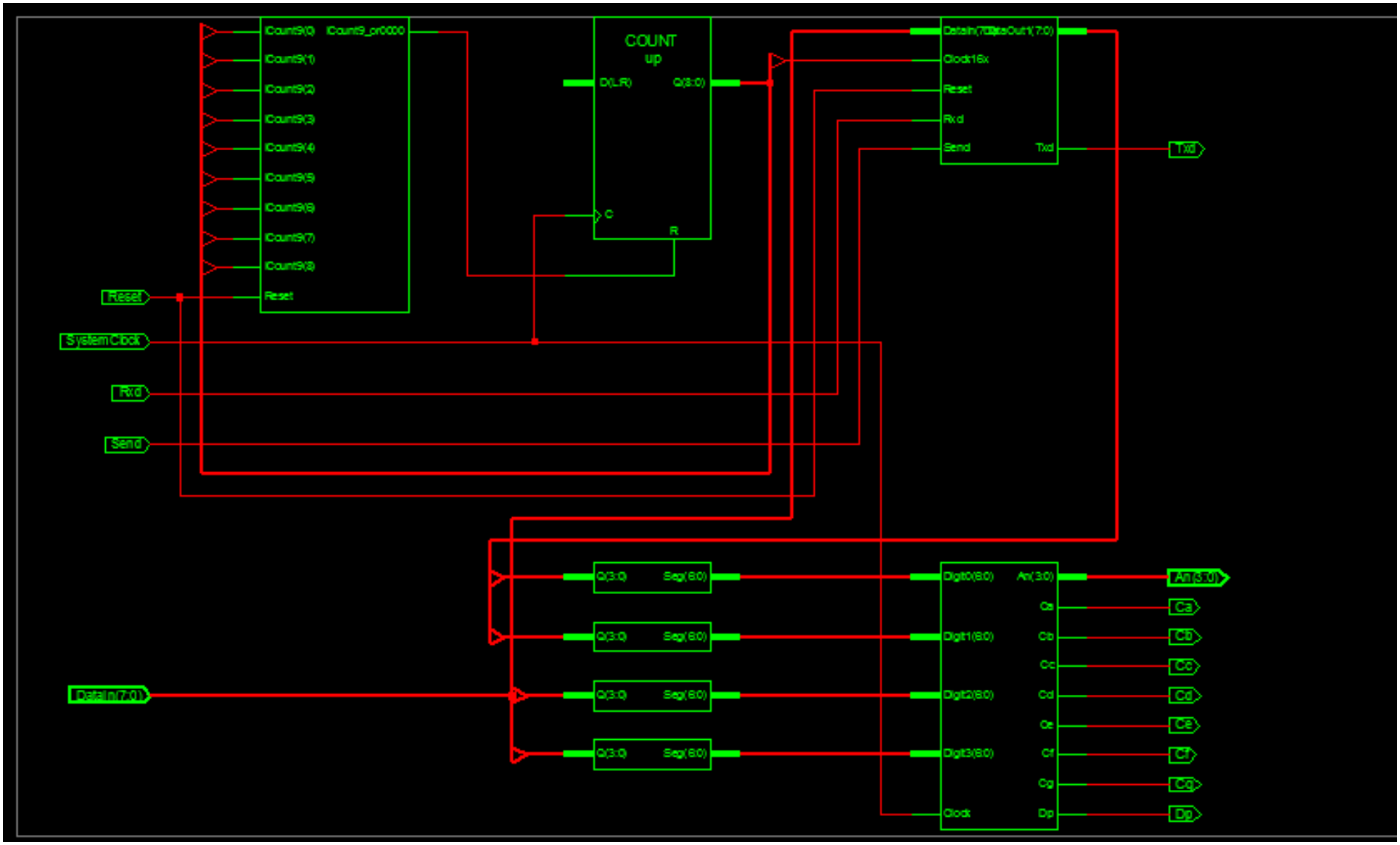
# Scan Display – HDL Design

```
-- Generate the scan clock 50MHz/2**16 (763Hz)
process(Clock)
begin
if Clock'event and Clock='1' then
   iCount16 <= iCount16 + '1';
end if;
end process;
--Send four digits to four 7-segment display using scan mode
with iCount16 (15 downto 14) select
   iDigitOut <=   Digit0 when "00", -- Connect Digit0 to the 7-segment display
                  Digit1 when "01", -- Connect Digit1 to the 7-segment display
                  Digit2 when "10", -- Connect Digit2 to the 7-segment display
                  Digit3 when "11", -- Connect Digit3 to the 7-segment display
                  Digit0 when others;

with iCount16 (15 downto 14) select
   An <= "1110" when "00", -- with AN0 low only
         "1101" when "01", -- with AN1 low only
         "1011" when "10", -- with AN2 low only
         "0111" when "11", -- with AN3 low only
         "1110" when others;
```

# RS 232 Top Level



# RTL View – RS232



# ASCII - Hex

| ASCII (HEX) | SYMBOL | ASCII (HEX) | SYMBOL | ASCII (HEX) | SYMBOL | ASCII (HEX) | SYMBOL |
|---|---|---|---|---|---|---|---|
| 00 | NUL | 20 | (SPACE) | 40 | @ | 60 | ` |
| 01 | SOH | 21 | ! | 41 | A | 61 | a |
| 02 | STX | 22 | “ | 42 | B | 62 | b |
| 03 | ETX | 23 | # | 43 | C | 63 | c |
| 04 | EOT | 24 | $ | 44 | D | 64 | d |
| 05 | ENQ | 25 | % | 45 | E | 65 | e |
| 06 | ACK | 26 | & | 46 | F | 66 | f |
| 07 | BEL | 27 | ‘ | 47 | G | 67 | g |
| 08 | BS | 28 | ( | 48 | H | 68 | h |
| 09 | TAB | 29 | ) | 49 | I | 69 | i |
| 0A | LF | 2A | * | 4A | J | 6A | j |
| 0B | VT | 2B | + | 4B | K | 6B | k |
| 0C | FF | 2C | , | 4C | L | 6C | l |
| 0D | CR | 2D | - | 4D | M | 6D | m |
| 0E | SO | 2E | . | 4E | N | 6E | n |
| 0F | SI | 2F | / | 4F | O | 6F | o |
| 10 | DLE | 30 | 0 | 50 | P | 70 | p |
| 11 | DC1 | 31 | 1 | 51 | Q | 71 | q |
| 12 | DC2 | 32 | 2 | 52 | R | 72 | r |
| 13 | DC3 | 33 | 3 | 53 | S | 73 | s |
| 14 | DC4 | 34 | 4 | 54 | T | 74 | t |
| 15 | NAK | 35 | 5 | 55 | U | 75 | u |
| 16 | SYN | 36 | 6 | 56 | V | 76 | v |
| 17 | ETB | 37 | 7 | 57 | W | 77 | w |
| 18 | CAN | 38 | 8 | 58 | X | 78 | x |
| 19 | EM | 39 | 9 | 59 | Y | 79 | y |
| 1A | SUB | 3A | : | 5A | Z | 7A | z |
| 1B | ESC | 3B | ; | 5B | [ | 7B | { |
| 1C | FS | 3C | < | 5C | \ | 7C | \| |
| 1D | GS | 3D | = | 5D | ] | 7D | } |
| 1E | RS | 3E | > | 5E | ^ | 7E | ~ |
| 1F | US | 3F | ? | 5F | _ | 7F | |

# THANK YOU