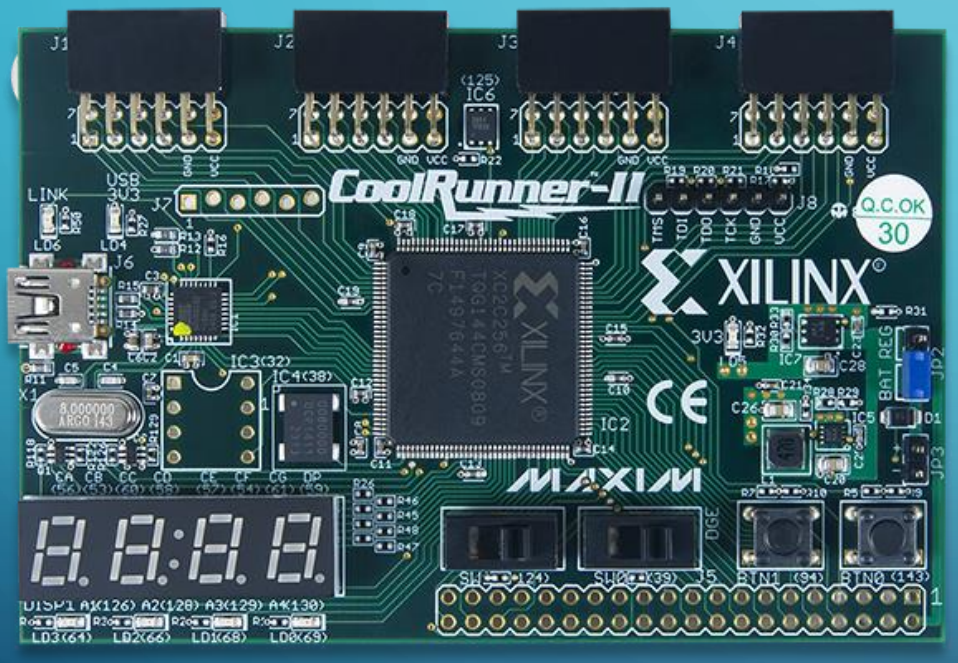
BINARY TO BCD CONVERTER WITH N BITS

CICT Project

ALBA RUIZ GÓMEZ AND NORMA SCHULZE GIMÉNEZ

CONTENT



- Introduction
- Utility of BCD
- Theoretical Basis
- Instruments used
- Connection board
- Implementation
- Examples
- Conclusion

Introduction

- Application: 16 bit Binary to 4 digit BCD converter
- Objective: Visualization of Binary to BCD converter
- Instruments used:
  - CoolRunner-II CPLD starter board
  - Expansion switches and LEDs board

Utility of BCD

- Electronic systems where numeric values must be shown
- Some companies such as IBM used it in their products: IBM1620 and IBM 1400

Theoretical basis

- BCD: Binary -Coded Binary
- Only encodes ten digits from 0 to 9
- There are six invalid code words: 1010, 1011, 1100, 1101, 1110, 1111

Decimal	Binary	BCD
0	0000	0000
1	0001	0001
2	0010	0010
3	0011	0011
4	0100	0100
5	0101	0101
6	0110	0110
7	0111	0111
8	1000	1000
9	1001	1001
10	1010	0001 0000
11	1011	0001 0001
12	1100	0001 0010
13	1101	0001 0011
14	1110	0001 0100
15	1111	0001 0101

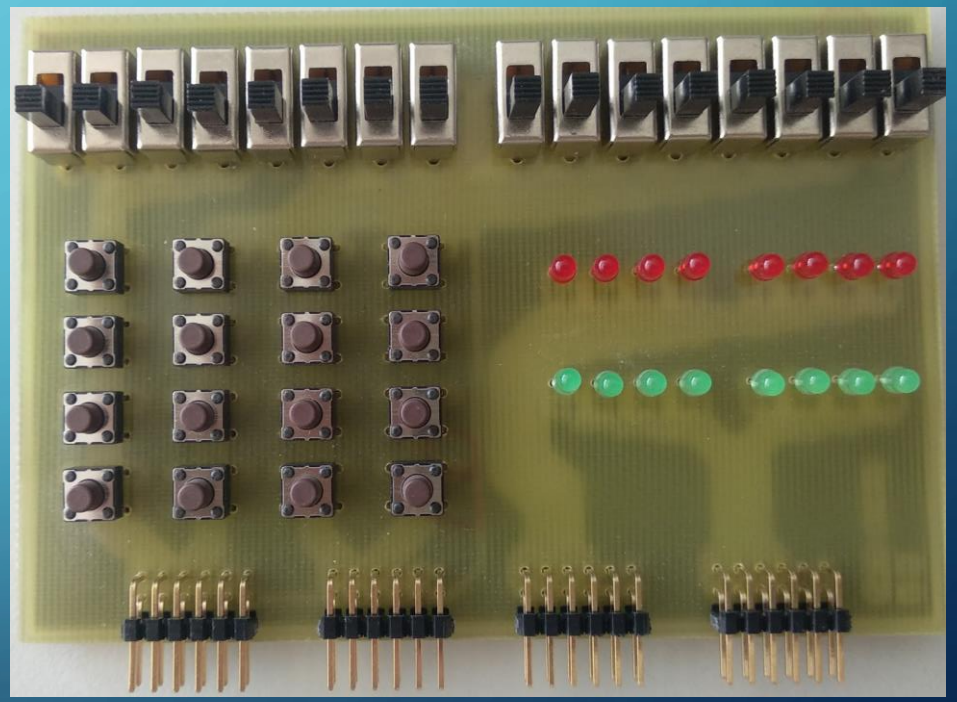
Theoretical basis

OPERATION	BCD			BINARY
	HUNDREDS (4 BITS)	TENS (4 BITS)	UNITS (4 BITS)	BINARY NUMBER (9 BITS)
Initial state				110100001
Shift to the left (1)			1	10100001
Shift to the left (2)			11	0100001
Shift to the left (3)			110	100001
Add three to the units			1001	100001
Shift to the left (4)		1	0011	00001
Shift to the left (5)		10	0110	0001
Add three to the units		10	1001	0001
Shift to the left (6)		101	0010	001
Add three to the tens		1000	0010	001
Shift to the left (7)	1	0000	0100	01

1. Shift the binary number to the left one bit
2. If some of the digits has a value equal or higher than five, add three
3. Repeat steps 1 and 2 the amounts of bits that the binary number that we want to convert.

Instruments used

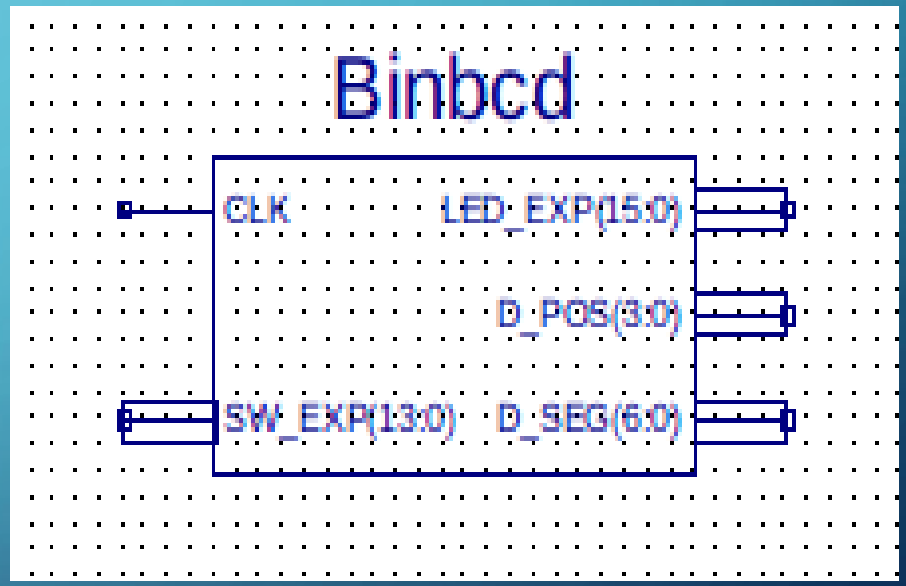
+



Connection board

```
entity binbcd is
    GENERIC(
        NBITS  : integer :=  14; -- Number of bits of the binary number
        NOUT: integer := 16 -- Number of bits of the BCD number (output).
    );

    PORT(
        SW_EXP: in std_logic_vector(NBITS-1   downto 0); -- Input, switches (Binary)
        CLK: in std_logic;                               -- Clock signal
        LED_EXP: out std_logic_vector(NOUT-1 downto 0);  -- Output,LEDs (BCD)
        D_POS: out std_logic_vector(3 downto 0);         -- Display positions
        D_SEG: out std_logic_vector(6 downto 0)          -- Display segments
       );
end binbcd;
```



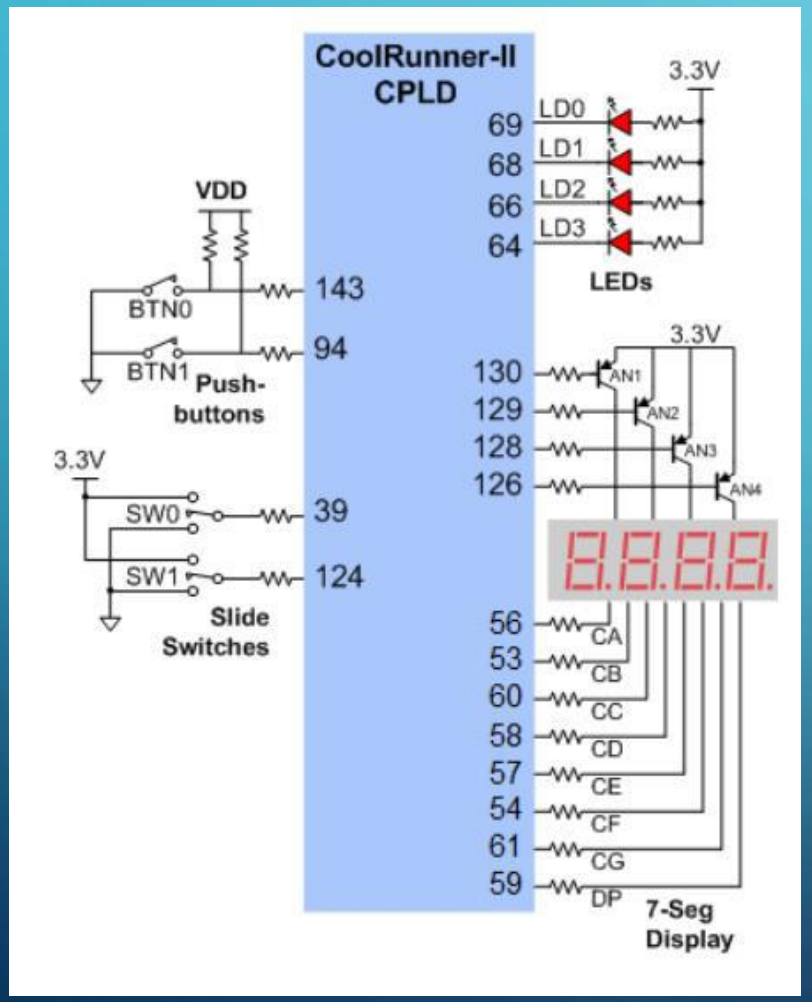
Implementation

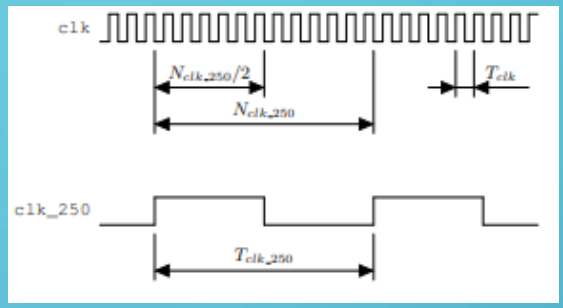
```
entity binbcd is
    GENERIC(
        NBITS  : integer :=  14; -- Number of bits of the binary number
        NOUT: integer := 16 -- Number of bits of the BCD number (output).
    );

    PORT(
        SW_EXP: in std_logic_vector(NBITS-1   downto 0); -- Input, switches (Binary)
        CLK: in std_logic;                               -- Clock signal
        LED_EXP: out std_logic_vector(NOUT-1 downto 0);  -- Output,LEDs (BCD)
        D_POS: out std_logic_vector(3 downto 0);         -- Display positions
        D_SEG: out std_logic_vector(6 downto 0)          -- Display segments
       );
end binbcd;
```

```
process_bcd: process(SW_EXP)
    variable z: STD_LOGIC_VECTOR(NBITS+NOUT-1 downto 0);
begin
    -- Initialization of data to zero
    z := (others => '0');
    -- First three left shifts
    z(NBITS+2 downto 3) := SW_EXP;
    -- Loop for the remaining shifts
    for i in 0 to NBITS-4 loop
        -- Units (4 bits).
        if z(NBITS+3 downto NBITS) > 4 then
            z(NBITS+3 downto NBITS) := z(NBITS+3 downto NBITS) + 3;
        end if;
        -- Tens (4 bits).
        if z(NBITS+7 downto NBITS+4) > 4 then
            z(NBITS+7 downto NBITS+4) := z(NBITS+7 downto NBITS+4) + 3;
        end if;
        -- Hundreds (4 bits).
        if z(NBITS+11 downto NBITS+8) > 4 then
            z(NBITS+11 downto NBITS+8) := z(NBITS+11 downto NBITS+8) + 3;
        end if;
        -- Thousands (4 bits).
        if z(NBITS+14 downto NBITS+12) > 4 then
            z(NBITS+14 downto NBITS+12) := z(NBITS+14 downto NBITS+12) + 3;
        end if;
        -- Shift to the left.
        z(NBITS+NOUT-1 downto 1) := z(NBITS+NOUT-2 downto 0);
    end loop;
    -- Assign z data to our BCD variable.
    num_bcd <= z(NBITS+NOUT-1 downto NBITS);
end process;
```

Implementation





Tx	N0.1ms	N0.1ms/2	N0.1ms/2
0.1ms	100	50	32

```
-- Switch between displays and its corresponding values --
process (clk_10)
begin
    if rising_edge(clk_10) then
        if position = "0111" then
            numdis <= num_bcd(3 downto 0);
            position <= "1110";
        elsif position = "1110" then
            numdis <= num_bcd(7 downto 4);
            position <= "1101";
        elsif position = "1101" then
            numdis <= num_bcd(11 downto 8);
            position <= "1011";
        elsif position <= "1011" then
            numdis <= num_bcd(15 downto 12);
            position <= "0111";
        end if;
     end if;
end process;
```

Examples


CoolRunner-II
XILINX


CoolRunner-II
XILINX


CoolRunner-II
XILINX


CoolRunner-II
XILINX

Conclusion

The manipulation of the numeric data to show becomes easier. This simplify the hardware.

Difficulties:

- understanding how the converter works.
- choosing the clock to display the value of the 7-segment display.
- problems with the device