Binary Coded Numbers Lesson
By Prawat Nagvajara

Synopsis
Binary Coded Number lesson includes a tutorial binary to decimal number representation and exercises at the end of the lesson. The lesson covers hierarchical design concepts and iterative array structure in the radix conversion algorithm.

Tutorial
An implementation of a combinational circuit converting 7-bit unsigned number \( x: \) in std_logic_vector(6 downto 0) to 2-digit decimal number \( z: \) out std_logic_vector(7 downto 0) where

\[
x(6)*64 + x(5)*32 + x(4)*16 + x(3)*8 + x(2)*4 + x(1)*2 + x(0) = z(1)*10 + z(0).
\]  

The conversion is based on accumulating the left side of (1) using an array of 2-digit BCD (Binary Coded Decimal) adders. The right side of (1), \( z = (z(1), z(0)) \) is a 2-digit BCD number, where \( z(i) \) is a 4-bits vector representing a decimal number. When \( x(i) = '1' \) the array accumulates \( 2^i, i=0,\ldots,6 \); The constants \( 2^i, 1 \leq i \leq 6 \) are 2-digit number. For instance, 64 is (“0110”,“0100”) and 8 is (“0000”,“1000”). Figure 1 shows an array of seven 2-digit BCD adders accumulating the \( 2^i, 1 = 0, \ldots, 6 \), constants. Depending whether the \( x(i) \) is a ‘1’ or ‘0’, the design uses AND-gates to pass two 4-bit vectors representing \( 2^i \) as a 2-digit decimal number or (“0000”,“0000”) representing 00 decimal number.

Fig. 1 Array of 2-Digit BCD Adders Converting 6-bit Binary to 2-digit BCD

An VHDL declaration of the entity binary2bcd of the array (Fig. 1) is

```vhdl
entity binary2bcd is
port (x: in std_logic_vector(6 downto 0);
     z: out std_logic_vector(7 downto 0));
end binary2bcd;
```

The 2-digit constants in Fig.1 – 00, 64, 32 and 01 are internal signals each digit is a 4-bit vector.
**N-digit BDC Adder**
The BCD Add blocks in Fig. 1 are copies of a two-digit BCD adder. An N-digit BCD (Fig. 2) is an array of BCD adder where \( x(i) \), \( y(i) \) and \( c(i) \) are binary coded decimal representation (4-bit vector).

![Fig. 2 N-digit Binary Coded Decimal Adder](image)

A two-digit BCD declaration is

```vhdl
entity bcd2 is
  port (a, b: in bcd_vector(1 downto 0);
        c: out bcd_vector(1 downto 0);
        cin: in std_logic;
        cout: out std_logic);
end bcd2;
```

where `bcd_vector` is an array of bcd signal – `std_logic_vector(3 downto 0)`. The `bcd_vector` is declared in a package as

```vhdl
package binary2bcdpack is
type bcd_vector is array (natural range <>) of std_logic_vector(3 downto 0);
end binary2bcdpack;
```

**BCD adder**
This section shows a VHDL code for the BCD add block used in Fig. 2.

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity bcd is
  port (a, b: in std_logic_vector(3 downto 0);
        c: out std_logic_vector(3 downto 0);
        cin: in std_logic;
        cout: out std_logic);
end bcd;
```

architecture Behavioral of bcd is
begin
process(a, b, cin)
  variable temp, tempa, tempb: std_logic_vector(4 downto 0);
begin
  tempa := '0'&a;
```
tempb := '0'&b;
temp := tempa + tempb;
if cin = '1' then temp := temp + "00001" end if;
if temp > "01001" then temp := temp + "00110" end if;
else cout <= '0';
end if;
c <= temp(3 downto 0);
end process;
end Behavioral;

Two-digit BCD adder
This section shows a VHDL code for a two-digit BCD adder (Fig.2 with \( N = 2 \)).

library IEEE;
use IEEE.STD_LOGIC_1164.all;

package binary2bcdpack is
type bcd_vector is array (natural range <>) of std_logic_vector(3 downto 0);
end binary2bcdpack;

library IEEE;
use IEEE.STD_LOGIC_1164.ALL; use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use work.binary2bcdpack.all;

entity bcd2 is
port (a, b: in bcd_vector(1 downto 0);
c: out bcd_vector(1 downto 0);
cin: in std_logic;
cout: out std_logic);
end bcd2;

architecture struc of bcd2 is
signal carry: std_logic_vector(2 downto 0);
component bcd
port (a, b: in std_logic_vector(3 downto 0);
c: out std_logic_vector(3 downto 0);
cin: in std_logic;
cout: out std_logic);
end component;
begin
carry(0) <= cin; cout <= carry(2);
G1: for i in 0 to 1 generate
U1: bcd port map (a(i), b(i), c(i), carry(i), carry(i+1));
end generate G1;
end struc;

Converting 6-bit Binary Number to 2-digit Binary Coded Decimal
This section shows a VHDL code for the circuit shown in Fig. 1.

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use work.binary2bcdpack.all;
entity binary2bcd is
port (x: in std_logic_vector(6 downto 0);
     z: out std_logic_vector(7 downto 0));
end binary2bcd;

architecture struc of binary2bcd is
begin
  type bcd2_vector is array (natural range <>) of bcd_vector(1 downto 0);
  signal p_s, temp, bcd2_template : bcd2_vector(6 downto 0);
  signal carry : std_logic_vector(6 downto 0);
  signal tempz : bcd_vector(1 downto 0);--temporary signal for output ports

  component bcd2
  port (a, b: in bcd_vector(1 downto 0);
        c: out bcd_vector(1 downto 0);
        cin: in std_logic;
        cout: out std_logic);
  end component;

  bcd2_template(0) <= ("0000","0001"); bcd2_template(1) <= ("0000","0010");
  bcd2_template(2) <= ("0000","0100"); bcd2_template(3) <= ("0000","1000");
  bcd2_template(4) <= ("0001","0110"); bcd2_template(5) <= ("0011","0010");
  bcd2_template(6) <= ("0110","0100");

  process(x)
  begin
    for i in 0 to 6 loop
      if x(i) = '0' then temp(i) <= ("0000","0000");
      else temp(i) <= bcd2_template(i);
      end if;
    end loop;
  end process;

  p_s(0) <= ("0000","0000"); carry(0) <= '0';
  z(7 downto 4) <= tempz(1); z(3 downto 0) <= tempz(0);

  -- initial signals for partial sum and wires to output ports

  -- wire the array
  G1: for i in 0 to 6 generate
  G2: if i < 6 generate
    U2: bcd2 port map (temp(i), p_s(i), p_s(i+1), carry(i), carry(i+1));
  end generate G2;
  G3: if i = 6 generate
    U2: bcd2 port map (temp(i), p_s(i), tempz, carry(i), open);
  end generate G3;
  end generate G1;

end struc;

Exercises

1. Implement converting 6-bit binary number to 4-character Radix-3 number combinational circuit

\[
x(6)\times 64 + x(5)\times 32 + x(4)\times 16 + x(3)\times 8 + x(2)\times 4 + x(1)\times 2 + x(0) = z(3)\times 27 + z(2)\times 9 + z(1)\times 3 + z(0)\]

(2)
using array of 4-character binary coded Base-3 (radix-3) adder. $z(i)$ is binary coded using a 2-bit vector. $z(i)$, for $i$ in 3 downto 0, are displayed on 8 LEDs. $z_i$, $i=0,...,6$ are in the binary coded of radix-3 positional number system, e.g., 64 is ("10", "01", "00", "01") = $2*27 + 1*9 + 0*3 +1$.

2. Radix-12 combinational adder, e.g., $x=B$ (11 in decimal), $y=9$, carry_in=1, the output $z = 9$, carry_out=1 (radix-12) (21 decimal)

3. Implement a 14-bit binary to 4-digit decimal converter.