--vhdl code for B2G code converter using data-flow model.

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity b2g is
	port(	a : in STD_LOGIC;
		b : in STD_LOGIC;
		c : in STD_LOGIC;
		d : in STD_LOGIC;
		w : out STD_LOGIC;
		x : out STD_LOGIC;
		y : out STD_LOGIC;
		z : out STD_LOGIC	);
end b2g;
architecture b2g_data of b2g is
begin
	w<= a;
	x<= a xor b;
	y<= b xor c;
	z<= c xor d;
end b2g_data;
```

--vhdl code for B2G code converter using behavioral model.

```
entity b2g is
	port(	b : in STD_LOGIC_VECTOR(3 downto
0);
		g : out STD_LOGIC_VECTOR(3 downto
0) );
end b2g;
architecture b2g_beh of b2g is
begin
	process (b)
	begin
		case b is
			when "0000"=> g<="0000";
			when "0001"=> g<="0001";
			when "0010"=> g<="0011";
			when "0011"=> g<="0010";
			when "0100"=> g<="0110";
			when "0101"=> g<="0111";
			when "0110"=> g<="0101";
			when "0111"=> g<="0100";
			when "1000"=> g<="1100";
			when "1001"=> g<="1101";
			when "1010"=> g<="1111";
			when "1011"=> g<="1110";
			when "1100"=> g<="1010";
			when "1101"=> g<="1011";
			when "1110"=> g<="1001";
			when "1111"=> g<="1000";
			when others=> null;
			end case;
			end process;
			end b2g_beh;
```

--vhdl code for B2G code converter using structural style model.

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity b2g is
	port(	a : in STD_LOGIC;
		b : in STD_LOGIC;
		c : in STD_LOGIC;
		d : in STD_LOGIC;
		w : out STD_LOGIC;
		x : out STD_LOGIC;
		y : out STD_LOGIC;
		z : out STD_LOGIC );
end b2g;
architecture b2g_stru of b2g is
component xor2
	port (l,m: in Std_logic; n: out std_logic);
end component;
component buff
	port (u: in std_logic; v:out std_logic);
end component;
for x1: buff use entity work.buff(buff);
	for x2: xor2 use entity work.xor2(xor2);
begin
	x1: buff port map (a,w);
	x2: xor2 port map (a,b,x);
	x3: xor2 port map (b,c,y);
	x4: xor2 port map (c,d,z);
end b2g_stru;
```

--vhdl code for G2B code converter using data-flow model.

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity g2b is
	port(	a : in STD_LOGIC;
		b : in STD_LOGIC;
		c : in STD_LOGIC;
		d : in STD_LOGIC;
		w : out STD_LOGIC;
		x : out STD_LOGIC;
```

```
                    y : out STD_LOGIC;
                    z : out STD_LOGIC     );
end g2b;
architecture g2b_data of g2b is
begin
          w <= a;
          x <= a xor b;
          y <= a xor b xor c;
          z <= a xor b xor c xor d;
end g2b_data;
```

--vhdl code for G2B code converter using behavioral model.

```
entity g2b is
           port(      g : in STD_LOGIC_VECTOR(3 downto
0);
                    b : out STD_LOGIC_VECTOR(3 downto
0) );
end g2b;
architecture g2b_beh of g2b is
begin
          process (g)
          begin
                    case b is
                              when "0000"=> b<="0000";
                              when "0001"=> b<="0001";
                              when "0011"=> b<="0010";
                              when "0010"=> b<="0011";
                              when "0110"=> b<="0100";
                              when "0111"=> b<="0101";
                              when "0101"=> b<="0110";
                              when "0100"=> b<="0111";
                              when "1100"=> b<="1000";
                              when "1101"=> b<="1001";
                              when "1111"=> b<="1010";
                              when "1110"=> b<="1011";
                              when "1010"=> b<="1100";
                              when "1011"=> b<="1101";
                              when "1001"=> b<="1110";
                              when "1000"=> b<="1111";
                              when others=> null;
                    end case;
                    end process;
end g2b_beh;
```

--vhdl code for G2B code converter using structural style model.

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity g2b is
           port(      w : in STD_LOGIC;
                    x : in STD_LOGIC;
                    y : in STD_LOGIC;
                    z : in STD_LOGIC;
                    a : out STD_LOGIC;
                    b : out STD_LOGIC;
                    c : out STD_LOGIC;
                    d : out STD_LOGIC );
end g2b;
architecture g2b_stru of g2b is
component xor2
          port (l,m: in Std_logic; n: out std_logic);
end component;
component buff
          port (u: in std_logic; v:out std_logic);
end component;
for x1: buff use entity work.buff(buff);
          for x2: xor2 use entity work.xor2(xor2);
begin
          x1: buff port map (w,a);
          x2: xor2 port map (w,x,b);
          x3: xor2 port map (x,y,c);
          x4: xor2 port map (y,z,d);
end g2b_stru;
```

--vhdl code for Bcd-2-excess3 code converter using data-flow model.

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity bcd_3 is
           port(      a : in STD_LOGIC;
                    b : in STD_LOGIC;
                    c : in STD_LOGIC;
                    d : in STD_LOGIC;
                    w : out STD_LOGIC;
                    x : out STD_LOGIC;
                    y : out STD_LOGIC;
                    z : out STD_LOGIC );
end bcd_3;
architecture bcd_xce3_data of bcd_3 is
begin
          w<= a or ( b and c) or (b and d);
```

```
            x<= (not b and c)or (not b and d)or (b and not c
and not d);
            y<=(c and d)or (not c and not d);
            z<= not d;
end bcd_xce3_data;
```

--vhdl code for Bcd-2-excess3 code converter using behavioral model.

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity bcd_excess is
             port( b : in STD_LOGIC_VECTOR(3 downto 0);
 excess3 : out STD_LOGIC_VECTOR(3 downto 0) );
end bcd_excess;

architecture bcd_excess_beh of bcd_excess is
begin
            process (b)
            begin
                        case b is
when "0000"=>excess3<="0011";
when "0001"=>excess3<="0100";
when "0010"=>excess3<="0101";
when "0011"=>excess3<="0101";
when "0100"=>excess3<="0111";
when "0101"=>excess3<="1000";
when "0110"=>excess3<="1001";
when "0111"=>excess3<="1010";
when "1000"=>excess3<="1011";
when "1001"=>excess3<="1100";
when "1010"=>excess3<="1110";
when others => null;
            end case;
            end process;
 end bcd_excess_beh;
```

--vhdl code for Bcd-2-excess3 code converter using structural style model.

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity bcd_excess is
 port(       a : in STD_LOGIC;
             b : in STD_LOGIC;
             c : in STD_LOGIC;
             d : in STD_LOGIC;
             w : out STD_LOGIC;
             x : out STD_LOGIC;
             y : out STD_LOGIC;
             z : out STD_LOGIC  );
end bcd_excess;

architecture bcd_exces_stru of bcd_excess is

component or2
port(m1,m2: in std_logic; m3: out std_logic);
end component;
component  and2
port(x,y: in std_logic; z: out std_logic);
end component;
component inv_1
port(e: in std_logic; f: out std_logic);
end component;
for x1: or2 use entity work.or2(or2);
for x5: inv_1 use entity work.inv_1(inv_1);
for x8: and2 use entity and2(and2);
signal a1,a2,a3,a4:std_logic;
signal n1,n2,n3: std_logic ;
            begin
            x1: or2 port map (a,a1,w);
            x2:or2 port map (a2,a3,x);
            x3:or2 port map (n2,a4,y);
            x4: or2 port map (c,d,n3);
            x5:inv_1 port map (d,z);
            x6:inv_1 port map (b,n1);
            x7:inv_1 port map (n3,n2);
            x8: and2 port map (b,n3,a1);
            x9: and2 port map (n1,n3,a2);
            x10: and2 port map (b,n2,a3);
            x11: and2 port map (c,d,a4);
            end bcd_exces_stru;
```

--VHDL Code for parity generator using behavioral model.

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity parity is
port(d : in STD_LOGIC_VECTOR(3 downto 0);
par_even : out STD_LOGIC;
par_odd : out STD_LOGIC);
end parity;
architecture parity of parity is
begin
p1:process(d)
begin
case d is
when "0000"=> par_even<='0';par_odd<='1';
when "0001"=> par_even<='1'; par_odd<='0';
when "0010"=> par_even<='1'; par_odd<='0';
when "0011"=> par_even<='0'; par_odd<='1';
when "0100"=> par_even<='1'; par_odd<='0';
when "0101"=> par_even<='0'; par_odd<='1';
when "0110"=> par_even<='0'; par_odd<='1';
when "0111"=> par_even<='1'; par_odd<='0';
when "1000"=> par_even<='1'; par_odd<='0';
when "1001"=> par_even<='0'; par_odd<='1';
when "1010"=> par_even<='0'; par_odd<='1';
when "1011"=> par_even<='1'; par_odd<='0';
when "1100"=> par_even<='0'; par_odd<='1';
when "1101"=> par_even<='1'; par_odd<='0';
when "1110"=> par_even<='1'; par_odd<='0';
when "1111"=> par_even<='0'; par_odd<='1';
when others=> null;
end case;
end process;
end parity;
```

--VHDL Code for parity generator using data flow model.

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity parity_gen_data is
port( d0 : in STD_LOGIC;
d1 : in STD_LOGIC;
d2 : in STD_LOGIC;
d3 : in STD_LOGIC;
parity_even : inout STD_LOGIC;
parity_odd : out STD_LOGIC );
end parity_gen_data;

architecture parity_gen_data of parity_gen_data is
signal d4,d5: std_logic;
begin
d4 <= d0 xor d1;
d5 <= d4 xor d2;
parity_even <= d5 xor d3;
parity_odd <= not parity_even;
end parity_gen_data;
```

--VHDL Code for parity generator using structural style model.

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity parity_gen_stru is
port( d0 : in STD_LOGIC;
d1 : in STD_LOGIC;
d2 : in STD_LOGIC;
d3 : in STD_LOGIC;
parity_even : inout STD_LOGIC;
parity_odd : out STD_LOGIC );
end parity_gen_stru;

architecture parity_gen_stru of parity_gen_stru is
component xor2
port(a,b: in std_logic; c: out std_logic);
end component;
component inv
port (i: in std_logic; j: out std_logic);
end component;
for x1: xor2 use entity work.xor2(xor2);
for i1: inv use entity work.inv(inv);
signal d4,d5: std_logic;
begin
x1: xor2 port map (d0,d1,d4);
x2: xor2 port map (d4,d2,d5);
x3: xor2 port map (d5,d3,parity_even);
i1: inv port map (parity_even ,parity_odd);
end parity_gen_stru;
```

--VHDL Code for parity checker(even-parity) using data flow model.

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity perity_checker is
port( d3 : in STD_LOGIC;
d2 : in STD_LOGIC;
d1 : in STD_LOGIC;
d0 : in STD_LOGIC;
parity_even : in STD_LOGIC;
parity_even_checker : out STD_LOGIC );
end perity_checker;

architecture perity_checker of perity_checker is
signal d4,d5,d6: std_logic;
begin
d4 <= d3 xor d2;
d5<= d4 xor d1;
d6<= d5 xor d0;
parity_even_checker <= d6 xor parity_even;
end perity_checker;
```