package data\_types\_pkg is

type X\_Record is record

bin : std\_logic\_vector(9 downto 0);

hex : std\_logic\_vector(15 downto 0);

uint : integer range 0 to 1023;

end record;

type Y\_Record is record

dec : integer range 0 to 1023;

gray : std\_logic\_vector(15 downto 0);

bcd : std\_logic\_vector(11 downto 0); -- For 3-digit decimal BCD

end record;

end package;

library IEEE;

use IEEE.std\_logic\_1164.all;

use IEEE.numeric\_std.all;

use work.data\_types\_pkg.all;

entity DataConverter is

Port (

clk : in std\_logic;

A : in integer range 0 to 1023;

B : in std\_logic\_vector(9 downto 0);

X : out X\_Record;

Y : out Y\_Record

);

end DataConverter;

architecture Behavioral of DataConverter is

signal bin\_B : std\_logic\_vector(9 downto 0);

begin

process(clk)

variable gray\_val : std\_logic\_vector(16 downto 0);

variable bcd\_val : std\_logic\_vector(11 downto 0);

variable dec\_val : integer;

variable i : integer;

begin

if rising\_edge(clk) then

-- A to X\_Record

X.uint <= A;

X.bin <= std\_logic\_vector(to\_unsigned(A, 10));

X.hex <= std\_logic\_vector(to\_unsigned(A mod 16, 4));

-- B to Y\_Record

bin\_B <= B;

dec\_val := 0;

-- Binary to Decimal (manual)

for i in 0 to 9 loop

dec\_val := dec\_val \* 2;

if bin\_B(i) = '1' then

dec\_val := dec\_val + 1;

end if;

end loop;

Y.dec <= dec\_val;

-- Binary to Gray code (manual)

gray\_val(9) := bin\_B(9);

for i in 8 downto 0 loop

gray\_val(i) := bin\_B(i+1) xor bin\_B(i);

end loop;

Y.gray <= gray\_val;

-- Binary to BCD (manual using simplified Double Dabble)

bcd\_val := (others => '0');

for i in 9 downto 0 loop

-- Shift left

bcd\_val := bcd\_val(10 downto 0) & bin\_B(i);

-- If any digit ≥ 5, add 3

if bcd\_val(3 downto 0) > "0100" then

bcd\_val(3 downto 0) := std\_logic\_vector(unsigned(bcd\_val(3 downto 0)) + 3);

end if;

if bcd\_val(7 downto 4) > "0100" then

bcd\_val(7 downto 4) := std\_logic\_vector(unsigned(bcd\_val(7 downto 4)) + 3);

end if;

if bcd\_val(11 downto 8) > "0100" then

bcd\_val(11 downto 8) := std\_logic\_vector(unsigned(bcd\_val(11 downto 8)) + 3);

end if;

end loop;

Y.bcd <= bcd\_val;

end if;

end process;

end Behavioral;