```
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;
```