```
------------------------------------------------------------------------------------------------------------------------
--
-- CORSAIR TECHNOLOGY CONFIDENTIAL
--
------------------------------------------------------------------------------------------------------------------------
-- (C) Copyright 2017 Corsair Technology Limited
-- All Rights Reserved
--
-- NOTICE: All information contained herein is, and remains the property of Corsair Technology Limited and its
-- suppliers, if any.  The intellectual and technical concepts contained herein are proprietary to Corsair Technology
-- Limited and its suppliers and may be covered by U.S. and Foreign Patents, patents in process, and are protected
-- by trade secret or copyright law. Dissemination of this information or reproduction of this material is strictly
-- forbidden unless prior written permission is obtained from Corsair Technology Limited.
--
-- File Name  : xxtea_encrypt.vhd
--
-- Author     : E Lister
--
-- Version    : V00.01
--
------------------------------------------------------------------------------------------------------------------------
--
-- Purpose :
--
------------------------------------------------------------------------------------------------------------------------
-- (C) Copyright 2017 Corsair Technology Limited. All rights reserved
------------------------------------------------------------------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all ;

------------------------------------------------------------------------------------------------------------------------
entity xxtea_encrypt is
  port
  (
    ----------------------------------------------------------------------------------------------------------------------
    -- Inputs
    ----------------------------------------------------------------------------------------------------------------------
    clk                       : in  std_logic ;
```

```
    en_encrypt                : in  std_logic ;
    encryption_key            : in  unsigned(127 downto  0) ;
    counter                   : in  unsigned( 63 downto  0) ;
    ------------------------------------------------------------------------------------------------------------------------
    -- Outputs
    ------------------------------------------------------------------------------------------------------------------------
    cypher                    : out unsigned( 63 downto  0)
  ) ;
end entity xxtea_encrypt ;

------------------------------------------------------------------------------------------------------------------------
architecture rtl of xxtea_encrypt is

------------------------------------------------------------------------------------------------------------------------
-- CONSTANTS
------------------------------------------------------------------------------------------------------------------------
constant DELTA                : unsigned(31 downto  0) := x"9e3779b9" ;

------------------------------------------------------------------------------------------------------------------------
-- SIGNALS
------------------------------------------------------------------------------------------------------------------------
signal MX                     : unsigned(31 downto  0) := (others => '0') ;
signal key                    : unsigned(31 downto  0) := (others => '0') ;
signal q                      : integer range 0 to 32  := 0 ;
signal sum                    : unsigned(31 downto  0) := (others => '0') ;
signal v0                     : unsigned(31 downto  0) := (others => '0') ;
signal v1                     : unsigned(31 downto  0) := (others => '0') ;
signal z                      : unsigned(31 downto  0) := (others => '0') ;
signal clk_count              : integer range 0 to 7   := 0 ;

------------------------------------------------------------------------------------------------------------------------
-- ALIAS
------------------------------------------------------------------------------------------------------------------------
alias e                       : unsigned( 1 downto  0) is sum(3 downto  2) ;

------------------------------------------------------------------------------------------------------------------------
begin

------------------------------------------------------------------------------------------------------------------------
```

```
-- Within the encryption process we use different 32-bit slices of the 128-bit encryption key. The selection of the
-- appropriate key slice is based upon the value of e for the unique case of only encoding a 64-bit counter value.
-------------------------------------------------------------------------------------------------------------------

key_selector : process (encryption_key, e) is
begin
  case (e) is
    when b"11" =>
      key                     <= encryption_key( 31 downto  0) ;
    when b"10" =>
      key                     <= encryption_key( 63 downto 32) ;
    when b"01" =>
      key                     <= encryption_key( 95 downto 64) ;
    when others =>
       key                    <= encryption_key(127 downto 96) ;
  end case ;
end process key_selector ;

-------------------------------------------------------------------------------------------------------------------
-- Process for generating the MX value. It takes one clock cycle for the value of Mx to be updated when either e, z
-- or sum change.
-------------------------------------------------------------------------------------------------------------------

MX_calculator : process (clk) is
begin
  if rising_edge(clk) then
    if (en_encrypt = '0') then
      MX                      <= (others => '0') ;
    else
      MX                      <= (((b"00000" & z(31 downto 5)) xor (z(29 downto  0) & b"00")) +
                                 ((b"000" & z(31 downto 3)) xor (z(27 downto  0) & b"0000")))
                                 xor ((key xor z) + (sum xor z)) ;
    end if ;
  end if ;
end process MX_calculator ;

-------------------------------------------------------------------------------------------------------------------
-- The encoding scheme uses two embedded loops with the indexes of q and clk_cnt. The clk_cnt loop is only in the range
-- 0 to 5 and the q loops is in the range 1 to 32.
```

```
--
-- The encryption process changes the values of the parameters that feed into the MX_calculator so this means that the
-- loop needs to run at a lower clock rate to allow the intermediate terms to be resolved. This reduced in clock rate is
-- achieved by using the clk_cnt value to allow all of the logic to operate at the same overall system clock rate.
--
-- The full encryption process takes a total of 192 clock cycles to resolve.
-------------------------------------------------------------------------------------------------------------------------
encryption_loop : process (clk) is
begin
  if rising_edge(clk) then
    if (en_encrypt = '0') then
      q                       <= 0 ;
      z                       <= counter(31 downto  0) ;
      v0                      <= counter(63 downto 32) ;
      v1                      <= counter(31 downto  0) ;
      sum                     <= (others => '0') ;
      clk_count               <= 0 ;
    else
      if (q /= 32) then
        case clk_count is
          when 0 =>
            sum               <= sum + DELTA ;
          when 3 =>
            z                 <= v0 + MX ;
            v0                <= v0 + MX ;
          when 5 =>
            z                 <= v1 + MX ;
            v1                <= v1 + MX ;
          when others =>
            null ;
        end case ;
        if (clk_count = 5) then
          clk_count           <= 0 ;
          q                   <= q + 1 ;
        else
          clk_count           <= clk_count + 1 ;
        end if ;
      end if ;
    end if ;
```

```
  end if ;
end process encryption_loop ;

-------------------------------------------------------------------------------------------------------------------------
-- Once the encryption is complete then output the cypher value.
-------------------------------------------------------------------------------------------------------------------------
cypher_output : process (clk) is
begin
  if rising_edge(clk) then
    if (q = 32) then
      cypher                  <= v0 & v1 ;
    end if ;
  end if ;
end process cypher_output ;

-------------------------------------------------------------------------------------------------------------------------
end architecture rtl ;
-------------------------------------------------------------------------------------------------------------------------
```