```
library ieee;
use ieee.std_logic_1164.all;
use work.DES_PACK.all;

entity DES is
    port (
        DATA:           inout   std_logic_vector (7 downto 0);
        CLOCK:          in      std_logic;
        RESET_N:        in      std_logic;
        DIE_N:          in      std_logic;
        DOE_N:          in      std_logic;
        KEY_N:          in      std_logic;
        ENCRYPT:        in      std_logic;
        DI_REQ_N:       out     std_logic;
        KEY_REQ_N:      out     std_logic;
        DO_RDY_N:       out     std_logic
    );
end ;

architecture BEHAVE of DES is

-- internal signals

    signal CLK:         std_logic;
    signal RESET:       std_logic;
    signal DIN:         std_logic_vector (1 to 8);
    signal DOUT:        std_logic_vector (1 to 8);

    signal DIREQ_N:     std_logic;
    signal DORDY_N:     std_logic;
    signal KEYREQ_N:    std_logic;
    signal DIE:         std_logic;
    signal DOE:         std_logic;
    signal KEYE:        std_logic;
    signal ENCR:        std_logic;

    signal PIN_OE:      std_logic := ’1’;

  -- Left/Right Register Controls:

    signal LR_LOAD:     std_logic;
    signal LR_SHFT_EN:  std_logic;

  -- Key Register Controls:

    signal CD_DIR:      std_logic;
    signal CD_MODE:     std_logic;
    signal CD_SHFT_EN:  std_logic;

  -- Internal Buses:

    signal CD:          std_logic_vector (1 to 56);
    signal KS:          std_logic_vector (1 to 48);
    signal PI:          std_logic_vector (1 to 32);
    signal PO:          std_logic_vector (1 to 32);

    -- Top level visible E Permutation signals:

    signal R1:          std_logic;
    signal R8:          std_logic;      -- also to DOUT(1)
    signal R9:          std_logic;
    signal R16:         std_logic;      -- also to DOUT(3)
    signal R17:         std_logic;
    signal R24:         std_logic;      -- also to DOUT(5)
    signal R25:         std_logic;
    signal R32:         std_logic;      -- also to DOUT(6)

    begin

-- I/O Buffers:

DB1: BIDIR port map (Z => DATA(7),Y => DIN(1), OE => DOE, A => DOUT(1));
DB2: BIDIR port map (Z => DATA(6),Y => DIN(2), OE => DOE, A => DOUT(2));
DB3: BIDIR port map (Z => DATA(5),Y => DIN(3), OE => DOE, A => DOUT(3));
DB4: BIDIR port map (Z => DATA(4),Y => DIN(4), OE => DOE, A => DOUT(4));
DB5: BIDIR port map (Z => DATA(3),Y => DIN(5), OE => DOE, A => DOUT(5));
DB6: BIDIR port map (Z => DATA(2),Y => DIN(6), OE => DOE, A => DOUT(6));
DB7: BIDIR port map (Z => DATA(1),Y => DIN(7), OE => DOE, A => DOUT(7));
DB8: BIDIR port map (Z => DATA(0),Y => DIN(8), OE => DOE, A => DOUT(8));

-- CLOCK BUFFER:

CLKB: CLKBUF port map (A => CLOCK, Z => CLK);

-- RESET BUFFER:

RESETB: INVBUF port map ( A => RESET_N, Z => RESET);

-- Input Buffers:

DI_EN:  INVBUF port map ( A => DIE_N, Z => DIE);

DO_EN:  INVBUF port map ( A => DOE_N, Z => DOE);

KY_EN:  INVBUF port map ( A => KEY_N, Z => KEYE);

ENCODE: INBUF port map ( A => ENCRYPT, Z => ENCR);

-- Output Buffers:

DIRQ:  OUTBUF port map ( A => DIREQ_N, OE => PIN_OE, Z => DI_REQ_N);

DORQ:  OUTBUF port map ( A => DORDY_N, OE => PIN_OE, Z => DO_RDY_N);

KYRQ:  OUTBUF port map ( A => KEYREQ_N, OE => PIN_OE, Z => KEY_REQ_N);
```

```
STATEMACH: STATEM
    port map (
        CLK             => CLK,
        RESET           => RESET,
        DIE             => DIE,
        DOE             => DOE,
        KEYE            => KEYE,
        ENCR            => ENCR,
        DIREQ_N         => DIREQ_N,
        DORDY_N         => DORDY_N,
        KEYREQ_N        => KEYREQ_N,
        LR_LOAD         => LR_LOAD,
        LR_SHFT_EN      => LR_SHFT_EN,
        CD_DIR          => CD_DIR,
        CD_MODE         => CD_MODE,
        CD_SHFT_EN      => CD_SHFT_EN
    );

DSLICE0: DSLICE                 -- 1 byte each of L and R
    port map (
        RI      => DIN(1),
        LI      => DIN(2),
        PI      => PI (1 to 8),
        EL      => R32,
        ER      => R9,
        K       => KS(1 to 12),
        CLK     => CLK,
        RESET   => RESET,
        LD      => LR_LOAD,
        SE      => LR_SHFT_EN,
        PO      => PO (1 to 8),
        RRO     => R8,
        RLO     => R1,
        LO      => DOUT(1)
    );

    DOUT(2) <= R8;

DSLICE1: DSLICE
    port map (
        RI      => DIN(3),
        LI      => DIN(4),
        PI      => PI (9 to 16),
        EL      => R8,
        ER      => R17,
        K       => KS(13 to 24),
        CLK     => CLK,
        RESET   => RESET,
        LD      => LR_LOAD,
        SE      => LR_SHFT_EN,
        PO      => PO (9 to 16),
        RRO     => R16,
        RLO     => R9,
        LO      => DOUT(3)
    );

    DOUT(4) <= R16;

CREG: CD_REG                    -- key for DSLICE0/1
    port map (
        D1              => DIN(1),
        D2              => DIN(2),
        D3              => DIN(3),
        D4              => DIN(4),
        CD_DIR          => CD_DIR,
        CD_MODE         => CD_MODE,
        CD_SHFT_EN      => CD_SHFT_EN,
        CLK             => CLK,
        RESET           => RESET,
        CDOUT           => CD(1 to 28)
    );

PC2C:
    process(CD)                 -- C(9),C(18),C(22), C(25) not used
        begin
        KS(1)  <= CD(14); KS(2)  <= CD(17); KS(3)  <= CD(11); KS(4)  <= CD(24);
        KS(5)  <= CD(1);  KS(6)  <= CD(5);  KS(7)  <= CD(3);  KS(8)  <= CD(28);
        KS(9)  <= CD(15); KS(10) <= CD(6);  KS(11) <= CD(21); KS(12) <= CD(10);
        KS(13) <= CD(23); KS(14) <= CD(19); KS(15) <= CD(12); KS(16) <= CD(4);
        KS(17) <= CD(26); KS(18) <= CD(8);  KS(19) <= CD(16); KS(20) <= CD(7);
        KS(21) <= CD(27); KS(22) <= CD(20); KS(23) <= CD(13); KS(24) <= CD(2);
    end process;

DSLICE2: DSLICE
    port map (
        RI      => DIN(5),
        LI      => DIN(6),
        PI      => PI (17 to 24),
        EL      => R16,
        ER      => R25,
        K       => KS(25 to 36),
        CLK     => CLK,
        RESET   => RESET,
        LD      => LR_LOAD,
        SE      => LR_SHFT_EN,
        PO      => PO (17 to 24),
        RRO     => R24,
        RLO     => R17,
        LO      => DOUT(5)
    );

    DOUT(6) <= R24;

DSLICE3: DSLICE
    port map (
        RI      => DIN(7),
        LI      => DIN(8),
        PI      => PI (25 to 32),
        EL      => R24,
        ER      => R1,
        K       => KS(37 to 48),
        CLK     => CLK,
        RESET   => RESET,
        LD      => LR_LOAD,
        SE      => LR_SHFT_EN,
        PO      => PO (25 to 32),
        RRO     => R32,
        RLO     => R25,
        LO      => DOUT(7)
    );

    DOUT(8) <= R32;

DREG: CD_REG                            -- key for DSLICE2/3
    port map (
        D1              => DIN(7),
        D2              => DIN(6),
        D3              => DIN(5),
        D4              => CD(28),      -- as per PC1
        CD_DIR          => CD_DIR,
        CD_MODE         => CD_MODE,
        CD_SHFT_EN      => CD_SHFT_EN,
        CLK             => CLK,
        RESET           => RESET,
        CDOUT           => CD(29 to 56)
    );

PC2D:
    process(CD)         -- D(7),D(10),D(15),D(26) not used (add 28 to get KS)
        begin
        KS(25) <= CD(41); KS(26) <= CD(52); KS(27) <= CD(31); KS(28) <= CD(37);
        KS(29) <= CD(47); KS(30) <= CD(55); KS(31) <= CD(30); KS(32) <= CD(40);
        KS(33) <= CD(51); KS(34) <= CD(45); KS(35) <= CD(33); KS(36) <= CD(48);
        KS(37) <= CD(44); KS(38) <= CD(49); KS(39) <= CD(39); KS(40) <= CD(56);
        KS(41) <= CD(34); KS(42) <= CD(53); KS(43) <= CD(46); KS(44) <= CD(42);
        KS(45) <= CD(50); KS(46) <= CD(36); KS(47) <= CD(29); KS(48) <= CD(32);
    end process;

P_PERM:                 -- P Permutation for all DSLICEs
    process(PO)
        begin
        PI(1)  <= PO(16); PI(2)  <= PO(7);  PI(3)  <= PO(20); PI(4)  <= PO(21);
        PI(5)  <= PO(29); PI(6)  <= PO(12); PI(7)  <= PO(28); PI(8)  <= PO(17);
        PI(9)  <= PO(1);  PI(10) <= PO(15); PI(11) <= PO(23); PI(12) <= PO(26);
        PI(13) <= PO(5);  PI(14) <= PO(18); PI(15) <= PO(31); PI(16) <= PO(10);
        PI(17) <= PO(2);  PI(18) <= PO(8);  PI(19) <= PO(24); PI(20) <= PO(14);
        PI(21) <= PO(32); PI(22) <= PO(27); PI(23) <= PO(3);  PI(24) <= PO(9);
        PI(25) <= PO(19); PI(26) <= PO(13); PI(27) <= PO(30); PI(28) <= PO(6);
        PI(29) <= PO(22); PI(30) <= PO(11); PI(31) <= PO(4);  PI(32) <= PO(25);
    end process;

end BEHAVE;

configuration BEHAVE_CONFIG of DES is
    for BEHAVE
        for DSLICE0: DSLICE
            for BEHAVE
                for S1: SBOX
                    use entity work.SBOX1(BEHAVE);
                end for;
                for S2: SBOX
                    use entity work.SBOX2(BEHAVE);
                end for;
            end for;
        end for;
        for DSLICE1: DSLICE
            for BEHAVE
                for S1: SBOX
                    use entity work.SBOX3(BEHAVE);
                end for;
                for S2: SBOX
                    use entity work.SBOX4(BEHAVE);
                end for;
            end for;
        end for;
        for DSLICE2: DSLICE
            for BEHAVE
                for S1: SBOX
                    use entity work.SBOX5(BEHAVE);
                end for;
                for S2: SBOX
                    use entity work.SBOX6(BEHAVE);
                end for;
            end for;
        end for;
        for DSLICE3: DSLICE
            for BEHAVE
                for S1: SBOX
                    use entity work.SBOX7(BEHAVE);
                end for;
                for S2: SBOX
                    use entity work.SBOX8(BEHAVE);
                end for;
            end for;
        end for;
    end for;
end BEHAVE_CONFIG;
```