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

entity DSLICE is
    port (
        RI:             in      std_logic;
        LI:             in      std_logic;
        PI:             in      std_logic_vector (1 to 8);
        EL:             in      std_logic;
        ER:             in      std_logic;
        K:              in      std_logic_vector (1 to 12);
        CLK:            in      std_logic;
        RESET:          in      std_logic;
        LD:             in      std_logic;
        SE:             in      std_logic;
        PO:             out     std_logic_vector (1 to 8);
        RRO:            out     std_logic;
        RLO:            out     std_logic;
        LO:             out     std_logic
    );
end ;

architecture BEHAVE of DSLICE is

component REG8S
    port (
        SI:     in      std_logic;
        CLK:    in      std_logic;
        SE:     in      std_logic;
        RESET:  in      std_logic;
        LD:     in      std_logic;
        D:      in      std_logic_vector (1 to 8);
        Q:      out     std_logic_vector (1 to 8);
        SLO:    out     std_logic;
        SRO:    out     std_logic
    );
end component;

component SBOX
    port (
        B:      in      std_logic_vector (1 to 6);
        S:      out     std_logic_vector (1 to 4)
    );
end component;

    signal L:           std_logic_vector (1 to 8);
    signal R:           std_logic_vector (1 to 8);
    signal E:           std_logic_vector (1 to 12);
    signal EXK:         std_logic_vector (1 to 12);
    signal PXL:         std_logic_vector (1 to 8);

    begin
```

```
-- Right and Left register slices

RIGHT: REG8S
    port map (
        SI      => RI,
        CLK     => CLK,
        SE      => SE,
        RESET   => RESET,
        LD      => LD,
        D       => PXL,
        Q       => R,
        SLO     => RLO, -- for E Perm
        SRO     => RRO  -- for E Perm and output
    );

LEFT: REG8S
    port map (
        SI      => LI,
        CLK     => CLK,
        SE      => SE,
        RESET   => RESET,
        LD      => LD,
        D       => R,
        Q       => L,
        SLO     => open,
        SRO     => LO  -- for output
    );
------------
-- f(R,K) --
------------

-- Expansion Permutation

EPERM:
-- DSLICE0      R32,  R1,  R2,  R3,  R4,  R5 (R32,R9  imported)
--       R4,  R5,  R6,  R7,  R8,  R9 (R1,R8  exported)

-- DSLICE1       R8,  R9, R10, R11, R12, R13 (R8,R17  imported)
--      R12, R13, R14, R15, R16, R17 (R9,R16  exported)

-- DSLICE2   R16, R17, R18, R19, R20, R21 (R16,R24 imported)
--      R20, R21, R22, R23, R24, R25 (R17,R24 exported)

-- DSLICE3   R24, R25, R26, R27, R28, R29 (R24,R1  imported)
--      R28, R29, R30, R31, R32,  R1 (R25,R32 exported)

        E <= EL & R(1 to 5) & R(4 to 8) & ER;

-- Pre S

EXORK:
        EXK <= E xor K;         -- 12 XOR gates
```

```
-- S Boxes  (configured uniquely)

S1: SBOX
    port map (
        B       => EXK(1 to 6),
        S       => PO(1 to 4)
    );

S2: SBOX
    port map (
        B       => EXK(7 to 12),
        S       => PO(5 to 8)
    );
-----------------------
-- R <= f(R,K) xor L --
-----------------------

-- P Permutation made by external connectivity

PXORL:
        PXL <= PI xor L;        -- 8 XOR gates

end BEHAVE;
```