# DES MACROBLOCK

## 1. Overview

The DES macroblock implements the Data Encryption Standard first described in FIPS publication 46, 15 January 1977. The same standard is also known under ANSI X3.92. The block operates in Electronic CodeBook (ECB) mode. It accepts 64 bits of input and 56 bits of key, returning 64 bits of output. Both enciphering and deciphering functions are supported. The key and data are written to and read from block using 8 bit bus.

## 2. Architecture

The DES algorithm is implemented sequentially with S-BOXES optimized to gates. The block is synchronous and expects block IO operations to be related to clock signal.

## 3. DES Block input/output ports

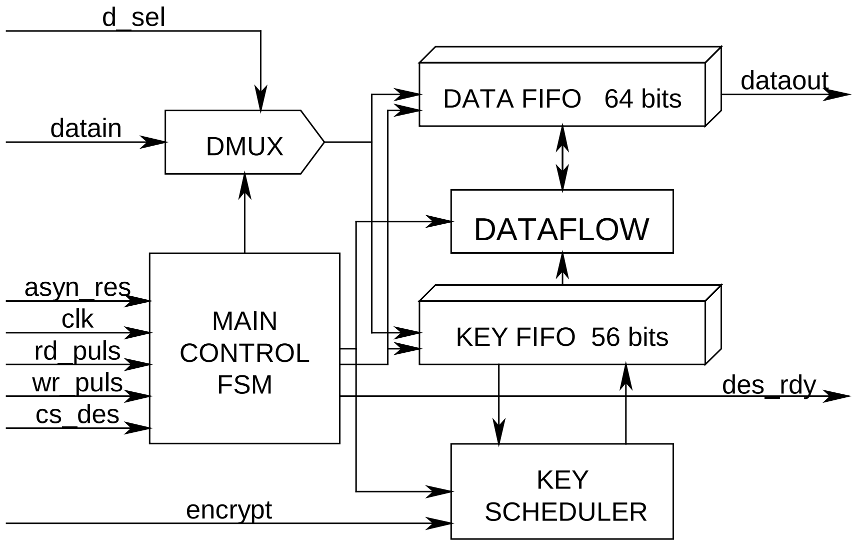
### 3.1 Inputs

| | | |
|---|---|---|
| • Data input bus | DATAIN | 8 |
| • Asynchronous reset | ASYN_RES | 1 |
| • Clock, active positive front | CLK | 1 |
| • DES function select | ENCRYPT | 1 |
| • Read FIFOS | RD_PULS | 1 |
| • Write FIFOS | WR_PULS | 1 |
| • DES block select | CS_DES | 1 |
| • Key or Data FIFO select | D_SEL | 1 |

### 3.2 Outputs

| | | |
|---|---|---|
| • Data output bus | DATAOUT | 8 |
| • DES block ready | DES_RDY | 1 |

## 4. Block operating descriptions



4.0 Before using the block reset line should be held high for at least one clock cycle to initialise internal data and control structures

4.1 Write data to block

| | | |
|---|---|---|
| 4.2.1 Select | DATA FIFO with | *d_sel = 1* or |
| | KEY FIFO | *d_sel = 0* |

4.2.2 Put 8 bits of data to block input, LSB first

4.2.3 Select write to FIFO with setting *WR_PULS=0*

4.2.4 After at least one clock cycle set *WR_PULS=1.* If slow device needs to extend this time then block will wait for WR_PULS to become inactive, writing data to internal FIFO with each rising clock edge during *WR_PULS=0.*

4.2.5 Repeat 4.2.2 to 4.2.4 for remaining bytes

4.3 Read data from block

4.3.1 Select DATA FIFO with *d_sel = 1*

4.3.2 You can now read out the lowest bytes from DES BLOCK

4.3.3 Increment read pointer with setting *RD_PULS=0* . The same holds place with read pulse width as with write pulse.

4.3.4 Set RD_PULS=1 and repeat steps 4.3.2 to 4.3.3 to read out all bytes

## 4.4 DES functions (encryption / decryption)

4.4.1 Fill  DATA FIFO as described earlier.

4.4.2 If you want to use DES     encryption function set  *ENCRYPT* =  *1*
decryption  function set  *ENCRYPT* =  *0*

4.4.3 If you write to KEY FIFO then the last byte written into block will initiate selected DES function.

4.4.1 After 31 Clock cycles *DES_RDY*  should go to active state and you can read out the results from DATA FIFO as described earlier.

Notes.

- High active DES_CS signal enables all block functions.
- Both RD_PULS and WR_PULS signals are checked for being inactive before coming out from IO/DES cycles. This ensures that slower devices can properly communicate with block. The block will insert WAIT cycles if control signals are held active for more than required, what is one clock cycle.
- IO operation takes 2 clock cycles for byte IO
- DES functions take 31 cycles for DES encrypt or  decrypt
- Input signals ( *d_sel, encrypt*  ) are  not latched and should not change state during operation cycle.
- The expected gate count should not exceed 4,5Kgates. The results with mapping to 2 input NOR, Inverter and D type flip-flop gave the following figures: 129 DFF,  2768 NOR2, 1678 INV
- The speed depends on target technology. The design has S-Boxes in different hierarchy, for the  speed of conversion cycle depends solely on these delays. If required, S-Boxes could be flattened to produce faster circuits. As it is now the slowest S-BOX has 7  gate  delays.
- To use the block for 3DES implementations you should use it 3 times with the following  schedule:

  1.1 Fill the DATA FIFO
  1.2 Select the encrypt mode
  1.3 Fill the KEY FIFO with first key. This will run DES 1.
  1.4 Wait for DES_RDY  to become active (31 cycles)
  1.5 select the decrypt mode
  1.6 Fill the KEY FIFO with second key. This  will run  DES  2.
  1.7 Wait for DES_RDY to become active (31 cycles)
  1.8 select the encrypt mode
  1.9 Fill the KEY FIFO with third key. This will run DES 3.
  1.10 Wait for DES_RDY to become active (31 cycles)
  1.11 read out DATA FIFO

## 5. DES MACROBLOCK validation

### 5.1 General

DES block is verified using testbench file. It exercises the block with test vectors for DES Electronic Code Book (ECB) implementation, taken from: "Validating the Correctness of Hardware Implementations of the NBS Data Encryption Standard" , NBS Special Publication 500-20, 1980.
The vectors will test

- Initial Permutation and Expansion
- Inverse Permutation and Expansion
- Key Permutation
- Right-shifts in Decryption
- Data permutation
- S-Boxes

The test vectors are in file des.test. This original test vector file contains entries like the following:

```
encrypt
#
 0101010101010101 95F8A5E5DD31D900 8000000000000000
```

Here the first entry is the key, the second the plaintext ant the third the expected result. The function selector `encrypt,decrypt` show DES function type. The key is in standard format, where low bits of each key byte represent data for parity check.
First this file is edited to gather all decryptions together. The resulting file des.test_decrypt_0to56_encrypt_the_rest contains first 57 decryptions and the remaining encryptions. All the comments are stripped off. The resulting file contains only data and the first line is:

```
8001010101010101 95A8D72813DAA94D 0000000000000000
```

This data has to be converted for use with DES block what has 8 bits IO bus. For that purposes use PERL script FORMAT_DESD what creates file DATA.in for testbench. The start of DATA.in is(\n is enter as usually):

```
00\n00\n00\n00\n00\n00\n80
4D\nA9\nDA\n13\n28\nD7\nA8\n95
00\n00\n00\n00\n00\n00\n00\n00
```

The plaintext and ciphertext are the same, the key has parity data thrown away. They are in file LSB first.

## 5.2 TESTBENCH is in file DESTEST.vhd

- **the libraries used**

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_misc.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_textio.all;
use std.textio.all;
entity TESTER is
-- test bench, no io
end TESTER;
architecture AA of TESTER is
```

- **DES block component**

```
component DESBLOCK
port (        ASYN_RES: in    std_logic;    -- ASYNC RESET
                   CLK: in    std_logic;    -- CLOCK, +edge
               ENCRYPT: in    std_logic;    -- ENCRYPT=1, DECRYPT=0
               RD_PULS: in    std_logic;    -- read  fifo=0
               WR_PULS: in    std_logic;    -- write fifo
                CS_DES: in    std_logic;    -- chip select
                 D_SEL: in    std_logic;    -- 1=data, 0=key
                DATAIN: in    std_logic_vector(7 downto 0);  -- data in
               DES_RDY: out   std_logic;    -- DES   READY
               DATAOUT: out   std_logic_vector(7 downto 0)   -- data out
     );

end component;
signal    DATAIN: std_logic_vector( 7 downto 0); -- data to des block
signal   DATAOUT: std_logic_vector( 7 downto 0); -- data from des block
signal   RD_PULS: std_logic;               -- fifo read  pulse
signal   WR_PULS: std_logic;               -- fifo write pulse
signal    CS_DES: std_logic;                     -- des block select
signal   ENCRYPT: std_logic;               -- mode 0 - decrypt, 1-
encrypt
signal     CLOCK: std_logic;                 -- clock input to system
signal     D_SEL: std_logic;               -- DATA/KEY FIFO select
signal     RESET: std_logic;               -- main ASYNC reset
signal   DES_RDY: std_logic;               -- Block ready
-- FILE IO temp variables
signal    FILE_IO: std_logic;              -- rising front active  IO
signal     DATAEN: std_logic;              -- FILE IO direction
-- TIMING CONSTANTS
CONSTANT      CLKCY: time := 50 NS;        -- CLOCK CYCLE LENGTH
CONSTANT HOLD_DELAY: time := 3  NS;        -- CLOCK Propagation + lib
hold
begin

DUT:DESBLOCK port map
(
      ASYN_RES => RESET,
          CLK  => CLOCK,
       ENCRYPT => ENCRYPT,
       RD_PULS => RD_PULS,
       WR_PULS => WR_PULS,
        CS_DES => CS_DES,
         D_SEL => D_SEL,
        DATAIN => DATAIN,
       DES_RDY => DES_RDY,
       DATAOUT => DATAOUT
);
```

- **This process interacts with disk file. The byte is read upon rising edge of FILE_IO from file DATA.in**

```
FILEIO: process
file  OUTWR: text is out "DATA.out";
file  INRD:  text is IN  "DATA.in";
variable ILINE, OLINE: line;
variable DAT_TMP_IN, DAT_TMP_OUT: std_logic_vector(7 downto 0);
begin
wait until FILE_IO'EVENT and FILE_IO='1' ;
-- Reading from file
if (DATAEN='0'          and
    RESET='1'        and
    (not  endfile(INRD))   )  then
readline(INRD,ILINE);
hread(ILINE, DAT_TMP_IN);
DATAIN<=DAT_TMP_IN;
end if;
-- Writing to file
if (DATAEN='1'          and
    RESET='1'           )  then
DAT_TMP_OUT:=DATAOUT;
hwrite(OLINE, DAT_TMP_OUT);
writeline(OUTWR,OLINE);
end if;
end process;
```

- **Clock creation**

```
-- Main system clock
KELL: process
begin
wait for CLKCY/2 ;
if( RESET = '0') then
CLOCK<='0';
else
CLOCK<= not CLOCK;
end if;
end process;
-- Test stimuli process
```

- **This process exercises the DES block**

```
TEST: process
-- read the testdata from file to compare
variable  FILEDATA: std_logic_vector( 63 downto 0);
-- des cycle data output
variable CHECKDATA: std_logic_vector( 63 downto 0);
-- error
variable      VIGA: std_logic;
-- conversions  counter
variable   COUNTER: integer range 0 to 1000;
begin
```

- **Initiate control for  File IO**

```
FILE_IO <= '0'; -- file IO will take place on rising edge of this signal
DATAEN  <= '0'; -- file IO direction, here only read
```

- **Reset the DES block**

```
RESET<='0';       -- main asynchronous reset
RD_PULS<='1';     -- read  pulse
WR_PULS<='1';     -- write pulse
CS_DES<='0';      -- select des, active hi
D_SEL<='1';       -- begin with data fifo select
VIGA:='0';        -- no error to begin with
COUNTER:=0;       -- conversion counter
wait for  CLKCY;  -- reset delay
```

- **Start with the main test loop**

```
RESET<='1';
while (VIGA='0') loop  -- all entries from file until error
                   -- last entry generates error by itself
```

- **Test cycle Initialisation**

```
-- In test table first entries are decrypt, then encrypt
if( COUNTER>57 ) then ENCRYPT<='1';
                 else ENCRYPT<='0';
                 end if;

CS_DES<='1';      -- select block
FILE_IO <= '0';  -- file IO on positive front
RD_PULS<='1';    -- FIFO read  pulse
WR_PULS<='1';    -- FIFO write pulse
D_SEL<='1';      -- DATA fifo select
wait until  CLOCK'EVENT and CLOCK='1'; -- to sync
wait for HOLD_DELAY;                   -- to prevent signals from
                                       -- changing at the same time
                                       -- as clock
```

- **Start with DATA input**

```
for I in 1 to 8 loop
FILE_IO<='1';   -- get the  data from file
wait until CLOCK'EVENT and CLOCK='1';
wait for HOLD_DELAY;
WR_PULS<='0';
wait until CLOCK'EVENT and CLOCK='1';
wait for HOLD_DELAY;
WR_PULS<='1';
FILE_IO<='0';
wait until CLOCK'EVENT and CLOCK='1';
wait for HOLD_DELAY;
end loop;

D_SEL<='0';    -- set input to keys
```

- **Then input keys**

```
for I in 1 to 7 loop
FILE_IO<='1';    -- get it from file
wait until CLOCK'EVENT and CLOCK='1';
wait for HOLD_DELAY;
WR_PULS<='0';
wait until CLOCK'EVENT and CLOCK='1';
wait for HOLD_DELAY;
WR_PULS<='1';
FILE_IO<='0';
wait until CLOCK'EVENT and CLOCK='1';
wait for HOLD_DELAY;
 end loop;
```

- **The last write has initiated conversion cycle, wait for block to finish**

```
wait until DES_RDY'EVENT and DES_RDY = '1';
```

- **Prepare to read in the comparison data from file**

```
CS_DES<='0';     -- disable des block;
wait until CLOCK'EVENT and CLOCK='1';
wait for HOLD_DELAY;
```

```
for I in 0 to 7  loop
WR_PULS<='0';    -- to test CS_DES functionality
RD_PULS<='1';    -- the same
FILE_IO<='1';
wait until CLOCK'EVENT and CLOCK='1';
wait for HOLD_DELAY;
RD_PULS<='0';    -- it too
WR_PULS<='1';    -- this also
FILE_IO<='0';
FILEDATA( ((I+1)*8)-1 downto I*8 ):=DATAIN;
wait until CLOCK'EVENT and CLOCK='1';
wait for HOLD_DELAY;
end loop;
```

• **now read out the data from DES block**

```
-- INITIALIZE for DATA readout
WR_PULS<='1';
RD_PULS<='1';
CS_DES<='1';
D_SEL<='1';
-- READ OUT 8 DATA BYTES
for I in 0 to 7  loop
wait until CLOCK'EVENT and CLOCK='1';
wait for HOLD_DELAY;
RD_PULS<='0';     -- pulse aadress
wait until CLOCK'EVENT and CLOCK='1';
wait for HOLD_DELAY;
RD_PULS<='1';
CHECKDATA(((I+1)*8)-1 downto I*8):=DATAOUT;
end loop;
```

• **and  compare it with testdata from file**

```
if(CHECKDATA=FILEDATA) then
COUNTER:=COUNTER + 1;
elsif(COUNTER < 290) then
assert FALSE report "ERROR in COMPARE" severity ERROR;
else
assert FALSE report "END OF FILE, ALL OK" severity ERROR;
end if;
end loop;
end process;
end AA;

configuration TB of TESTER is
for AA
for DUT:DESBLOCK use entity WORK.DESBLOCK(VER2_1);
end for;
end for;
end TB;
```

For any additional comments contact

Juri Poldre
jp@pld.ttu.ee
*PHONE/FAX +372-6202253*