# VHDL Based Implementation of SDES – Simplified Data Encryption and Decryption

This repository contains a complete VHDL implementation of the \*\*Simplified Data Encryption Standard (SDES)\*\*, a lightweight encryption algorithm often used for educational purposes in cryptography and digital design.

## ?? Project Description

- \*\*Language:\*\* VHDL

- \*\*Tool:\*\* Xilinx Vivado

- \*\*Type:\*\* Behavioral Modeling

- \*\*Modules Implemented:\*\*

- Key Generation (Key Scheduling)

- SDES Encryption Unit

- SDES Decryption Unit

- \*\*Simulation:\*\* Performed using Vivado Simulator with functional testbenches

This project showcases how a symmetric encryption algorithm can be realized in hardware using combinational and sequential logic in VHDL.

## ?? Objective

To design and simulate a hardware-level encryption and decryption system using SDES. This project demonstrates:

- Bit-level data manipulation

- Use of S-boxes and permutation functions

- Finite-state based functional decomposition

## ?? Functional Modules

| Module | Description |

|-----------------|-------------------------------------------------|

| `key\_sdes.vhd` | Generates two 8-bit subkeys from 10-bit main key|

| `encrypter\_sdes.vhd` | Encrypts an 8-bit plaintext using SDES |

| `decrypter\_sdes.vhd` | Decrypts an 8-bit ciphertext |

| `testbench.vhd` | Simulates the encryption-decryption loop |

## ?? Tools & Technologies

- ? \*\*Vivado Design Suite\*\* by Xilinx

- ? \*\*VHDL (IEEE 1164, NUMERIC\_STD libraries)\*\*

- ? Behavioral simulation for verification

## ?? Project Flow

A diagram of a process

AI-generated content may be incorrect.

##

Example :-

+-------------------------+

| Start Program |

+-----------+-------------+

|

v

+-------------------------------+

| Input: 10-bit Key = 1010000010 |

+-------------------------------+

|

v

+-----------------------+

| Key Generation |

| Key1 = 10100100 |

| Key2 = 01000011 |

+-----------+-----------+

|

+--------------------+----------------------+

| |

v v

+---------------------+ +----------------------+

| Plaintext Input | | Ciphertext Input |

| = 11010111 | | = 01101111 |

+----------+----------+ +----------+-----------+

| |

v v

+---------------------------+ +----------------------------+

| Initial Permutation (IP) | | Initial Permutation (IP) |

+-----------+---------------+ +-------------+--------------+

| |

v v

+---------------------------+ +-----------------------------+

| Round 1 with Key2 = 01000011 | | Round 1 with Key1 = 10100100 |

| - Expand & Permute (E/P) | | - Expand & Permute (E/P) |

| - XOR with Key | | - XOR with Key |

| - S-boxes S0 and S1 | | - S-boxes S0 and S1 |

| - P4 Permutation | | - P4 Permutation |

| - XOR with Left 4 bits | | - XOR with Left 4 bits |

+-----------+---------------+ +-------------+--------------+

| |

v v

+----------------------+ +----------------------+

| SWAP Halves | | SWAP Halves |

+----------+-----------+ +----------+-----------+

| |

v v

+---------------------------+ +-----------------------------+

| Round 2 with Key1 = 10100100| | Round 2 with Key2 = 01000011 |

| - Expand & Permute (E/P) | | - Expand & Permute (E/P) |

| - XOR with Key | | - XOR with Key |

| - S-boxes S0 and S1 | | - S-boxes S0 and S1 |

| - P4 Permutation | | - P4 Permutation |

| - XOR with Left 4 bits | | - XOR with Left 4 bits |

+-----------+---------------+ +-------------+--------------+

| |

v v

+---------------------------+ +---------------------------+

| Final Permutation (IP⁻¹) | | Final Permutation (IP⁻¹) |

+-----------+---------------+ +------------+--------------+

| |

v v

+---------------------------+ +---------------------------+

| Output: Ciphertext = | | Output: Recovered |

| 01101111 | | Plaintext = |

| | | 11010111 |

+---------------------------+ +---------------------------+

## ?? How to Run

1. Clone the repo:

```bash

git clone https://github.com/your-username/sdes-vhdl.git

```

2. Open Vivado and create a new project.

3. Add the `.vhd` source files under the `src/` folder.

4. Add `testbench.vhd` under simulation sources.

5. Run simulation to verify correctness.

## ?? License

This project is open source and available under the MIT License.