SIMULATION USER GUIDE

Compact 128 bit blockcypher CLEFIA Implementation

Type-III Compact CLEFIA Implementation on FPGA

GNU LGPL Licence

This file is part of CLEFIA Type-III FPGA IP-core.

CLEFIA Type-III is free FPGA IP-core: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

CLEFIA Type-III is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with CLEFIA Type-III.

If not, see <http://www.gnu.org/licenses/>.

1. Overview

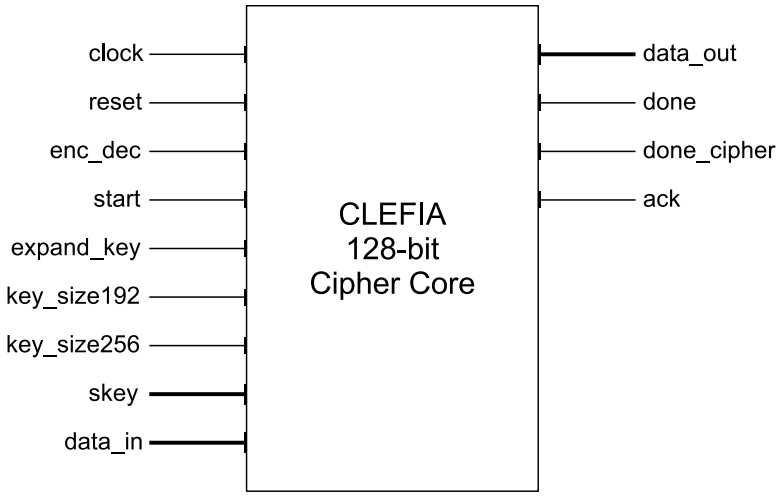
CLEFIA encryption algorithm is a novel symmetrical block ciphering algorithm proposed and developed by SONY Corporation focused on Digital Rights Management (DRM) purposes. This algorithm improves the security of encryption with the use of techniques such as Diffusion Switch Mechanisms, consisting of multiple diffusion matrices in a predetermined order, to ensure immunity against differential and linear attacks, and the use of Whitening Keys, combining data with portions of the key before the first round and after the last round.

The CLEFIA algorithm works with a key size of 128, 192, or 256 bits. It consists of a key scheduling phase and an input data block transformation phase computed over multiple rounds, employing a relatively homogeneous algorithm. The data path of CLEFIA Type-III is composed of a two 4-branch Feistel structure computed for several rounds, defined as $GFN_{8,n}$. However, this architecture places the two structures into the same design through pipeline and proper data scheduling.

The following architecture explores the resources provided by FPGA devices to provide a compact architecture capable of perform all CLEFIA modes of operation. This design it uses FPGA embbeded elements such as RAM memory and shift registers. The cypher block is designed in such way that it is capable of manage a 256 bit input block. Although CLEFIA cypher block is of 128 bit, this design is necessary in order to provide the data needed for key expansion and Round Key generation.

This document describes the CLEFIA Blockcypher simulation procedures. The following sections presents a brief overview of the CLEFIA Type-III design followed by the design specification and simulation characteristics.

1.1. Block Diagram



1.2. Signal Descriptions

Name	Length	Direction	Description
clock	1	Input	system main clock.
reset	1	Input	system main reset signal.
enc_dec	1	Input	enable encoding (1) and decoding (0) process.
start	1	Input	process start of operation signal.
expand_ key	1	Input	control system expansion key operation (1 = enable, 0 = disable).
key_ size192	1	Input	use a 192 bit key size.
key_ size256	1	Input	use a 256 bit key size.
skey	256	Input	key used for expansion nad whitening.
data_ in	128	Input	input block to be cyphered.
data_ out	128	Output	cyphered block.
done	1	Output	done key expansion signal.
done_cipher	1	Output	done cipher core signal.
ack	1	Output	indicates the pipeline fulfill.

1.3. Basic Operation

The CLEFIA blockcypher has two operation modes. The fist one is the ciphering operation which is placed into the three CLEFIA modes, based on keys sizes. The main difference between them is the number of rounds (also the number of round keys). The second is the key expansion process, in witch a provided key is placed into the data input block to generate a L key. This key is further used in Key Expansion, in order to generate the Round Keys. The round keys, as well as the constants, are stored in a FPGA memory block, decribed in Section 2.

In CLEFIA 128 bit key expansion the structure acts as a $GFN_{4,n}$ function architecture. For 192 and 256-bit key expansion the system works as a $GFN_{8,n}$ function capable of dealing with a 256-bit input block (the shared key). Each F-Function has a 32-bit input/output datapath. The different input key sizes that can be used in CLEFIA (128, 192, or 256 bits) has directly influence on the number of computed rounds, 18, 22, or 26, respectively.

2. Memory Specification

In order to proper simulate the design must have the specific device memories IP-cores. This section present them and also how they are specified.

2.1. Ciphering T-Box

T-Boxes merge the computation of the Substitution Boxes (S_0 and S_1) operations with the linear transformation layers, compressing the resulting structure into a lookup table, also resulting on a reduction of the critical path. In the CLEFIAs F-Functions operation, T-Boxes can be used to replace S_0, S_1, M_0 and M_1, by the lookup operations followed by XOR operations ($GF(2^8)$ additions).

For simulate the design one need to have the respective memory initialization files. The present memory files was designed for Xilinx Block ROM and its location is described in Section 3. The T-Boxes are also has registed input/output ports. The memory must have the following specification:

- Different clock input ports;
- Read Width: 32;
- Read Depth: 512;
- Read always enabled;
- Register port A and B of Memory Core;

Instance name: `tbox_0_0, tbox_1_0`.

2.2. Key Expansion

The Key Expancion memory block has acts as RAM and ROM. The RAM porch has 64 read depth reserved to store the generated Round Keys for the ciphering mode. The other sections are alocated for the constants and previously defined round keys used in key expansion procedure. This memory block must have the following specification:

- Different clock input ports;
- Read Width: 32;
- Read Depth: 384;
- Enable pin for port A and B;
- Write first policy;
- Register port B output of Memory Core;

Instance name: `KeyMemory`.

3. Directory Structure

The design has the following directory structure.

```
\doc -> full documentation repository
\rtl -> has the RTL VHDL files.
\sim -> has the simulation files
   \tb -> stores the design testbenches
   \xilinx-memory -> memory data files
\sys -> reference model and debug files
```


4. Testbench Description

4.1. Test Vectors

Name	Length	Description
plain_text	128	000102030405060708090a0b0c0d0e0f
shared_key	256	ffeeddccbbaa99887766554433221100f0e0d0c0b0a09080706050403020100

4.2. Basic Operation

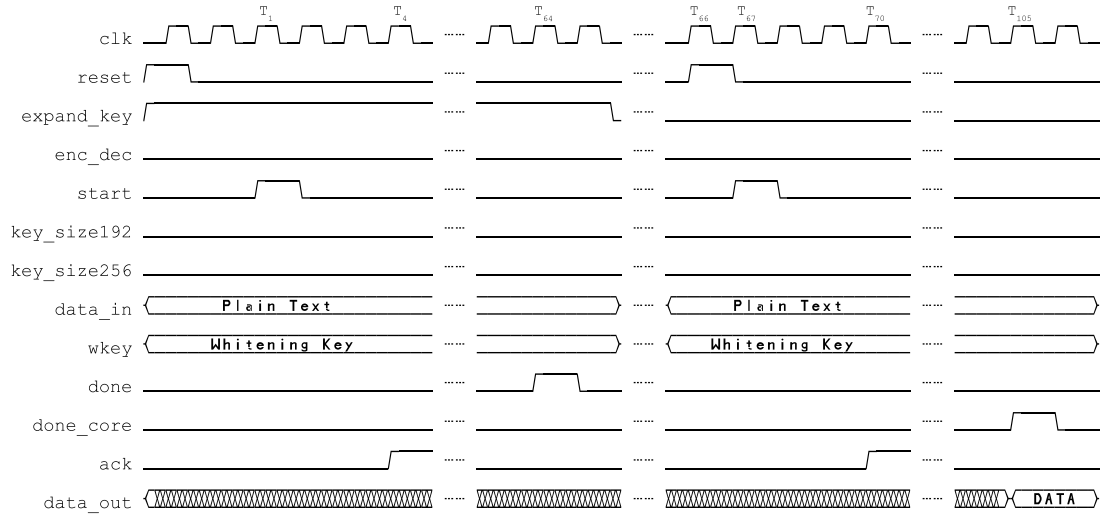
The first step is to configure the simulation. The proper solution is based in signal modification before simulation starts. The configuration signals are described in the following list.

- **Input Data** [data_in]: input data to be ciphered (in key expansion it is dismissed);
- **Whitening Key** [wk]: shared key used for key expansion and ciphering;
- **Key Size for 192-bit** [key_size_192]: defines the key size to 192-bit;
- **Key Size for 256-bit** [key_size_256]: defines the key size to 256-bit;
- **Encrypt Signal** [encrypt_decrypt]: defines if cyphering of de-cyphering mode is activated;
- **Expand Key** [expand_key]: enables key expansion module;

Simulation starts with the key expansion block simulation. In this phase, the round keys are generated accordingly to the ciphering key size. For a proper function the `expand_key` signal must be set to high. The following step is to enable the ciphering procedure. In this region, the `expand_key` signal must be set to low and it is necessary to define the key size. For a 128-bit key size one must place `key_size_192` and `key_size_256` signals to low.

5. Timing Example

The following timing diagram present an example of cyphering process using a 256 bit key size. It also considers the key exansion realization. The system operates in single clock and synchronous reset. All operations are activated during clock rising edge. This simulation is based on the main encoding testbench (`core_enc_tb`).



Time T_1

In this time the control must provide a signal to start the key-expansion processing. The following process encapsulates both the L-key generation and the Round Key computing.

Time T_4

At this time, the system has already registered all input words block. The ack signal means that the loop process is about to get started and the cyphering process is already running.

Time T_{64}

After the entire set of rounds keys had been generated, the process is completed and the done signal is set to high.

Time T_{66}

It is recommended to reset the design before start the cyphering process. This signal is used to assure the proper state machines setup.

Time T_{67}

In this time the control must provide a signal to start the ciphering processing.

Time T_{70}

At this time, the system has already registered all input words block. The ack signal means that the loop process is about to get started and the cyphering process is already running.

Time T_{105}

After the ciphering process is completed the $done_core$ signal is set to high. The ciphered key is valid only at this time.

6. Synthesis Informations

To synthetize the entire design it was described a input control module in order to solve the input port sizes. In order to run the project synthesis, one must create a new FPGA design project, or use one of our examples. The top level module is called $regcore$ ans it is located into the rtl design folder.

To provide the realistic scenario of synthesis report, one must also synthetize the $regcore$ component to check the chip area used accordingly to the selected chip.

Revision History

Date	Description	Owner
06/03/2014	Document conception	João Carlos
08/03/2014	• Improve timing description; • Contextual review;	João Carlos
16/03/2014	Memory block descriptions	João Carlos