TEA Encryption Logic Design Requirements Document

Author: ECE4170
Date: TBD

Objectives and Scope

This document provides the requirements for an intellectual property (IP) core for encryption. The intended audience includes the product marketing engineer and engineers designing components that would interface to this IP core. This document lists the input, output and functional requirements of the encryption logic. Design details and constraints will be described in a separate functional design document.

Definitions, Acronyms, and Abbreviations

TBD – To be Determined
Kbps – Kilobits per second
TEA – Tiny Encryption Algorithm

References

[1] http://users.ece.gatech.edu/~sudha/4170/Spring2006
[2] The TEA Encryption Algorithm, International Conference on Encryption, 2003.

Functional Requirements

The functional requirements are as follows:

1. The encryption logic will accept incoming words and encrypt the words using the TEA algorithm [2].
2. Each word is of length 32-bits.
3. The design shall support input and output bit rates of 32Kbps and 64Kbps
4. The input and output bit rates must be the same.
5. There is no requirement on the encryption latency.
6. The maximum power consumed by the design must be 1W

One might in principle also have additional physical requirements such as

1. The design must fit within an area of 1 mm^2
2. The design must fit on a medium size Xilinx Spartan-3 FPGA
3. The number of I/Os must not exceed 240 signals.

Input Requirements

1. Words of length 32-bits
2. Input bit-rate indicator (32Kbps or 64Kbps)
3. Control input enabling/disabling the encryption logic

Output Requirements

1. An encrypted output of length 32-bits
2. Flag indicating that the output is valid
3. Output bit-rate indicator (32Kbps or 64Kbps)

Integration Requirements

1. Identify any integration requirements; for example,
    a. must integrate with the Wishbone bus
    b. internal control registers must be read/written by an external processor
    c. must be OCP compliant
    d. documentation must conform to SPIRIT standards

Test & Debug Requirements

1. Identify testing requirements.Thinking through testing can save considerable hacking later to be able to test the design as well as iron out any subtle aspects of your design. For example,
    a. What are the set of input values that must be tested? Do you really need to test all possible combinations?
    b. What internal components do you need to test?
    c. How will you generate the test inputs?
        i. Example: generate them off-line and use file I/O + testbench
        ii. Example: generate them within a testbench, e.g., periodic signals with specific timing requirements generated in a custom process
    d. How will generate "golden outputs" – known correct values for the outputs. Most often a C/C++ implementation will be used as the baseline to generate correct results and you must factor this into your timeline.

Metrics and Evaluation

1. Define the metrics you will use to evaluate your design
    a. Example: speedup over an embedded processor in which case how are you going to measure the execution time on the embedded processor? You will have to get access to an open source simulator
    b. Area: compare the area of your design with that of published values for, say, the same encryption core as published by Xilinx or Altera

---