CS 714 – Real Time Computing Systems

Project proposal

Analysis of Timing Variability for Encryption Algorithms for the Atmega platform

Dinesh Dasarathan
ddasara@unity.ncsu.edu

Abstract

Security in Embedded systems is emerging as a growing issue in recent times. Limitations in processing power and communications bandwidth and the mismatch between wide arithmetic for security(32 bit word operations) and embedded data bus widths(8 or 16 bits) offer a great challenge in implementing encryption algorithms for embedded systems.

The variability of execution times is of particular interest to Real Time systems. Variations of execution times in small embedded systems may be due to the software mechanisms used by the compiler to implement certain features that are not present, such as caches that are prevalent in a normal processor. The results that come out of the variability analysis can be valuable in the analysis of worst-case execution times suited for schedulability analysis in the context of real-time systems.

Problem Statement

The analysis of how execution times for different cryptographic algorithms vary in the Atmega 128 processor forms the basis of this project. The results could be useful in the analysis of worst-case execution times that form an important discussion in determining the applicability of real-time systems.

Problem Details

The cryptographic algorithms used are IDEA, RC4 and RC5. RC4 is a stream cipher symmetric key algorithm. This algorithm is quite simple and operations involve the addition of 8 bit elements or swapping variables in a 256-byte state table. IDEA (International Data Encryption Algorithm) is a symmetric-key block cipher that operates on 64 bit plaintext blocks. The algorithm primarily includes operations from three algebraic groups: XOR, addition modulo 216, multiplication modulo 216+1. RC5 is a fast symmetric block cipher with a variety of parameters: block size, key size and number of rounds. It uses the XOR, addition and rotation operations.

The Atmega128 is at the high end of the AVR family's performance spectrum. It is a RISC architecture featuring 8 bit native word size, 32 general purpose registers and support for 16 bit operations. It features a 2 stage pipeline and has a 2-cycle multiply instruction.

Approaching the solution

The entire project is to be done on Linux. Tools to compile programs to hex format and to upload them into the Atmega are available. The AVR-GCC compiles the C program into Intel hex format that can be executed by the Atmega processor. Another tool UISP is used to upload this hex file into the Atmega128 and the results are transmitted through the UART on the stk500 board that houses the Atmega processor. These results are received by minicom, a hyperterminal in Linux that reads the serial port. All programming and data input output is done via the serial port.

References

[1] "Encryption Overhead for Sensor Networks and Embedded Systems: Modeling and Analysis" by Ramnath Venugopalan, Prasanth Ganesan, Pushkin Peddabachagari, Alexander Dean, Frank Mueller and Mihail Sichitiu in Conference on Compiler, Architecture and Synthesis on Embedded Systems (CASES'03), Oct/Nov 2003, pages 188-197

[2] "Analyzing and Modeling Encryption Overhead for Sensor Network Nodes" by Prasanth Ganesan, Ramnath Venugopalan, Pushkin Peddabachagari, Alexander Dean, Frank Mueller and Mihail Sichitiu in Workshop on Wireless Sensor Networks and Applications (WSNA '03) with MobiCom'03, Sep 2003, pages (accepted).

[3] R. Rivest, “The RC5 encryption algorithm,” in Proc. of the 1994 Leuven Workshop on Fast Software Encryption. Springer-Verlag, 1995, pp. 86-96. [Online]. Available: http://citeseer.nj.nec.com/rivest95rc.html

[4] J.Burke, J.McDonald, and T.Austin, “Architectural support for fast symmetric-key cryptography,” in Proc. of ASPLOS-IX, 2000, pp. 178-189.