Non-Pipelined Versus Pipelined Implementation of Encryption Standards in VHDL

Preston Maness
Texas State University at San Marcos
pmm50@txstate.edu

***Abstract*—In this term paper for EE4321 VHDL with Dr. Salamy, an investigation is made into the advantages of pipelined architectures/dataflows through the lense of common encryption algorithms. Two reference papers are investigated and analyzed, one implementing the outdated DES and the other implementing its successor, AES.**

CONTENTS

I. PAPER IMPORTANCE AND RELATED WORK

THE importance of encryption algorithms has grown over time. Traditionally, these algorithms were implemented in software and bottlenecked on the CPU. Thus hardware implementations were born and baked into modern CPUs and programmed onto FPGAs. While this was an improvement the throughput of these designs still left much to be desired.

These two reference papers present methods of pipelining two common encryption algorithms. Pipelining permits the output of final data on every clock cycle once the pipeline is filled. Given the relative simplicity of these algorithms in comparison to, say, pipelined ISAs, the hazards normally encountered when pipelining are largely mitigated. Cheap hardware is capable of working with encrypted data at throughputs and latencies comparable to non-encrypted data.

The first reference paper is by Taherkhani, et. al.[1] and covers DES. The second reference paper is by Gomes et. al.[2] and covers DES' successor, AES. Both investigate improvements of pipelined implementations over non-pipelined implementations.

II. INTRODUCTION TO CONCEPTS

BOTH DES and AES are known as symmetric encryption schemes. A single shared secret –called the private key– exists between two parties, both of which will use the key for both encryption and decryption of messages. This is in contrast to public key cryptography, in which both parties have a pair of distinct public and private keys that are related in such a way that one may encrypt a message against the public key of another while only the holder of the private key may decrypt the message.

The trade-off is apparent: symmetric key cryptography on its own has a single point of failure in the private key. However, public key cryptography requires substantially larger key-lengths to remain robust against brute-force attack in comparison to symmetric algorithms. For this reason symmetric cryptography is often used –especially when the exposure of the shared secret is difficult and apparent when it has occured, as in tamper-resistant and evident hardware devices– since the algorithms require less total effort. (An aside: often, the two are combined, with initial public key cryptography facilitating the exchange of shared "ephermeral" symmetric keys that are then used for subsequent communication; this is how SSL works).

A. DES Algorithm

Part of the elegance of the DES algorithm is that it makes use of only bit shifting and substitution. Each of these is simple to understand and implement. This makes DES an ideal introduction to cryptographic algorithms and their hardware implementations. It should be noted, however, that DES has effectively been deprecated in favor of its successor, AES.

The algorithm is detailed in Figure 1. In effect, the key defines the type of shifting and substitution to make in each pass. In each of the 16 passes, a subkey is generated through binary rotation and permutation, and then that subkey is combined with the right half of the input data with a cipher function, before being XORed with the left half of the input data. At that point, the next round begins.

The form of mixing that is employed is non-linear, making it impossible to "work backwards" without knowledge of the private key. The shifting, permutation, and cipher function are further described... TBD TBD TBD TBD

B. AES Algorithm

AES is substantially more complicated. No joke. Just check out this comic strip explaining it:

- http://www.moserware.com/2009/09/stick-figure-guide-to-advanced.html

Yeah. It's bananas. But it works. Ok, it's not THAT bad, but it IS sensitive to multiple side-channel attacks: malevolent manipulation of CPU cache, and timing-based attacks are two examples. There are effective counter- measures to them, but they're tricky.

You can get a general picture of the algorithm by looking at Figure 2. Each of the blocks –SubBytes, ShiftRows, MixColumns, and AddRoundKey– require their own explanation that may be found in later figures. TBD TBD TBD.

III. Summary and Approach

AFTER covering the basics of the algorithm, each paper then describes how their particular FPGA can implement state machines, and how this affects the design choices they made.

IV. Major Contributions

A Simple introduction to pipelining with practical applications.

V. Analysis and Improvements

YOU know, without their source code, I can't make any meaningful analysis or improvements.

VI. Shortcomings

WHERE is the source code? It isn't in the paper itself and isn't referenced to exist anywhere else! A block diagram is nice, but please: source code so we can replicate your work! This goes for BOTH papers. Neither has source code or datasets (at least that I can find).

This is what I'm talking about:

> Various implementations are presented in various platforms for DES algorithm in [1], [3], [4], [9], [10]. However the design provided in this study is implemented in Virtex6 FPGA. In the current pipelined design, we put buffers in input and output stages of each round. The key scheduler part does quite a novel task in current implementation. It floods the sub keys to appropriate round. The key scheduler implementation specifications are as follow:

If I can't replicate your work, then it's not science, and I can't trust any results you give me.

VII. Conclusion

GOOD god. Please stop screen-shotting non-free software waveform viewers. Stop screen-shotting period! And please, PLEASE make the source code and raw data available and easy to find!

Seriously.

References

[1] S. Taherkhani, E. Ever, and O. Gemikonakli, "Implementation of non-pipelined and pipelined data encryption standard (des) using xilinx virtex-6 fpga technology," in *Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on*, June 2010, pp. 1257–1262.

[2] O. Gomes, R. Moreno, and T. Pimenta, "A fast cryptography pipelined hardware developed in fpga with vhdl," in *Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), 2011 3rd International Congress on*, Oct 2011, pp. 1–6.

LIST OF FIGURES

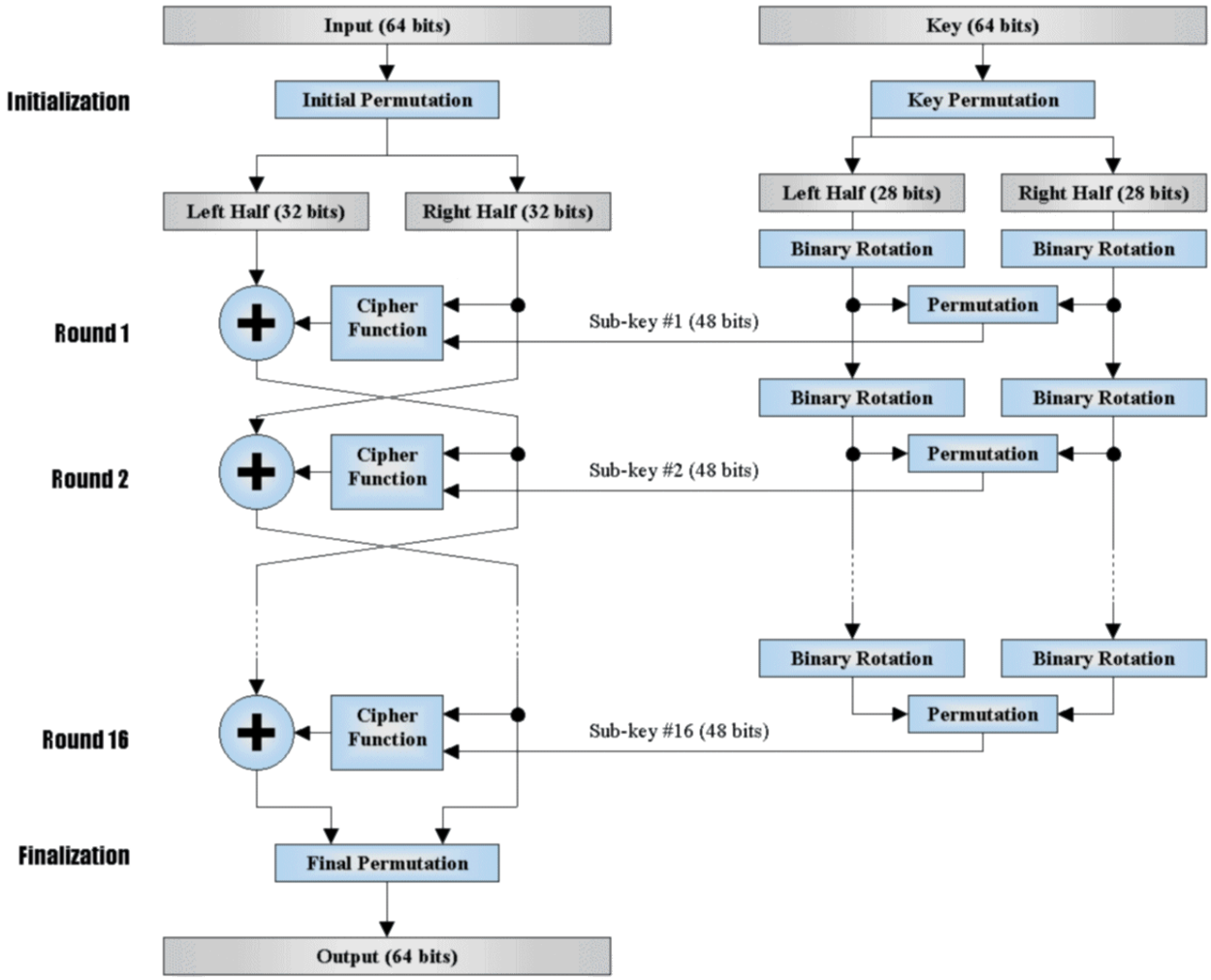

Fig. 1: The DES Algorithm. The plus sign indicates an XOR.

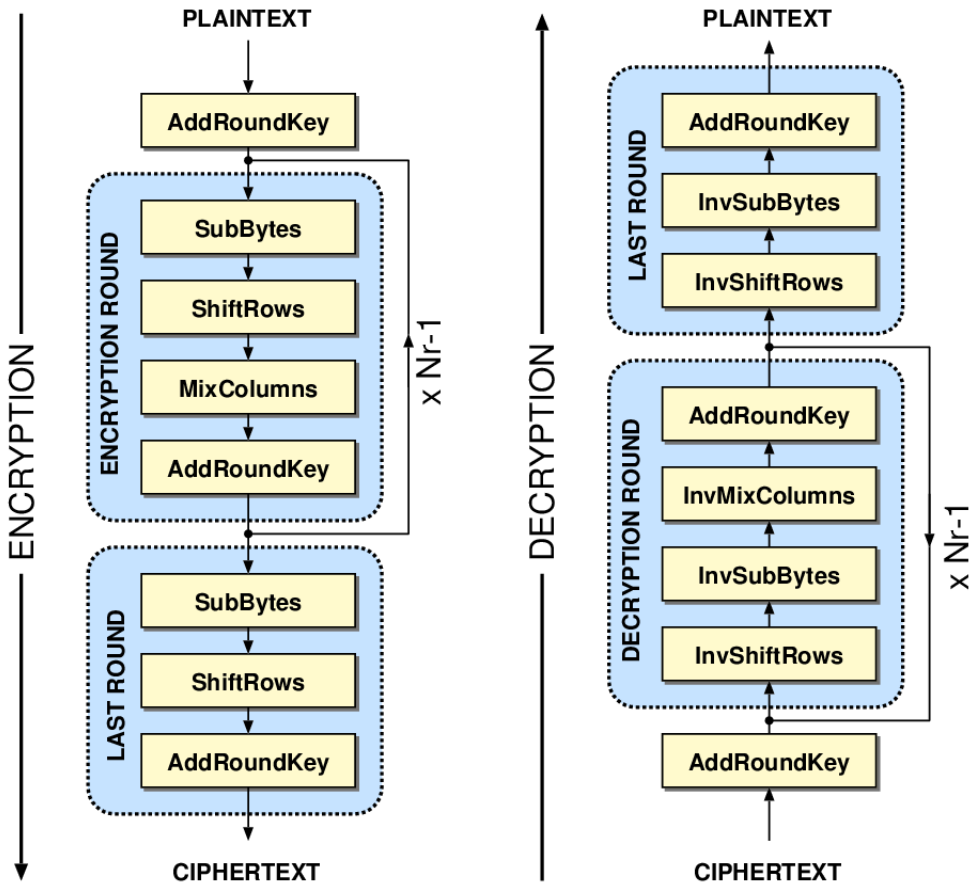

Fig. 2: The AES Algorithm.