## Lines that will appear in the proceedings

## Lines that will appear in the program

6C-28 Design and Implementation of a Secret Key Steganographic Micro-Architecture Employing FPGA
M. M. Saeb, H. A. Abdul Moneim (Arab Academy for Science, Tech. and Maritime Transport, Egypt)

# Design and Implementation of a Secret Key Steganographic Micro-Architecture Employing FPGA

Hala A. Farouk, Magdy Saeb

Computer Engineering Department
Arab Academy for Science, Technology and Maritime Transport
Alexandria, EGYPT
Tel:+20106009703
e-mail: mail@magdysaeb.net

**Abstract** – In the well-known "prisoners' problem", a representative example of steganography, two persons attempt to communicate covertly without alerting the warden. One approach to achieve this task is to embed the message in an innocent-looking cover-media. In our model, the message contents are scattered in the cover in a certain way that is based on a secret key known only to the sender and receiver. Therefore, even if the warden discovers the existence of the message, he will not be able to recover it. In other words a covert or subliminal communication channel is opened between two persons who possess a secret key to reassemble its contents. In this article, we propose a video or audio steganographic model in which the hidden message can be composed and inserted in the cover in real-time. This is realized by designing and implementing a secret key steganographic micro-architecture employing Field Programmable Gate Arrays FPGA.

## I. INTRODUCTION

In this work, we present a hardware implementation of a secret-key steganographic algorithm. The basic idea of our algorithm is selecting the hiding bits in a pseudorandom manner as a function of a secret key to increase obscurity. The receiver needs only the modulated cover and the secret key to recover the message, i.e., no original cover is required.

## II. THE ALGORITHM

```
ALGORITHM STEGO
Input: Message M, Cover C, Key K, State Register SR
1. Load a block of the message Blk_i
   into the message cache MC
2. Load Key into the key Cache KC
3. Generate an address Ad_i
4. Address memory to get one cover word CW
5. Hide two message bits (m_i, m_i+1)
7. If message cache is not empty
        7.1 Circulate key cache one bit right
        7.2 Shift message cache one bit right
        7.3 Goto 3
8. Else if message cache is empty
      If message not finished
        8.1 Load next block into message cache
        8.2 Goto 3
        Else if message is finished then halt
   End Algorithm.
        Output: Modulated Cover CM
```

## III. THE MICRO-ARCHITECTURE

The architecture is divided into an embedder processor and an SDRAM controller. The embedder processor basic components are shown in Figure 1. The embedder processor issues read and write commands to the memory, which are processed and reformatted by the SDRAM control and waits for a confirmation from memory to ensure stabilized output. The controller halts the process when hiding is complete. In the next section we discuss the various building blocks of our proposed micro-architecture.

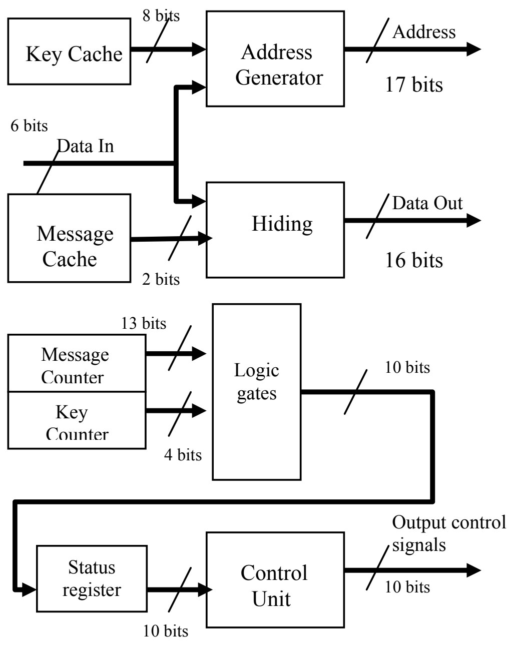

*Figure 1: Micro-architecture block diagram*

### The Embedder Processor Overview and Organization

The embedder processor generates addresses to initially cache the message and key from memory. It also generates addresses to access the cover randomly for message bit hiding. The embedder processor is composed of an address generator module, key cache, memory cache, key cache counter, message counter, message cache counter, message pointer, stegoblock, address multiplexer, control unit, and status register.
The address generator is composed of a shuffler, a block pointer memory and a shift & concatenate unit. The address generator receives an eight bit key and outputs an address of 17 bits as shown in Figure 2. We have chosen 17 bits only in order to access an image of size no more than 128 Kwords, e.g. 256Kbytes. It is a common size where video frames most probably never exceed.

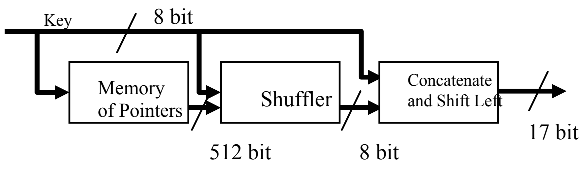

*Figure 2: Address generator block diagram*

The shuffler module receives 512 bits from the block pointer memory module and based on the key, it selects one of 64 pointers to be transmitted to the shift and concatenate unit. Each pointer is eight bits. Therefore, the address space for each pointer is 256 words. We consider these 256 words as one block. Therefore, if each time the octet generated from the key is different from the one generated before, then the message bit will be inserted into a different block in the image. As there are only 64 pointers, only 64 blocks can be addressed. This means that only 64x256 words can be used for hiding. This problem was overcome by using the upper bits of the octet generated from the key as a segment selector. Each segment is 16384-word large. As a result of this improvement, the message bits may be 16384 words apart in the best case and one word apart in worst case. This worst case will happen if a large number of octets in the key are repeated. Therefore, we have developed a short program for generating a key that covers the whole cover image and attempts all blocks evenly. This new key also avoids large repetition in the key octets. Therefore, the hiding will not be biased towards a certain area of the image.
The message cache is organized as a rather large shift register. Blocks of this message is cached during the hiding process. The bits that are shifted out are the message bits that are needed to be hidden in the cover image.
The message cache improves performance as it saves eight memory calls per bit, during the hiding process. The memory used in this design is a synchronous dynamic RAM, which requires more cycles in the read and write operations than the static RAM. The dynamic RAM on the other hand is larger and can support large images as is needed in our case. The SDRAM is divided into 512 columns and 4096 rows. The addressing of consecutive words in same memory row requires three clock cycles. The addressing of words spaced out in different memory rows requires from 8 to 9 clock cycles. Since, changing the address from the cover location to the message location in the memory requires from 8 to 9 clock cycles, it is better to address the message consecutively and then address the cover. This improves performance and is achieved by caching the message before the start of the embedding process.
The control signals are generated in the hardwired control unit and provide control inputs for all integrated modules. The block diagram of the control unit is shown in Figure 3.

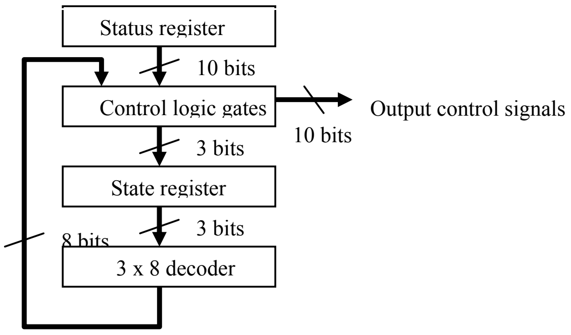

*Figure 3: Control unit block diagram*

It consists of one decoder, a state register, and a number of control logic gates. Based on some logic operations and on the state signals generated from the state register, a 10-bit output is generated by the control unit. This output consists of a ***msgptrinc*** bit, which increments the address of the next message block to be loaded into cache, a ***msgcacheload*** bit that controls message cache loading, a ***memwr*** and a ***memrd*** bit that control the read/write signal of the SDRAM and some other control signals that synchronizes the dataflow between all the above mentioned components.

## SUMMARY AND CONCLUSION

Motivated by the need for a fast hardware implementation suitable for real-time applications, we have provided a micro-architecture of a secret-key steganographic FPGA implementation. The distinctive features of this design are as follows:

- The address generator generates an output every clock cycle. This is a major advantage as compared to other algorithm like SHA-based algorithm that requires 210 cycles or MD5 designs with 342 cycles.
- The shuffler design is our conceptually developed hardware that provides the required randomization in the embedding process.
- The shuffler operates in parallel with the hiding module. This saves about 25 ns for each hidden bit.
- This address generator design is particularly suitable for a special-purpose processor design since it needs large sizes of busses, like 64-bit and 512-bit busses, which cannot be supported by general-purpose processors.
- The address generator is capable of generating an address in a 32,768-byte block or in multiple of these blocks based on the user preference. This allows the user to efficiently handle different image sizes.
- The address generator can generate various sequences of addresses for different frames from a single 32-byte key by XORing with the cover. This is an essential requirement for video hiding schemes, since it is not realistic to ask the user for a new key for each video frame.

We believe that our approach is refined enough to escape the watchful eyes of a passive adversary (the warden). Comparing this architecture with other types of steganographic algorithm implementations, we have demonstrated the dominance of our algorithm in time-critical applications.

## IMPLEMENTATION RESULTS

Target Device: x2s100
Target Package: tq144
Target Speed : -6
Mapper Version: spartan2 -- C.22

**<u>The Total Design</u>**

<u>Timing Summary:</u>

Minimum period: 48.811ns (Maximum frequency: 20.487MHz)
Maximum combinational path delay: 49.783ns
Maximum net delay: 11.980ns

<u>Device utilization summary:</u>

| | | |
|---|---|---|
| Number of External GCLKIOBs | 2 out of 4 | 50% |
| Number of External IOBs | 43 out of 92 | 46% |
| Number of SLICEs | 1195 out of 1200 | 99% |
| Number of DLLs | 2 out of 4 | 50% |
| Number of GCLKs | 1 out of 4 | 25% |
| Number of TBUFs | 2 out of 1280 | 1% |

## CIRCUIT REALIZATIONS

This design is implemented and downloaded on the FPGA Spartan 2S100TQ144-6 chip. Figure 4 shown the design placed and routed on the FPGA chip.

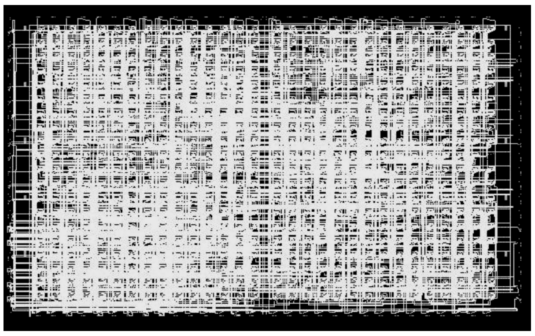

*Figure 4: Image of the Design placement on the FPGA*