FPGA-Based Real-Time Implementation of AES Algorithm for Video Encryption

SONIA KOTEL1,2, MEDIEN ZEGHID2,3, ADEL BAGANNE1, TAOUFIK SAIDANI2, YOUSEF IBRAHIM DARADKEH1, TOURKI RACHED2,
1Laboratory of Information Science and Technology, Communication and Knowledge (Lab-STICC),
University of South Brittany (Lorient-France)
2Laboratory of Electronics and Microelectronics, Faculty of Sciences, University of Monastir, Tunisia
3College of Engineering -at Wadi Ad-Dawasir, university salman bin abdulaziz, KSA
medien.zeghid@fsm.rnu.tn

Abstract: - Multimedia data security is becoming an important concern due to the fact that multimedia applications affect many aspects of our life. To deal with the increasing use of multimedia in industrial process, security technologies are being developed. Multimedia encryption algorithms implemented in hardware have emerged as the most viable solution for improving the performance of Multimedia encryption systems. The introduction of reconfigurable devices and high level hardware programming languages has further accelerated the design of encryption technology in FPGA. In this paper, we report on the implementation and hardware platform of a real time video encryption processing. The processing encrypts videos in real time using the AES algorithm. We propose a computationally efficient architecture for AES. The system is optimized in terms of execution speed and hardware utilization. The design as know AES encryption processor is developed in Xilinx System Generator and integrated as a dedicated hardware peripheral to the Microblaze 32 bit soft RISC processor with the EDK embedded system and implemented targeting a Spartan3A DSP 3400 device (XC3SD3400A-4FGG676C). The video encryption processing has been verified successfully. The input comes from a live video acquired from a CMOS camera and the encrypted video is displayed on a DVI display screen.

Key-Words: - Multimedia, Encryption, Real Time, AES, Field Programmable Gate Array, Xilinx System Generator (XSG), Microblaze Processor, Embedded Development Kit (EDK).

1 Introduction

Nowadays, the wide use of digital images and videos in various multimedia applications brings serious attention for keeping data secure from unauthorized attackers. Video on demand, Internet television and Video conferences are typical examples. Confidentiality is one of the primary concerns for commercial uses of multimedia communication. For example, in a business video conference only the participating members are allowed to receive the audio and video data to protect the privacy of the negotiation. So data encryption is a suitable method to this privacy issue. Multimedia encryption challenges originate from two realities. Firstly, multimedia data have great volumes. Secondly, they need real-time uses [1].

Some video encryption schemes for secure transfer of multimedia information have been developed in the past decade [2]. Many of these approaches exploit the inherent characteristics of video data and the steps taken to compress and encode video information. These techniques can be classified into three types according to the target data selected from compression stages. The first one is named “Spatial Domain” schemes, which apply encryption to the original video data directly as in [3]. The second scheme is named “Bitstream Domain”, which encrypts the code words of compressed bitstream [4]. The third schemes known as “Frequency Domain” has been proposed, which use the results of DCT or DWT transformation and quantization stages from compression process [5]. Commonly the Advanced Encryption Standard (AES) algorithm is used for video encryption [5][6][7]. The AES is the most popular algorithm used in symmetric key cryptography due to its performance and security level [8]. It is considered to be efficient both for hardware and software implementations. Several authors developed different AES architectures for high-speed and high-performance applications [9-10-11-12-13-14-15-16-17-18-19]. Hardware cryptographic algorithms implementations are, by nature, more physically secure, as they cannot easily be read or modified by
an outside attacker. Reconfigurable hardware devices such as FPGAs has been proposed as a way of obtaining high performance for video encryption processing, even under real time requirements [20]. FPGAs offer many performance benefits for executing video processing applications. A Xilinx tool, the System Generator for DSP [21-22], offers an efficient and straightforward method for transitioning from a PC-based model in Simulink to a real-time FPGA based hardware implementation. Furthermore, Xilinx Embedded Development Kit (EDK) tools make it possible to implement a complete video processing system on a single FPGA using hardware/software codesign methods.

The goal of this work was to develop a real-time video encryption system based on AES with a input from a CMOS camera and output to a DVI display and verified the results video in real time, with a focus on achieving overall high performance, low cost and development time. So, we define and we develop a unified hardware architecture can run the AES Encryption-Decryption algorithm to encrypt a video application. Furthermore, this paper implements the proposed AES-video encryption processor in a Xilinx FPGA using System Generator for DSP.

This paper is structured as follows. The next section details the AES algorithm. Section 3 discusses different metrics in order to choose a suitable architecture for AES in multimedia application. Section 4 details the proposed architecture and the overall design of the AES video encryption processor implementation. Also presented are the area and timing and some discussion and comparison in section 5. In Section 6, Hardware/Software Co-Design in System Generator and experimental results are detailed and also issues related to security levels of the encryption video processor considered are discussed. Finally, concluding remarks are made in the last section.

2 The AES Encryption-Decryption Algorithm

In November 2001, the National Institute of Standards and Technology (NIST) chose the Rijndael algorithm as the new Advanced Encryption Standard (AES) [8]. The choice of this algorithm is based on three major evaluation criteria categories:

- Security or resistance of the algorithm to cryptanalysis.
- Cost that encompassed computational efficiency (speed) on various platforms, and memory requirements.
- Implementation characteristics such as flexibility, hardware and software suitability, and algorithm simplicity.

The AES algorithm operates on 128 bits of data as known plaintext and generates 128 bits of output. The length of the key used to encrypt this input data can be 128, 192 or 256 bits. Each block of the input data with the encryption key operate on array of bytes and organized as a 4×4 matrix that is called the state [8].

For full encryption process, the plaintext is passed through Nr rounds (Nr = 10, 12, 14) [8]. These rounds are governed by four transformations: AddRoundkey; Subbytes; Shiftrows; MixColumn. Our design has covered the standard AES version (128 bits-AES). 128 bits AES calculates a 128-bit cipher data with a fixed 128-bit key length and it consists of the following steps:

- Initial AddRoundkey transformation with the original key.
- 128 bits AES requires 10 rounds to produce the 128-bit cipherdata. Each round requires the previous round’s results, cipherdata(t-1), as well as the round key Key(t). Each round performs the four different transformations of the AES except the final round. The last AES round omits the MixColumns transformation.

The decryption structure has exactly the same sequence of the encryption process. It includes the reverse of the all AES operations used in encryption process as known: AddRoundkey; InvSubbytes; InvShiftrows; InvMixColumn.

3 Choice of An architecture for AES Video Encryption Processor

The choice of a suitable architecture for a given application has a significant impact on system performance. This can be viewed as throughput and the resources consumed, however this purpose are driven by the modes of operation for the cipher. To this end, our objective is to define appropriate architecture for the AES processor.

3.1 Operating Mode Choice

In order to choose the efficient mode, which can be the most suitable to our AES-processor we can used two metrics: Security and the implementation cost. The FIPS-197 specification details a number of modes of operation for the cipher, there are two kinds of modes: feedback and non feedback modes.
Table 1 summarizes modes of operation including the name of the mode as well as the metrics.

<table>
<thead>
<tr>
<th>Metrics-Modes</th>
<th>ECB</th>
<th>CTR</th>
<th>CBC</th>
<th>CFB</th>
<th>OFB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security</td>
<td>Low</td>
<td>Medium</td>
<td>High</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>parallelism</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Decryption</td>
<td>Yes</td>
<td>Now</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Random access</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Rapidity</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Complexity</td>
<td>No</td>
<td>Now</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Error propagation</td>
<td>No</td>
<td>Low</td>
<td>High</td>
<td>High</td>
<td>High</td>
</tr>
</tbody>
</table>

Table 1: Comparison of Different Operating modes

It is clear from table 1 that the feedback modes, like Cipher Block Chaining (CBC) and Cipher Feedback (CFB) mode enable very high security level; but they imply large cost. In the non feedback modes, like Electronic Code Book (ECB) mode and counter (CTR) mode, encryption of each subsequent block of data can be performed independently from processing other blocks, thus all blocks can be encrypted in parallel. So, ECB and CTR modes assure the maximum speed with a medium level of security against other modes. Hence, we chose CTR mode because its security level is higher compared to the ECB mode.

3.1 Architecture Choice

For the implementation of the AES algorithm on FPGA, several parameters have a significant impact on the systems performances [9] [10] [11]. These metrics are the throughput, the power consumption, the area and the modes operations.

There are many Implementations of AES in the literature. The High speed implementation of AES can be divided into two main kinds of architectures: the iterative approach for implementation and the loop unrolling architectures.

So far, the iterated architecture leads to the smallest implementation. Such architecture consists in a round component, which is fed to its output. The loop-unrolled architectures achieve two or more rounds per clock cycle and the execution of the cipher is iterated. So, loop-unrolled architectures enable very high-speed implementations; but they imply large area and high power consumption. The speed increases with degree of unrolling.

However, here we aim to produce fast real time implementations of the AES because multimedia application needs real-time encryption.

So, the iterated architecture is unattractive for multimedia application. In order to have the maximum speed/area/power ratio for video encryption, we choose the limited loop unrolling architecture in our processor implementation. The section below presents the details of the design.

4 Choice of An architecture for AES Video Encryption Processor

4.1 Proposed Processor Design

In this section we present the proposed architectural design of the AES-CTR video encryption processor. The top level view of the AES-CTR video encryption processor for a partial unrolling design is shown in Figure 1.

In our architecture, the encryption-decryption datapath is based on three rounds implementation of the AES algorithm. The first, the second and the third AESRound perform the four different steps of the AES algorithm namely SubBytes, Shiftrows, MixColumns and AddRoundKey. Therefore, it takes total of 11 cycles to encrypt or decrypt a 128-bit block of data.

In our proposed architecture, the input and the output interfaces units have also been integrated in order for the proposed processor to communicate efficiently with the external environment. They take care of reading input data and writing encrypted-decrypted output. They are controlled by the data-ready (StartAES), ciphertext-ready (End-Encrypt) and clk signals. When the bus puts a data to be read or write this signal is selected and the data is taken. In the following, the system architecture’s basic units are described.

4.2 Processor Implementation

In our architecture the goal is to permit next round’s available inputs to calculate instantaneously an intermediate result. The aim is to obtain a resource
efficient implementation, increasing the achievable data throughput.

4.2.1 Key Expander Unit
The Key Expander unit is used to compute a set of round keys based on a ciphered key. A synchronous finite state machine (FSM) is used for this purpose. After the initialization phase, the Key Expander unit is ready to accept the initial key and to generate new keys for all rounds. The START signal is used to acknowledge an original key request from the core. Thereafter the corresponding keys are generated. The end of the iteration is indicated by a low-state of the START signal. All generated keys are registered in three Block-RAMs according to the key round number.

4.2.2 Embedded Block RAM Components
For our proposed architecture, a memory elements configured as RAM are used to store the RoundKeys (Figure 2). Three RAM’s with 2-bit input address and 128 bit output are used. The three RAM’s as know IPRam1, IPRam2 and IPRam3 blocks are automatically generated by the Xilinx tool. The first AESRound uses the encryption key from the IPRAM1. Then, each turn uses its own key. This arrangement allows simple decoding logic to select the appropriate key for the both AESRound.

4.2.3 AES Round Unit
An unrolled architecture implements multiple rounds of the core encryption function in combinational logic, thereby reducing the number of clock cycles required to compute the encrypted. This technique allows for an improvement in the throughput. Our approach is to condense three rounds in one.

As was already mentioned in the previous section, each basic round of the architecture is composed of basic building blocks. The encryption data generation is mainly centered in the AES Round unit. The AES Round unit is the main data path component of the system architecture. In order to increase the achievable data throughput the AES Round unit is composed on three AES rounds unit implementation as know AESRound1, AESRound2 and AESRound3 units. In this case the latency of our architecture is reduced to three clock cycles. And a new set of encryption-decryption output is generated every three cycles.

The AESRound1 unit is ready to accept data for encryption when GoAESRound is asserted. The GoAESRound signal is used to acknowledge a data request from the controller unit. After 4 cycles, the signal GoAESRound2 is asserted as the AESRound2 unit which continues the data encryption and the AESRound1 start another encryption process. In addition AESRound2 unit follow the encryption ending in 4 cycles and so that the AESRound3 unit can complete its operation in three cycles after receiving a GoAESRound3. Figure 3 shows the timing diagram for the AES Round Units.

4.2.4 Controller Unit
The Controller Unit is designed to control the flow of data in the design, as well as the movement of data between the AES interface Unit, key Expander unit and AES Round Unit. The Controller Unit coordinates all the system operations and processes. A FSM is used for this purpose. Figure 4 shows the state machine.
During the initialization phase, the controller unit gets the first operation XOR between the InitVect (the first Initialization Vector) and the original key. After the initialization phase, the control unit is totally responsible for the system operation. It defines the appropriate control-signals for the key expander unit to start the generation of the AES Round Keys that will be stored in the Block RAM Components. The read/write addresses are also generated by the Controller unit. It also controls the AES Round unit to start the requested encryption operation.

5 Implementation Results and Comparison

We implemented an FPGA design that efficiently performs AES video encryption scheme. The described circuits have been implemented in VHDL using the Model Technology’s ModelSim Simulator and synthesized, placed, and routed using target device of Xilinx (Xilinx Spartan 3A XC3SD3400A-4FGG676C FPGA). The architecture was simulated for verification of the correct functionality, by using the test vectors provided by the AES standard. Four components. The read/write addresses are also generated by the Controller unit. It also controls the AES Round unit to start the requested encryption operation.

<table>
<thead>
<tr>
<th>References</th>
<th>Architecture</th>
<th>Area (Luts)</th>
<th># of Bram</th>
<th>D Gbps</th>
<th>TPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>[9] XCV1000</td>
<td>Rolling Encryption</td>
<td>1743</td>
<td>-</td>
<td>1.850</td>
<td>0.106</td>
</tr>
<tr>
<td>[9] XCV1000</td>
<td>Rolling Encry/Decry</td>
<td>3555</td>
<td>-</td>
<td>1.049</td>
<td>0.02</td>
</tr>
<tr>
<td>[10] XCV2000E-7</td>
<td>Pipelined Encryption</td>
<td>17425</td>
<td>-</td>
<td>25.10</td>
<td>0.14</td>
</tr>
<tr>
<td>[12] XCV1000E-7</td>
<td>Pipelined Encry/Decry</td>
<td>7761</td>
<td>400</td>
<td>16.08</td>
<td>0.207</td>
</tr>
<tr>
<td>[13] XCV2VP70-7</td>
<td>Pipelined Encryption</td>
<td>5177</td>
<td>84</td>
<td>21.54</td>
<td>0.41</td>
</tr>
<tr>
<td>[15] XCV24000</td>
<td>Pipelined Encryption</td>
<td>16938</td>
<td>0</td>
<td>23.57</td>
<td>0.139</td>
</tr>
<tr>
<td>[16] XCV1000</td>
<td>Pipelined Encry/Decry</td>
<td>17314</td>
<td>32</td>
<td>3.65</td>
<td>0.02</td>
</tr>
</tbody>
</table>

Table 2 FPGA Synthesis Results

<table>
<thead>
<tr>
<th>References</th>
<th>Architecture</th>
<th>Area (slices)</th>
<th>Frequency (Mhz)</th>
<th>Power (mW)</th>
<th>Throughput (Gbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[9] XCV1000</td>
<td>Rolling Encryption</td>
<td>3047</td>
<td>100.329</td>
<td>4.296</td>
<td>4.28</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>References</th>
<th>Architecture</th>
<th>Area (slices)</th>
<th>Frequency (Mhz)</th>
<th>Power (mW)</th>
<th>Throughput (Gbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[9] XCV1000</td>
<td>Rolling Encry/Decry</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>[10] XCV2000E-7</td>
<td>Pipelined Encryption</td>
<td>5177</td>
<td>84</td>
<td>21.54</td>
<td>0.41</td>
</tr>
<tr>
<td>[13] XCV2VP70-7</td>
<td>Pipelined Encryption</td>
<td>7761</td>
<td>400</td>
<td>16.08</td>
<td>0.207</td>
</tr>
<tr>
<td>[15] XCV24000</td>
<td>Pipelined Encryption</td>
<td>16938</td>
<td>0</td>
<td>23.57</td>
<td>0.139</td>
</tr>
<tr>
<td>[16] XCV1000</td>
<td>Pipelined Encry/Decry</td>
<td>17314</td>
<td>32</td>
<td>3.65</td>
<td>0.02</td>
</tr>
</tbody>
</table>

Table 3: Performance comparison results


Using the throughput metric the proposed design is more proved better by compared with [9]-[16]-[17]. When compared with [10-11-13-14] the results show higher reduction in area and number of BRAMs.

To achieve more accurate measure of chip utilization, CLB slice count as chosen as the reliable area measurement. Therefore, to measure the hardware resource cost associated with an implementation’s resultant throughput; the throughput per slice (TPS) metric is used. We defined it as

\[
\text{TPS} = \frac{\text{Throughput}}{\text{Slice Count}}
\]
Using the TPS metric our design is proved better by compared with [12-15]. When compared with [10-11-14], architectures for high throughput, the results show higher reduction in number of BRAMs. Our design has 6 embedded RAM. The BRAMs are used in the keysRound storage.

6 Hardware/Software Co-Simulation of AES-CTR Video Encryption Processor

6.1 Description of the Design Flow

Our main goal is to verify that the hardware and software specifications are valid. This requires us to test each AES processor unit and verify the efficiency of the system in the presence of real-time constraints (throughput, time, etc.).

The integration of an IP (Intellectual Property) Core in a real-time hardware design is the most complex task, which requires a methodology for implementing real-time applications on a reconfigurable logic platform.

The flow consists in developing and synthesizing our IP, as know AES-CTR processor to be integrated with the Xilinx System Generator tool in the EDK flow which used to transform the RTL implementation into a complete FPGA implementation [21]-[23][24]. Once our IP is valid, after its integration and export as a PCORE to Platform Studio Project, the following step is the communication between the MicroBlaze processor and the PCORE often occurs over shared bus connectivity.

This communication also provides a hardware/software Co-Simulation environment to test the embedded processor design. The conception flow of the AES-CTR processor real-time design is shown in figure 5.

6.2 Co-Design of the AES-CTR Processor

Before integrating our AES-CTR processor in the real-time embedded and reconfigurable system, we first need to check its functionality by implementing video processing designs as a MATLAB/Simulink model using the optional Video and Image Processing Blockset. Many Co-simulation models are developed for our application. They depend on the type of input multimedia data (text, image and video). For these different Co-simulation types we have used the 'Black Box' of System Generator Blockset to incorporate the VHDL code of the AES-CTR processor into a System Generator model.

6.2.1 Co-simulation model: color image

The co-simulation model for Encryption/Decryption color image includes a block of image acquisition, a conversion block, another block for the display before and after encryption and the XSG Black Box integrating the AES-CTR processor. These different blocks of the Simulink/System Generator Co-Simulation model allow the serialization of the image to be processed and its reconstruction once the output pixels are generated by the encryption/decryption hardware model. Figure 6 illustrates the general structure of the proposed model.

Fig. 6 Co-Simulation model: RGB Image Encryption

Figure 7 shows the obtained results of the Co-simulation model.
6.2.2 Co-simulation model: Video
The co-simulation model for Encryption/Decryption video includes an acquisition block of the video, a conversion block, another block for the display before and after encryption and the XSG Black Box corresponding to the AES-CTR processor. Figure 8 shows the results obtained after the encryption and decryption of a video sequence by the AES-CTR processor.

6.3 Real-time validation of the AES-CTR Processor
We have designed our AES-CTR processor block in System Generator and integrated it as a dedicated hardware peripheral with the EDK embedded system. This integration can convert DSP designs into custom peripherals for Xilinx Platform Studio, and connect them to the MicroBlaze soft-core processor using the PLB bus microprocessor.

The VSK reference designs are provides for accelerate the development of video applications running on Xilinx FPGAs [24]. We integrated our AES-CTR processor in the Camera Frame Buffer reference design which is shown in figure 10. The Camera Frame Buffer reference design allows capturing a video stream from a camera, performing processing on the video stream, buffering the video stream in external memory and displaying the processed video at a different rate.

The Video Starter Kit (VSK) consisting of Spartan 3A DSP XCSD3400A FPGA shown in figure 9 is connected to a Micron CMOS camera of resolution 720 x 480 pixels delivering frames at 60 fps through a FPGA [24]. Mezzanine Card (FMC) Daughter is used for decoding the data arriving through the serial LVDS camera interface. The deserialized input consists of V-Sync, H-Sync and 8 line data bus which are used as the input for the AES-CTR processor model. The AES-CTR processor model is applied in the Camera Processing block on the input signal arriving from the Camera In block. The output signal is Gamma corrected for the output DVI monitor and is driven by Display controller to the DVI output monitor. Video to VFBC and MPMC core helps us to store the image data and buffer them to the output screen.

In addition, the high-level model of VSK_Camera_VOP model has three inputs (vs_in, hs_in and data_in) and six outputs (vs_out, de_out, hs_out, red_out, green_out and blue_out).
Our AES-CTR processor real-time Hardware Co-simulation consists in a video signal encryption/decryption which provides acquisition, processing and streaming of a video signal from a video source (camera, DVI). So the real time for us is the fact of having the same flow of images between the input video source and the output streaming according to our encryption/decryption processing.

The final implementation of our programmable platform occupies 39% of the inputs/outputs and 49% of the logic elements.

Despite the circuit complexity, the software/hardware integration has been possible by exploiting the flexibility of the Spartan 3A DSP device to configure a hardware architecture optimised for a security system.

At the time level, 71% would be required for camera and VGA transfers and 29% for encryption/decryption real-time processing. Furthermore, it is obvious that this solution of Hardware Co-Simulation using System Generator, and integration of hardware peripherals and embedded processing accelerates the development and the validation of our AES-CTR processor for video encryption.

The two figures below show the result of the cryptographic system development chain (acquisition, processing (encryption/decryption), displaying a video signal).

6.4 The Security Analysis

An efficient cryptosystem should be resist against all kinds of know attack such as statistical attacks. In this section, the statistical analysis has been performed on the AES-CTR processor. This is shown by a test on the histograms of the enciphered video, the correlation of adjacent pixels in the ciphered video and on the entropy and PSNR. Many test video are used in our case.

![Fig.12. Video encryption tests.](image)

**Fig.12. Video encryption tests.**

- **Histograms of Encrypted video**

  We select several video having different contents, and we calculate their histograms. Three typical examples among them are shown in figure 13. We can see that the histogram of the ciphered video is fairly uniform and is significantly different from that of the original video. Therefore, it does not provide any indication to employ any statistical attack on the video under consideration. Moreover, there is no loss of video quality after performing the encryption/decryption steps.

  ![Fig.13. Histograms of the plain video and ciphered video](image)

  **Fig.13. Histograms of the plain video and ciphered video**

- **Correlation coefficients Analysis**

  We have also analyzed the correlation between two vertically, horizontally and diagonally adjacent in plain video and cipher video respectively. We calculate the correlation coefficient of each pair by using the following formula.

  \[
  r_{xy} = \frac{\text{cov}(x, y)}{\sqrt{D(x)}\sqrt{D(y)}}
  \]

  Correlation results for Horizontal, vertical and diagonal directions were obtained as shown in Table IV. It is clear that from the table 4 and figure 14 that
there is negligible correlation between the two adjacent pixels in the cipher video. However, the two adjacent pixels in the plain video are highly correlated.

<table>
<thead>
<tr>
<th>correlation</th>
<th>Original video</th>
<th>Encrypted video</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Akiyo</td>
<td>Foreman</td>
</tr>
<tr>
<td>Horizontal</td>
<td>0.984</td>
<td>0.7373</td>
</tr>
<tr>
<td>Vertical</td>
<td>0.950</td>
<td>0.8493</td>
</tr>
<tr>
<td>Diagonal</td>
<td>0.978</td>
<td>0.804</td>
</tr>
</tbody>
</table>

Table 4 Correlation coefficients of two adjacent pixels in original video and encrypted video

![Correlation of adjacent pixels in the original video and the encrypted video](image)

Fig. 14. Correlation of adjacent pixels in the original video and the encrypted video

- Peak Signal-to Noise ratio (PSNR)

The PSNR measure tests the quality of the reconstructed video that is given in Table V for different test video (Akiyo, Foreman and Mother-daughter). PSNR of encrypted video and original video is computed as follows:

$$\text{PSNR} = 10 \log_{10} \left( \frac{d^2}{EQM} \right)$$  \hspace{1cm} (3)

As shown in Table 5, the PSNR value between the encrypted video and the original video is small. The PSNR is less than 10 dB. This fact shows the high security of the both ciphered video.

<table>
<thead>
<tr>
<th>Original Video</th>
<th>Akiyo</th>
<th>Foreman</th>
<th>Mother-daughter</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSNR</td>
<td>9.314</td>
<td>8.673</td>
<td>8.431</td>
</tr>
</tbody>
</table>

Table 5 The PSNR Sensitive Tests

- Information Entropy

Entropy is a measure of the predictability of random source. As for a video, if the entropy value is 8, then the source is truly random. But, if the entropy is lower, the source has some degree of predictability. A secure system should satisfy a condition on the information entropy that is the cipher video should not provide any information about the original video. The information entropy is calculated by using the following formula:

$$H = - \sum p_i \log_2 p_i = \sum p_i Q_i$$  \hspace{1cm} (4)

As shown in Table 6, the value obtained is very close to the theoretical value of 8. This means that the encryption process can’t leak the original information and our cryptosystem is secure upon the entropy attack. This results analysis shows the AES-CTR processor does not provide any indication to employ any statistical attack on the encrypted video.

<table>
<thead>
<tr>
<th>Original Video</th>
<th>Akiyo</th>
<th>Foreman</th>
<th>Mother-daughter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entropy</td>
<td>7.984</td>
<td>7.7373</td>
<td>7.9825</td>
</tr>
</tbody>
</table>

Table 6: Entropies of the encrypted video

7 Conclusion

In this paper, a design for real time video encryption processing on Spartan 3A DSP 3400 (XC3SD3400A-4FGG676C) is proposed. To design the video encryption system we have used the Xilinx System Generator. DSP designs captured in System Generator can be converted into custom peripherals for Platform Studio and connected to the embedded system using the processor local bus. So, in this paper we applied the AES algorithm toward the cryptography of continuous video streams. To express the tradeoffs between security and real time application requirements; we have proposed a partial unrolling architecture AES. Our circuit has a 3x-unrolled core. The proposed architecture has compared in terms of consumed resources, throughput and TPS product with other related well known works, our system performs better performances. AES was implemented at a rate of 60 fps for an input frame of resolution 720x480. The implemented system architecture required 71% at the time level for camera with VGA transfers and 29% for encryption/decryption real-time processing.

Acknowledgements

We authors would like to thank Deanship of Scientific, Salman Bin Abdul Aziz University, Saudi Arabia for providing financial assistance to do this research a successful manner.

References:


