Table of Contents

1. Overview

1.1. Background and Motivation

Multimedia data security is becoming a significant concern since multimedia applications affect many aspects of our life. To deal with the increasing use of multimedia in the industrial process, security technologies are being developed. Multimedia encryption algorithms implemented on 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 technologies on FPGA. With good reliability and portability, taking up less resource, such implementations have lower cost and wider application prospects.

For some existing approaches, video encryption and decryption are implemented in software, which is easier to realize, but less efficient than hardware implementations. On the other hand, hardware cryptographic algorithms implementations are, by nature, more physically secure, as they cannot easily be read or modified by an outside attacker.

The Advanced Encryption Standard [1], as known as AES algorithm is a symmetric encryption algorithm which has excellent performance and security strength. There are different modes of AES algorithm (ECB, CTR, CBC, CFB, OFB). Considering speed, safety, and implementation cost, CTR mode has high speed, a medium level of security against other modes and a low cost (see Figure 1.1.1 [7]). This design implements the AES cipher in counter (CTR) mode on FPGA [2], it is well suited to process large amount of stream data in parallel, while maintaining reasonable security strength (see Figure 1.1.2).

Metrics-Modes	ECB	CTR	CBC	CFB	OFB
Security	Low	Medium	High	High	High
parallelism	Yes	Yes	No	No	No
Decryption	Yes	Now	Yes	No	No
Random access	Yes	Yes	No	No	No
Rapidity	Yes	Yes	No	No	No
Complexity	No	Now	Yes	Yes	Yes
Error propagation	No	Now	Yes	Yes	Yes
Implementation cost	Low	Low	High	High	High

Figure 1.1.1 Comparison of Different Operating modes

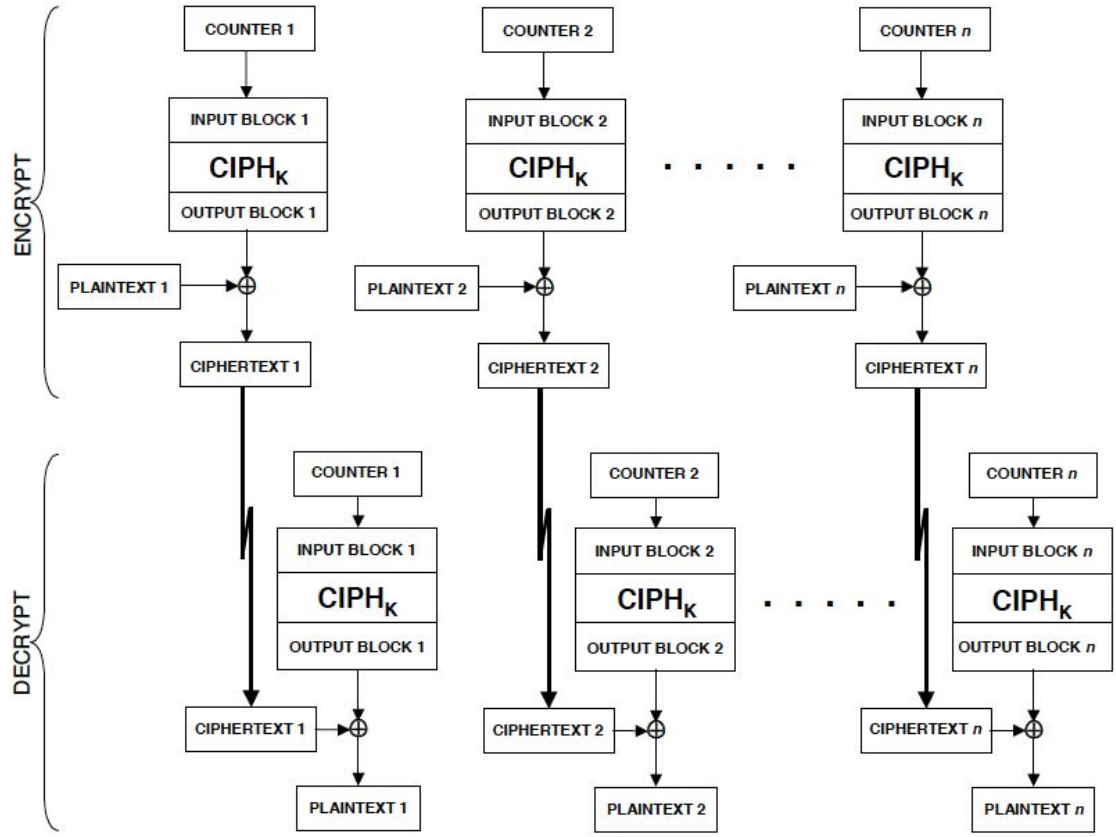

Figure 1.1.2 AES algorithm in counter (CTR) mode

1.2. Goal

The goal of this project is to develop a real-time video encryption and decryption system based on Advanced Encryption Standard (AES) in CTR mode on a Nexys Video Artix-7 FPGA boards. The system receives video stream through the HDMI IN port and outputs encrypted/decrypted video stream through the HDMI OUT port in real-time (30 fps ~ 60 fps) at 1080p.

1.3. Block Diagram

The following figure describes the system from a high level. The video stream first gets transferred into AXI4-Stream and stored into the DDR3 memory one frame at a time by the VDMA_0. The VDMA_1 reads each frame from the DDR3, feeds them into the AES_CTR for encryption/decryption, and stores the results into DDR3 again. Finally, those processed frames are read by the VDMA_0, encoded into HDMI signals and outputted through the HDMI OUT port.

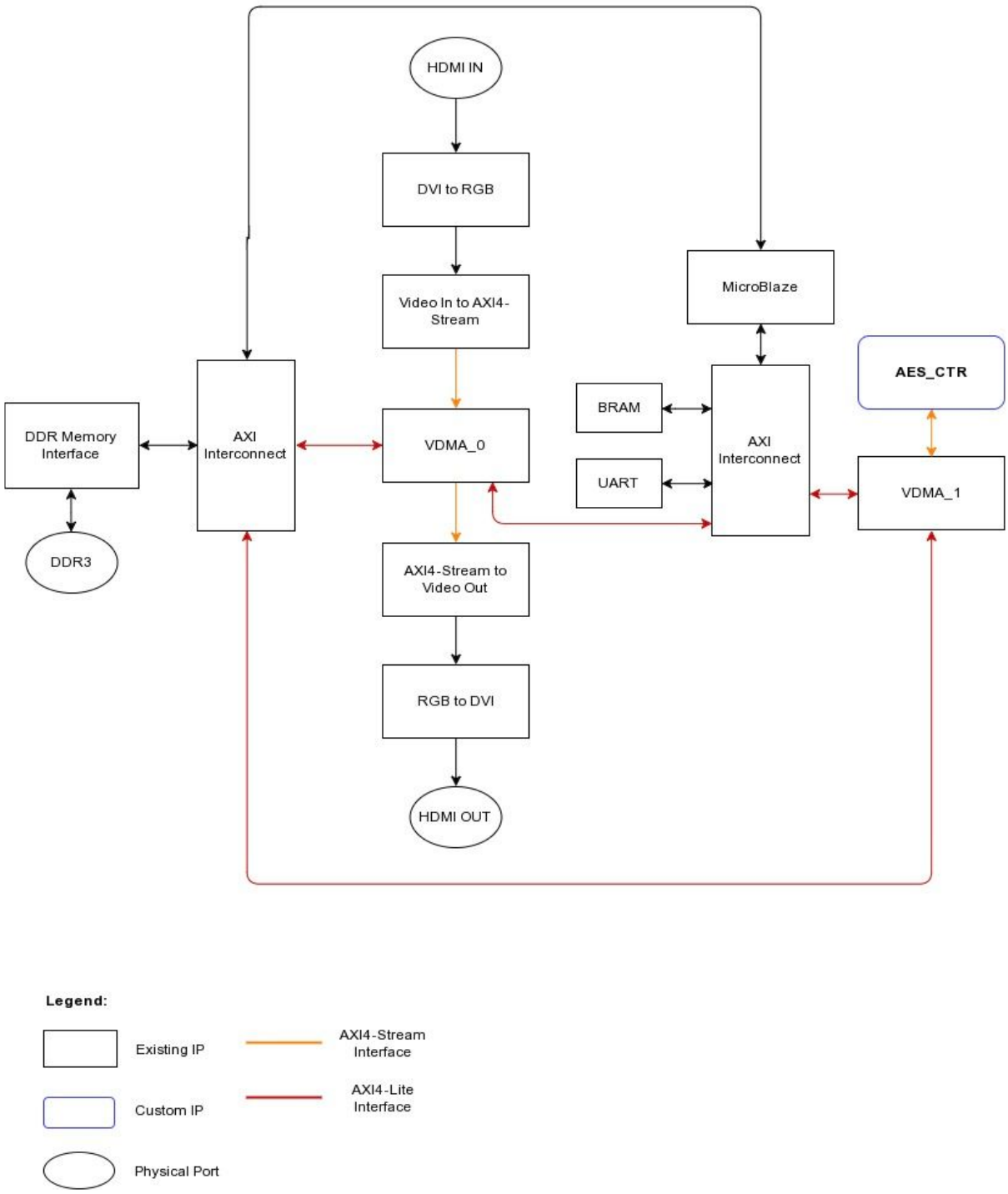

Figure 1.3.1 System-Level Block Diagram

1.4. IP Summary

The following table briefly describes the function and source of each IP.

Table 1.4.1 Brief IP descriptions

IP	**Source**	**Description**
Video in to AXI4 Stream	Xilinx	RGB stream to AXI4 Stream
AXI VDMA	Xilinx	Direct memory access between memory and AXI4-Stream video type targeted peripherals
AXI4 Stream to video out	Xilinx	AXI4 Stream to RGB stream
DDR Memory Interface	Xilinx	Memory controller, physical interface for DDR Memory
Microblaze	Xilinx	General purpose soft processor, system controller
AXI Uartlite	Xilinx	Interface and controller between UART data transmission and AXI-interface
BRAM	Xilinx	Configurable memory
AES_CTR	Team	Encrypt or decrypt Video Stream IP core
RGB to DVI	DIGILENT	Video encoder
DVI to RGB	DIGILENT	Video decoder

1.5 Referenced Design

The team used the HDMI demo project posted by Digilent on Github [1]. In the reference design, the system sends the video input stream directly to the HDMI output port; only one VDMA module is used to store frame data into the DDR3 memory. The team added another VDMA (VDMA_1 in figure 1.3.1) and the custom IP, AES_CTR, to accomplish the project requirements.

2. Outcome Results

2.1. Results

The team managed to achieve video encryption and decryption at 1080p. The plain video content cannot be distinguished in the encrypted video. Almost all the details are hidden after the encryption process, though the contour of some objects are still visually distinguishable (See Figure 2.1). The decrypted video stream looks exactly the same as the plain video, except that the frame rate is lower than expected.

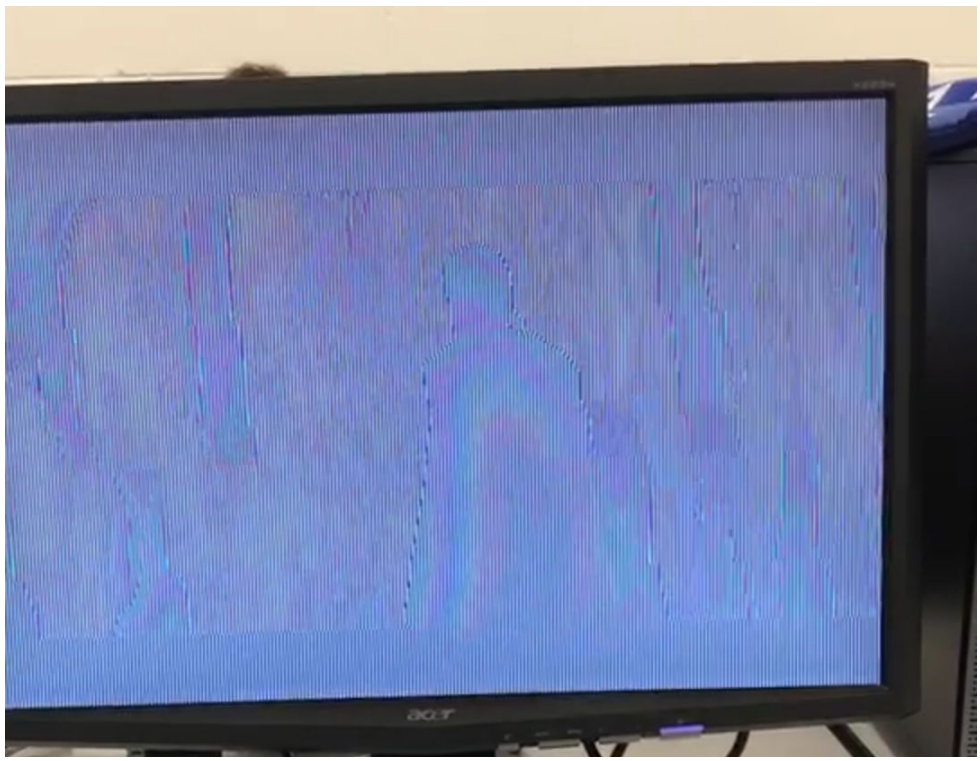

Figure 2.1 Screenshot of the encrypted video

The output encrypted/decrypted video stream has a low frame rate, about 2 ~ 5 frames per seconds at 1080p, which does not meet the requirement. The team tried to downscale the resolution to 720p, but due to the time limitation, we did not finish this.

The team overestimated the difficulty of translating the software implementation of the AES algorithm to Verilog modules. The team also underestimated the difficulty of using AXI4-Stream interface and software drivers of VDMA correctly. We would like to do more researches on how to use AXI4-Stream interface and VDMA in the early stage if we could start all over again.

2.2. Future Improvements

Several potential improvements include:

1. Improve the frame rate of the output encrypted/decrypted video stream.
2. Add an extra AXI-Lite interface to the AES_CTR IP to enable the user to set key and counter initial value through the software.
3. Randomize the counter value to improve the security strength.

3. Project Schedule

Table 3.1 Milestones from project proposal

Milestone #	Description
1	Set up a basic test bench for software realization of AES.
2	**Testbench demo** Demonstrate that the testbench is able to test HDMI IN/OUT module, as well as the overall system functionality.
3	Finish software version of AES encryption/decryption
4	Able to read and modify input video data from HDMI IN.
5	**Mid-Project Demo** Able to write the modified data and output to HDMI OUT. Start hardware

	implementation of AES encryption/decryption.
6	Finish hardware implementation of AES encryption/decryption.
7	**Final Demo** Everything should be done by the final demo.

Table 3.2 Actual weekly accomplishments

Milestone #	Description
1	Transformed HDMI demo project from Digilent and compiled it in Vivado version 2016.2. Connected a video source (a laptop through HDMI IN port) to the board. Successfully displayed the source video stream on the screen successfully through the HDMI OUT port.
2	The software implementation of AES algorithm in CTR mode was completed. The associated testbench was completed as well.
3	**Testbench demo** The testbench for hardware version of AES core was completed (without AXI4-Stream interface).
4	Hardware implementation of the AES algorithm was completed.
5	**Mid-Project Demo** The FSM control logic is added to the AES module and the module is verified. A custom IP with AXI stream interfaces and a simple state machine (intended to delay each data transfer for 2 clock cycles) inside was created (the AES module would be instantiated inside of this IP latter). This IP core was connected to the whole system. However, the output data was not correct.

6	Tried to find bugs of the custom IP core. This issue was not resolved in this week.
7	**Final Demo** Changed the placement of the custom IP core. Added the second VDMA. Rewrote the data transfer logic of the custom IP core. Instantiated the AES module inside of this IP core. System integration was done. Built the software program to read/write frames from/to the custom IP core.

The team completed the HDMI IN and OUT modules earlier than what was expected because of the HDMI demo project posted by Digilent. The hardware implementation of the AES algorithm was also finished earlier. The team stuck at the AXI4-Stream interface when creating the custom IP core to receive/transmit the video stream during Milestone 5 and 6. The overall system integration was barely finished in the last week.

4. Description of the Blocks

4.1 Reference Design IPs

The reference design (see section 1.5) used by the team includes the following IPs:

1. DVI to RGB, RGB to DVI
2. Video In to AXI4-Stream, AXI4-Stream to Video Out
3. VDMA_0
4. MicroBlaze
5. AXI Interconnect
6. Uartlite
7. BRAM
8. DDR memory interface generator
9. Video timing controller

The team keeps all of those IP in this design. IP #1 comes from DIGILENT. IP #2 to #9 come from Xilinx.

4.2 VDMA_1

The VDMA_1 was created from the Xilinx library with a few adjustments on its control logic. The control logic of the VDMA_1 was written in the video_demo.c file. Each read or write operation was performed in the sequence given by axivdma_v6_1 documentation: set up a DMA configuration operation, set up the DMA buffer, and kick off the DMA operation. To display both the encryption and decryption, the write buffer address of the encrypted frame is set to frame1, and the write buffer address of decrypted frame is set to frame2. By switching the frame the display is parking on, the plain video, the encrypted video pixels, and the decrypted video pixels were shown respectively. Since the AES algorithm in CTR mode utilizes the same scheme for encryption and decryption, the decrypted video was obtained by performing a DMA write operation to frame2, followed by a read operation from frame1, at which the encrypted video is set to. This fed the output encrypted video pixels back to the AES_CTR IP and the output of the IP were the decrypted video pixels.

4.3 AES_CTR

The AES_CTR is the custom IP core created by the team that contains the AES in CTR mode as well as AXI4-Stream master and slave interfaces to communicate with the VDMA_1. The AES module works both for encryption and decryption. The designed AES module satisfied AES128-CTR standards published by NIST in 2001 [5] and was tested using test vectors proposed by NIST [6]. The stream width (width of tdata) is 24-bit for both interfaces. Other AXI4-Stream interface signals include tvalid, tready, tuser, tlast and tkeep.

Inside of the IP, there are six FIFOs, three for storing input stream data (input FIFOs), three for storing the encrypted/decrypted stream data (output FIFOs). The input video is 1080p, which means there are 1920*1080 pixels in total for one frame. The incoming data stream is 24-bit wide, represents a pixel of the frame, where each 8-bit data represents red, green or blue data for a single pixel. Therefore, the FIFO is designed with 1920-bit depth and 8-bits width (defined by parameters in the module), so that every row of a frame (1920 pixels) is

stored in three FIFOs where each FIFO stores one of the three colors (RBG) of all incoming pixels. The other three FIFOs are of the same size but are used to store encrypted/decrypted data.

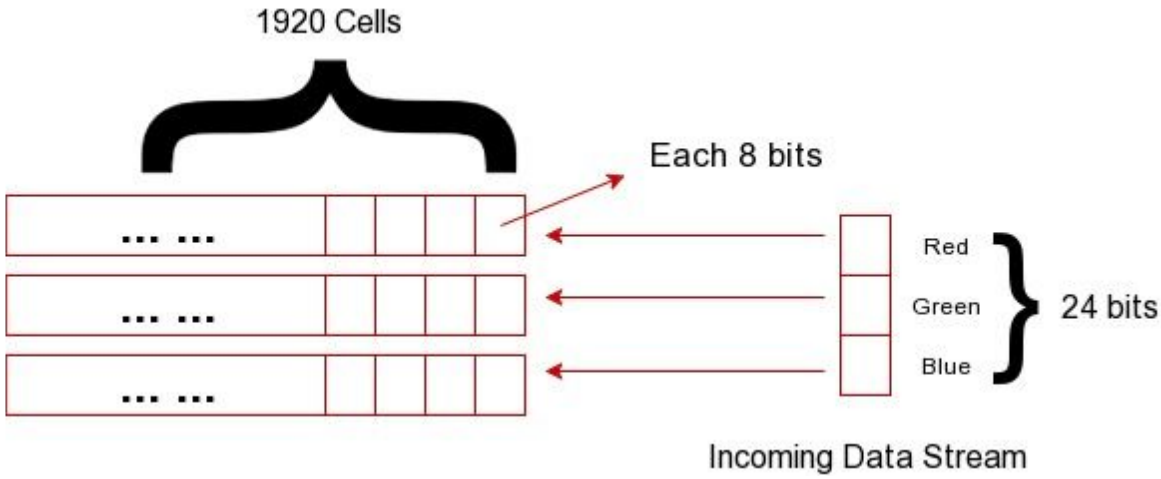

Figure 4.3.1 Data Receiving Diagram

The logic in this custom IP is controlled by an FSM which contains three states, storing state, store input data into input FIFOs; AES state, encrypt or decrypt data; output state, get processed data and store it into output FIFO. The custom IP starts the encryption/decryption process by waiting for the stream data to fulfill the three FIFOs. Once those FIFOs are filled (1920 * 8 * 3 bits, one line of pixel data in 1080p), all of the data are feed into 360 instances of the AES module in at the same time. Since each instance takes 128 bits data(the longest length for AES module) at once, so the total number of AES instances are 1920 * 8 * 3 / 128, which equals to 360. After all the AES instances have completed their jobs, three other FIFOs (output FIFOs) with the same size (1920 bits deep and 8 bits wide) will store these new processed data. The IP goes into the encryption state, where all AES instances are enabled, if and only if the input FIFOs are filled and the output FIFOs are empty. Both the slave interface and master interface stop receiving/sending stream data while at the encryption state. Otherwise, they are free to receive or send stream data if the input FIFOs are not filled or the output FIFOs are not empty respectively.

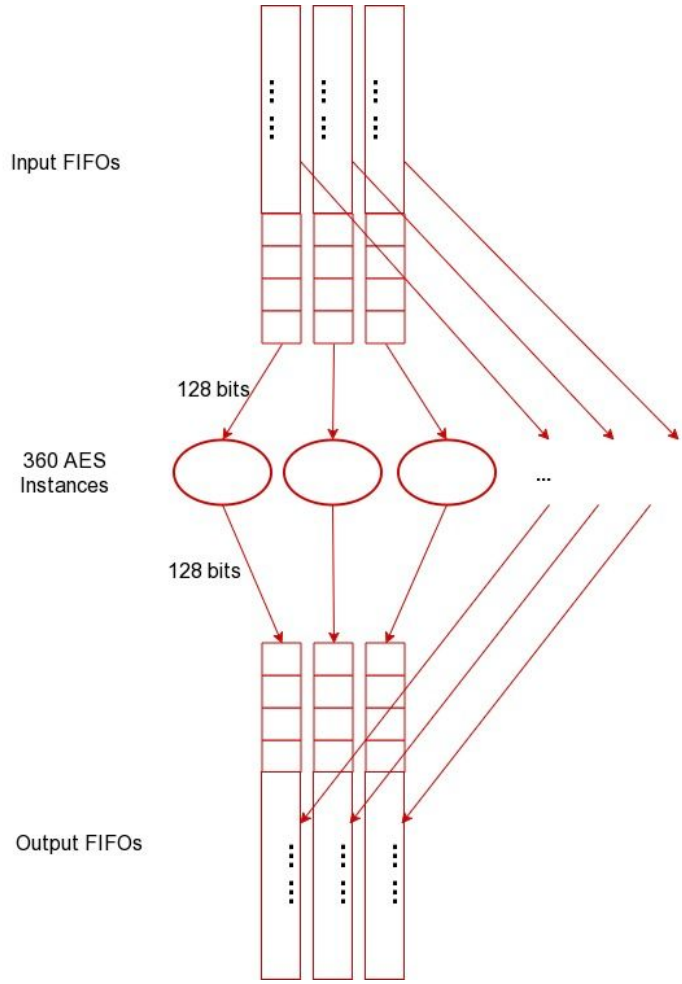

Figure 4.3.2 AES Data Transfer Diagram

The state machine also controls the read/write pointer for FIFOs, as well as other AXI4-Stream interface signals. The read/write pointer represents where the data should be read/stored from/into the FIFO. The tready signal of the slave interface represents if the IP is ready to receive data (input FIFOs have empty space). The tvalid signal of the master interface represents if the output data are valid. The tuser and tlast signals of the master interface represent if the current pixel data is the first one in a frame or the last one in a line respectively. The tkeep is simply assigned to constant value 1. The behavior of those interface signals follows the AXI4-Stream interface Video protocol as defined in the Video IP: AXI Feature Adoption section of the AXI Reference Guide (UG 1037, page 94) [4].

The AES encryption algorithm uses a round function that is composed of four different byte-oriented transformations, which generates ciphered data afterward. The CTR mode

uses an extra 128-bit counter as another input, which will be ciphered by the four transformations. The ciphered data generates after XOR with the counter. The decryption algorithm has the same steps, but its inputs are the ciphered data and the same counters.

In this design, the key values and the counter values for all AES instances are constant. All of the AES instances have the same key value but different counter values. By the nature of the AES algorithm in CTR mode, this custom IP is capable of decrypting the video stream by just taking the encrypted video stream as the input, as long as the key values and counter values remain the same as in the encryption process.

5. Description of the Design Tree

src: the vivado project

- Proj
    - AES_module: ready-to-use AES128-CTR module
    - hdmi.xpr: main project file
    - Hdmi.sdk/
        - demo/src
            - video_demo.c: main file contains all functions to run the design
            - vdma
                - vdma.c: contains functions to configure the second VDMA (VDMA_1)
- repo: custom AES IP
    - AES_CTR_1.0/hdl: the AXI-stream IP contains AES encryption & decryption modules
        - AES_CTR_v1_0.v
        - AES_CTR_v1_0_M00_AXIS.v
        - AES_CTR_v1_0_S00_AXIS.v
        - aes_top.v

- src: design constraints specification
    - constraints: the design constraint file
        - NexysVideo_Master.xdc

doc: group final report and final demo presentation slides

- Group Final Report
- Final Demo Presentation

6. Tips and Tricks

The design took up a large volume of hardware resources on board, and it took the Vivado tool a significant amount of time to complete one design run including running synthesis and implementation. Simulations are strongly recommended though developing testbench that covers most of the cases is not easy. Adding debug cores also enable run-time data and control analysis, but it requires a design run being completed in the first place. The software driver of VDMA is also a critical part of the design, which is also time-consuming without an appropriate use of the VDMA. Past similar projects and designs can provide guidance and ready-to-use IPs/algorithms/APIs and would definitely reduce development time.

7. References

[1] Margaret. Rouse, What is Advanced Encryption Standard (AES)?, SearchSecurity [Online]. Accessed April 7, 2017. Available: http://searchsecurity.techtarget.com/definition/Advanced-Encryption-Standard

[2] *Block cipher mode of operation*, Wikipedia[Online]. Accessed April 7, 2017. Available: https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

[3] *Digilent/Nexys-Video-HDMI,* Github[Online]. Accessed April 7, 2017. Available: https://github.com/Digilent/Nexys-Video-HDMI

[4] *Vivado Design Suite, Vivado AXI Reference (June 24, 2015),* Xilinx[Online]. Accessed April 12,2017.Available: https://www.xilinx.com/support/documentation/ip_documentation/axi_ref_guide/latest/ug1037-vivado-axi-reference-guide.pdf

[5] "Advanced encryption standard (AES)", 2001.

[6] M. Dworkin, "Recommendation for block cipher modes of operation :", 2010.

[7] S. KOTEL, M. ZEGHID, A. BAGANNE, T. SAIDANI, Y. I. DARADKEH and T. RACHED, “FPGA-Based Real-Time Implementation of AES Algorithm for Video Encryption,” *Recent Advances In Telecommunications, Informatics And Educational Technologies*, pp. 27-36, 2014.