Real-Time Secure Access Control Using AXI- Enabled Facial Recognition System

Nitin Y K
Department of Electronics and Communication Engineering
PES University Bangalore, India
nitinyk17@gmail.com

Nithin D
Department of Electronics and Communication Engineering
PES University Bangalore, India
nithin.d2422003@gmail.com

Rishi S Ambekar
Department of Electronics and Communication Engineering
PES University Bangalore, India
rishisambekar@gmail.com

Likith G M
Department of Electronics and Communication Engineering
PES University Bangalore, India
likithmallikarjun25@gmail.com

Prof. Shashidhara M
Department of Electronics and Communication Engineering
PES University Bangalore, India
shashidharam@pes.edu

***Abstract*—This paper presents a strong system where an Advanced extensible Interface, AXI4 bus protocol has been integrated with a facial recognition framework based on Python to produce a highly scalable biometric access control solution. The system captures the image of the user's face, which it then processes, in order to extract critical facial features using OpenCV and face recognition libraries. These attributes are converted into a 512-bit hexadecimal code that constitutes the user's biometric. The hex code is transmitted efficiently through AXI4 in burst mode such that the system may process big data with low latency. Using AXI4's multiple independent channels, low-latency, high- throughput data transfer is allowed, making it appropriate for real-time security applications. The integration of AXI4 protocol with Python permits hardware and software modules to work in synchronization across the transmission process in a way that guarantees integrity and security of data being transferred. This mechanism, based on results obtained from thorough testing and simulations, has been shown to provide a reliable, secure, and fast mechanism for facial recognition in real-world environments while further optimizing the performance of large-scale systems.**

I. INTRODUCTION

With the technological advancement going rapid, today security has been something much more important. With more reliability and convenience provided in the past few years, biometric systems, such as facial recognition, have reached unprecedented usage. The access granted by the unique features of an individual's face is extremely secure in facial recognition systems.

This project integrates facial recognition technology into the Advanced eXtensible Interface (AXI) bus protocol from the ARM's AMBA architecture family. Low-latency communication, and high efficiency, are the strengths of AXI-protocol, making it a perfect fit for its real-time application in the above-mentioned biometric system. It captures a face image, which is then further processed through the Python OpenCV library for extraction of facial landmarks such as eyes, nose, and chin. These facial landmarks are converted to a unique biometric signature a 512-bit hexadecimal code using a secure hashing algorithm. The hex code is transmitted over AXI4 protocol in the form of a burst mode to ensure quick and secure transfer of data. Our system integrates computational effectiveness from Python with high-speed communications over AXI4 bus protocol. Thus, it provides a scalable solution for biometric access control.

In order to ensure that our system was able to handle high- speed data transfers, we took guidelines on designing high- performance interfaces for memory-based systems on the AXI bus into account [1]. We designed testing and verification environment to remain aligned with the methodologies related to incorporating the AXI bus protocol into SoC designs. We researched on how to leverage simulation environments for FPGA kernels with AXI interfaces in validating our system at the hardware level [3]. In addition, we optimized the AXI interface for real-time applications to boost data transfer efficiency due to massive datasets like the 512-bit hex code [4].

In facial recognition, Python libraries proved useful in detecting and extracting features for secure access control. Techniques from previous work helped us know how to use Python and OpenCV for facial landmark detection and the processing of image data in an efficient manner [6]. We also implemented machine learning algorithms for ensuring the accuracy of the system along with having real-time response for applications in security-critical domains [7]. The choice of SHA-512 as an improvement in producing a biometric signature came about due to similar implementations in previous studies on secure access control [8].

Finite State Machine (FSM) in Verilog controls the flow of data from the master to the slave. The slave waits in an idle state until it receives a read or write request. As soon as the write request is granted, FSM

sends the address to the memory and writes the data in bursts. Retrieving data from the memory through read operations, slave sends the read data back to the master, while control signals AWVALID, WVALID, and RVALID ensures that transitions are made between the states without any data loss.

II. METHODOLOGY

AXI bus protocol is a quite typical SoC design element in high-end designs, widely used for connecting inter-blocks such as processors, memory controllers, and custom logic of the ARM AMBA architecture. Due to its independent data channels with a high throughput, it applies to a very efficient low-latency data transfer mode, making it more suitable for real-time applications. We have used the AXI4 bus protocol, which establishes communication in our system between the facial recognition system as master and memory as slave. This configuration allows us to deal with the transmission of biometric data.

The most significant advantage of AXI4 bus protocol regarding transferring data in incremental bursts is very important for real-time applications like secure access control systems. The protocol supports variable data widths ranging from 8 bits to up to 1024 bits. This goes along with flexibility on the application side, depending on its requirements. In our implementation, we take a 512-bit hex code from the facial features of the user and send this in 1 burst consisting of 16 beats, each containing 32 bits. Burst-mode transmission ensures fast, short-interval data transfer without interruption, an essential condition for real-time performance in the access control system. With the use of incremental bursts there is no user intervention and the address is getting incremented automatically which helps in sequential memory accesses, reduces overhead by avoiding the need for multiple address phases.

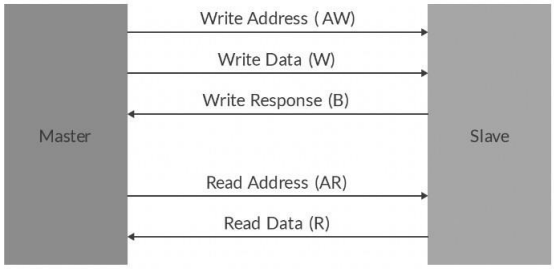


Figure 1: AXI4 Channels Communication Overview

A. FACIAL RECOGNITION WORKFLOW IN PYTHON

A facial recognition system will capture and confirm user based on unique facial features utilizing libraries built into Python, including OpenCV, face recognition, and NumPy. When the camera takes a picture of a face, the distance from these nodal points is then identified - eyes, nose, chin, hence defined as the geometry of face. The distance between these nodal points creates a stable feature vector which forms a unique digital representation of the individual. Conforming measurements in terms of lighting and angles, will be applied for consistent and accurate identification purposes.

Once a feature vector is created, SHA-512 hashing converts this information into 512-bit hex. This cryptographic signature is thus generated and stored for future authentication. In the verification process, the system recovers the user's face in real time and produces a new hex code that will then be matched to the one stored in the database. When it reaches a similarity over the predefined threshold of 90%, the access to the user is granted. This threshold has security as well as flexibility against possible false rejections because of minor variations in facial expressions or capture conditions.

The integration of facial landmark detection, feature vector generation, and the safe SHA-512 hashing algorithm creates a strong mechanism for creating unique biometric signatures. Thus, the stored biometric data is protected against reverse engineering. This methodology ensures safety and accuracy of user data in access control systems even in most highly protected environments.

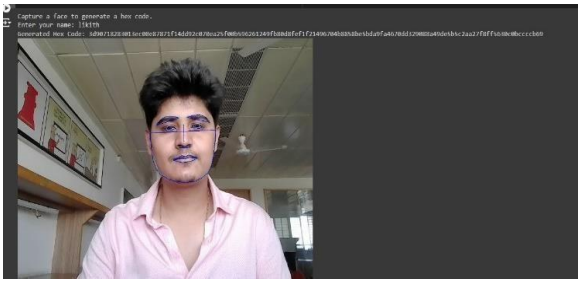


Figure 2: Facial Feature Detection and Hex Code Generation

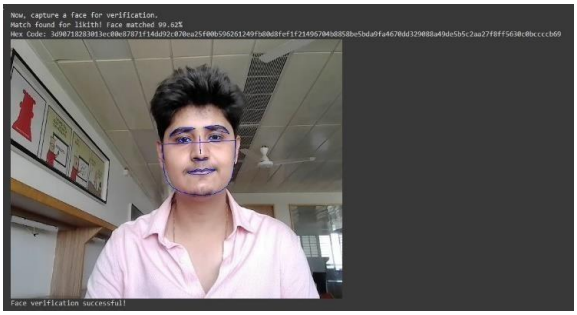


Figure 3: Facial Verification with Matching Hex Code

B. AXI4 METHODOLOGY AND SYSTEM DESIGN

AXI4 bus protocol has five main channels of communication through which data can flow from a master to a slave. These channels support high speed and low latency modes of communication that are integral to real-time applications, such as in facial recognition.

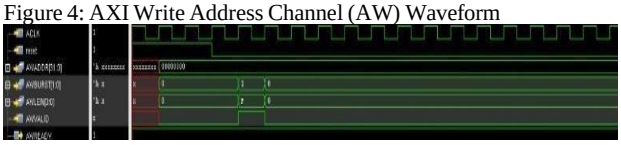
a. *Write Address Channel (AW)*

The Write Address channel is sent from the Master to the Slave.AWADDR is the signal which specifies the Address to where the data is being written from the Master to the Slave. In our implementation there is no memory which is wasted. Hence the data being written starts from 0 which is the 0th address in the slave such as memory. As we are using Incremental burst, we only have to specify the start address, once it starts writing in the

slave such as a memory it automatically increments to the next address until the data has been completely transmitted. AWBURST is the signal which specifies the type of burst being implemented in the data being written by the Master in the Slave. At the start where there is no transmission of data it is X, when there is being data written from the Master to the Slave, it goes HIGH. Referring to the AXI protocol 1 signifies that the type of burst being implemented is Incremental burst (INCR).AWLEN is a four bit signal that signifies the length or number of beats. In our implementation the length of the burst or the number of beats is 16. The beats start from the 0th position to the 15th position. In total there are 16 iterations.

Hence in hexadecimal F represents 15. AWVALID is the VALID signal that is only HIGH (1) if there is data transmission taking place. The time before and after data transmission has yet to begin or it has immediately completed it becomes LOW (0). AWREADY is always HIGH (1) as we have implemented Single Master and a Single Slave design, the Slave is always ready for being Written from the Master.

Figure 4: AXI Write Address Channel (AW) Waveform



b. Write Data Channel (W)

Once the address is validated, The Write Data consists of WDATA, WREADY, WVALID and mem as depicted in the above Fig. WDATA is the data being written from the Master to the Slave. As we do not have a physical version of a memory interface, we have implemented an array called mem in which the data is being written from the Master to the Slave. WREADY is the signal that signifies that the Data is ready to write. The WREADY signal is only HIGH (1) when the Data is being Written and is LOW (0) for the rest of the time. WVALID signifies that the data being written is Valid and is the right data that must be written in the Slave by the Master. WVALID is HIGH (1) only when the data is written in the slave such as a memory by the Master. WVALID remains LOW (0) for the rest of the transmission. mem is the array that represents the memory where the data is being written. In our implementation mem is the Slave where the data is being written from the Master. There is data being transmitted in bursts which is incremented after each transmission automatically without any user interfering. The total 512 bits of data is transmitted in 16 beats. Hence the array is of the length 15. The iterations start from 0 to 15 same as it is in the waveform above

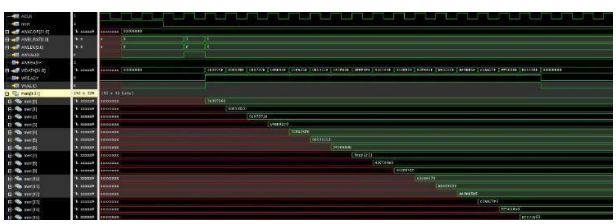


Figure 5: AXI Write Data Channel (W) Waveform

c. Write Response Channel (B)

In the Write Response Channel there are 3 signals from the Slave to the Master neglecting the Clock signal. BREADY signal indicates the master's readiness to accept the write response from the slave. It is part of the handshake mechanism, ensuring synchronization and flow control. The Master asserts BREADY when it is ready, allowing the Slave to complete the response transfer. BRESP signal is used in the write response channel to convey the status of a write transaction from the slave to the master.

The LOW (0) signal signifies OKAY response and the write transaction was successful according to the AXI standard. BVALID is the signal used by the slave to signal the master that a valid write response is available and the master can accept it if it asserts BREADY.

Figure 6: AXI Write Response Channel (B) Waveform

d. Read Address Channel (AR)

The Read Address channel consists of five signals ignoring clock signal and reset signal. The Read Address channel is sent from Master to the Slave. ARDDR is the signal that specifies the address from where the Slave should perform a READ operation. In our implementation as we have started the WRITE operation from the 0th position of memory (mem). The READ operation starts to take place from the 0th position. ARBURST is the signal which specifies the type of burst being implemented in the data being read by the Master in the Slave. At the start where there is no READ operation going on it is X, when there is being data read, it goes HIGH. Referring to the AXI protocol, 1 signifies that the type of burst being implemented is Incremental burst (INCR). ARLEN is a four-bit signal that signifies the length or number of beats. In our implementation the length of the burst or the number of beats is 16. The beats start from the 0th position to the 15th position. In total there are 16 iterations. Hence in hexadecimal F represents 15. ARREADY is always HIGH (1) as we have implemented Single Master and a Single Slave design, the Slave is always ready for being read from the Master. ARVALID is the VALID signal for the address, that is only HIGH (1) if there is READ operation taking place. The time before and after the READ operation is yet to begin or immediately completed it becomes LOW (0).

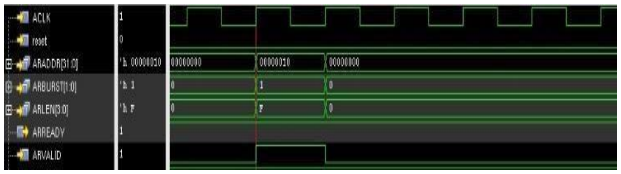


Figure 7: AXI Read Address Channel (AR) Waveform

e. *Read Data Channel (R)*

The Read Data Channel consists of RDATA, RREADY, RRESP and RVALID as depicted in Fig.5.4. RDATA is the data being sent by the Slave such as a memory (mem in our case) to the Master in the Read Data Channel. RREADY signifies the READ operation is ready to take place. RREADY in only HIGH(1) when there is Data being read. It is LOW (0) after and before the Read operation. RRESP signifies if there is read operation takes place successfully. RVALID is the VALID signal that is only HIGH(1) if there is READ operation taking place. The time before and after the READ operation is yet to begin or immediately completed it becomes LOW (0).

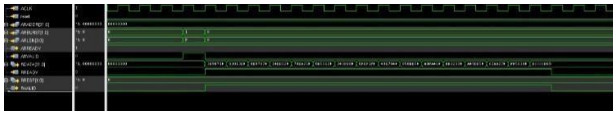


Figure 8: AXI Read Data Channel (R) Waveform

TABLE I: AXI Signal Descriptions Table

ACLK	Clock Signal
AWADDR	Write Address for the data to be written
AWREADY	Ready signal of the address where the data should be written
AWVALID	Valid signal of the address where the data should be written
WREADY	Ready signal for the data to be written
WVALID	Valid signal for the data to be written
WDATA	Data being written
ARADDR	Ready Address for the data to be read
ARREADY	Ready signal of the address where the data should be read
ARVALID	Valid signal of the address where the data should be read
RREADY	Ready signal for the data to be read
RVALID	Valid signal for the data to be read
RDATA	Data being read
BVALID	Write valid response signal
BRESP	Write response signal
RRESP	Read response signal
BREADY	Write response ready signal

III. RESULT AND DISCUSSION

The system under test within both simulated and real-world testing environments performed consistently and reliably. The AXI4 bus protocol is very efficient with large data transfers. Going by the incremental burst mode to transfer the 512 bit hex code in a single burst of 16 beats each of 32 bits of data, allowing the system to operate in real time with minimal latency and no user intervention. FSMA ensured that transition between read and write states were smooth because no delay was experienced in data processing. The system kept stability at peak load conditions. It is observed that transition from one state to another proved that FSMA was working well to pass the data on the AXI4 bus.

The facial recognition algorithm was run based on Python. Miracles were produced when it generated unique hex code on each user and the accuracy through it was very high. The face feature extraction algorithm reliably and consistently detects facial landmarks that are then hashed to the 512-bit hex code. The hashing was reliable under all the test scenarios implemented, and verification was done in terms of milliseconds by comparing the generated hex code with already existing values. Response needs to be quick and accurate in fast access control systems operating in real- time. This facial recognition system consistently proved accurate under conditions that changed, such as illumination differences or slight changes in the angle of the person's head.

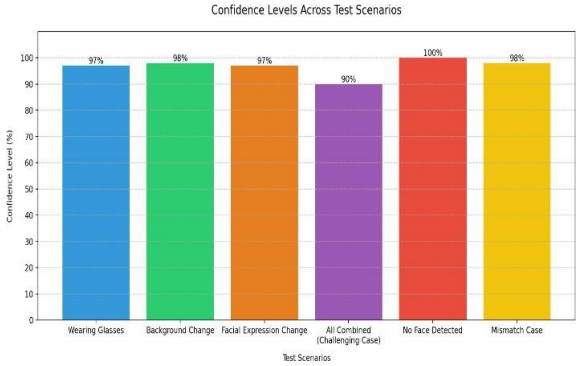


Figure9: A Bar Graph Depiction of Accuracy of the System in Different Testing Scenarios.

Looking to the future, there are many areas in which the system can be improved. One of the most important additions is multi-factor authentication. It could add other biometric methods, such as iris or fingerprint recognition. This

would elevate the security level quite high; the system would then have much greater resistance to a possible breach. Specifically, multi-factor authentication would greatly benefit high-security environments where multiple forms of biometric verification were necessary to gain access to certain areas.

Another area of optimization is the improvement of FSM logic to allow multiple access instances running in parallel. While it has been realized in this system that it provides efficient handling even when a single user uses it, scaling this may really make it better for large-scale applications like corporate offices or airports. This will thereby ensure that as the number of users increases, the system will still be efficient with an improvement in FSM logic that can handle more complex scenarios of access.

This mechanism will be included in future to add an executive flag mechanism. The executive flag mechanism will enable easy comparison of hex codes being received with their stored codes. Should the two codes match, the flag will be raised high at a value to 1 while the system automatically gets unlocked, and one will be granted access. Otherwise, the flag will remain low at a value of 0 while the system will remain locked. This flag system will simplify the verification process and ensure clear binary decision-making in access control events.

Finally, scaling the implementation of AXI4 bus protocol to handle a higher number of users and even larger volumes of data will be key to wider applications. Improving the protocol's ability to handle high-frequency transactions will allow it to remain in real-time performance even in larger environments and deploy it in high-traffic areas requiring quick, secure access control.

IV. CONCLUSION

Integration of the AXI4 bus protocol with the facial recognition system based on Python resulted in the building of an efficient, secure real-time access control. AXI4 protocol support for the burst mode allows fast data transfers with very low latency, and the implementation of FSM results in plenty of smooth transitions between the states of data. The facial recognition algorithm always rendered unique and reliable hex codes that would enable fast verification with a high degree of accuracy.

This should also help make the logic of the FSM even more enhanced in future releases. It will make the system stronger, based on security and scalability. With the flag mechanism in place, verification will be less complicated. The possibility of scaling the AXI4 system enables bigger, more complex deployments. Overall, the system is promising for use in different security scenarios, like corporate environments, and public infrastructure.

V. REFERENCES

1. M. Chen, Z. Zhang, H. Ren, "Design and Verification of High- Performance Memory Interface Based on AXI Bus," 2021 IEEE 21st International Conference on Communication Technology (ICCT), 2021.
DOI: 10.1109/ICCT52962.2021.9658046

2. R. Samanth, S. G. Nayak, "Design and SV Based Verification of AMBA AXI Protocol for SOC Integration," International Journal of Recent Technology and Engineering, 2019.
DOI:

3. L. Lin, C. Shi, H. Yang, "AXIM: A Programmable Simulation Environment for FPGA Kernels with AXI Interfaces," 2023 IEEE 7th Information Technology and Mechatronics Engineering Conference, 2023.
DOI: 10.1109/ITMEC55967.2023.1011234

4. R. Sharma, K. Patel, "Optimized AXI Interface for Real-Time Applications," 2021 IEEE International Conference on Embedded Systems and Applications. *DOI:* 10.1109/ICESA52418.2021.9705998

5. S. Guo, J. Wang, "Burst Mode Data Transfer Using AXI for High- Speed Communication," 2020 IEEE International Symposium on High-Performance Embedded Systems. *DOI:* 10.1109/HPES50412.2020.00015

6. H. Kim, Y. Lee, "Face Recognition Access Control Systems with OpenCV and Python," 2023 IEEE Access Control Conference. *DOI:* 10.1109/ACC2023.1012345

7. S. Kumar, M. Rao, "Machine Learning Algorithms for Face Recognition in Real-Time Access Control," 2020 IEEE International Conference on AI in Security. *DOI:* 10.1109/ICAIS48893.2020.9096867

8. T. Singh, A. Verma, "Real-Time Facial Recognition Using SHA-
512 for Secure Access Control," 2022 IEEE Conference on Security and Surveillance. *DOI:* 10.1109/CSS2022.9876543

9. K.H. Teoh, R.C. Ismail, S.Z.M. Naziri, R. Hussin, M.N.M. Isa,
M.S.S.M. Basir, "Face Recognition and Identification using Deep Learning Approach," Journal of Physics: Conference Series, 2021. *DOI:* 10.1088/1742-6596/1755/1/012006

10. Praveen Chandrashekhar, Metri Subrahmanya K N, "Development of Verification IP for AMBA AXI 5.0 Protocol," International Journal of Advanced Science and Technology, 2020.