Turkish Online Journal of Qualitative Inquiry (TOJQI)
Volume 12, Issue 10, October 2021: 1041-1050

High Performance Pipelined AES Implementation with AMBA AHB Interface for SoC's

[1] Mohammed Murtuza Mohiuddin, [2] Shaik Mohammed Rasool, [3] Dr. Mohammed Jabirullah

[1]PG Scholar, Department of ECE, Lords Institute of Engineering & Technology, Hyderabad, India
[2]Assistant Professor, Department of ECE, Lords Institute of Engineering & Technology, Hyderabad, India
[3]Associate Professor & HOD, Department of ECE, Lords Institute of Engineering & Technology, Hyderabad, India

Emails: [1]murtuzaece15@gmail.com, [2]skmohammedrasool@lords.ac.in, [3]drjabirullah@lords.ac.in

Abstract— Three alternative AES architectural designs will be presented in this study with the goal of establishing a symmetric block ciphering standard. Pipe structures and resource sharing were used in the development of the three concepts. The open standard AMBA AHB governs how blocks in a SoC are interconnected (SoC).

***Keywords*— Advanced High-performance Bus (AHB), Advanced Encryption Standard (AES).**

I. INTRODUCTION

The Advanced Encryption Standard was created in 2000 as a result of work done by the NIST (AES). The open competition for algorithm selection was won by Rijndael, a substitution-permutation block cypher. AES was first developed by the US federal government to protect sensitive data before becoming a global de facto standard. AES's flexibility to a broad variety of hardware and software settings is one of its most attractive features.

In this article, three different AES encryption schemes are examined. They may be integrated into more complex systems thanks to a common communication protocol, and they provide answers for many problems. To put it another way, it works like this: During encryption and key expansion, the AES algorithm adapts to all of the changes that occur. Section III discusses the techniques used to design individual architectural projects.

II. DESCRIPTION OF THE AES ALGORITHM

However, secret key sizes are most often found at 128, 192, or 256-bit levels, with AES using 128-bit data blocks. Figure 1 depicts the encryption process' iterated use of invertible transformations, often known as rounds. The first round is followed by iterations of the Nr-1 general round, and the final round has just one iteration. A 128-bit key has 10 round iterations; a 192-bit has twelve round iterations; and a 256-bit key has fourteen round iterations. The secret key must be used to generate a sub-key (likewise known as a overweight key) using an expansion algorithm for each round. There is XOR operation in the first round, whereas general rounds contain four alterations called SubBytes (SB), MixColumns (MC), Shiftrows (SR) and AddRoundkey (A) between the plain text block and the first round key (ARK). When it comes to the concluding round, it's comparable to other circles excluding that the MixColumns transformation is absent. It's omitted from the list. As a consequence of the intermediate results during encryption, all input and output data blocks are also encoded as a 4*4 matrix known as a state matrix, which corresponds to the sixteen bytes fashionable a 128-bit data. To learn more about different kinds of encryption, keep reading.

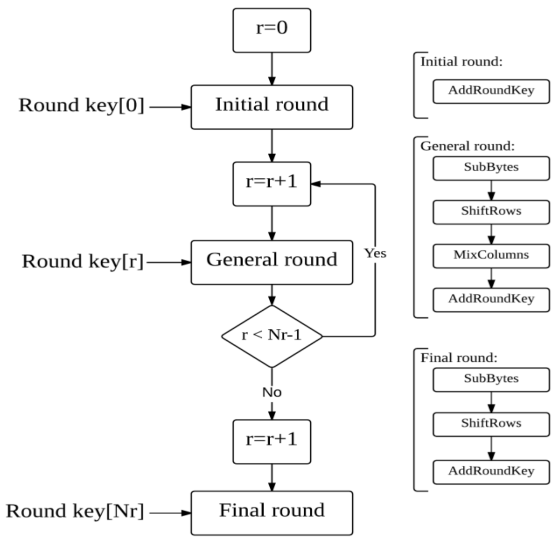


Fig.1: Diagram of the AES encryption algorithm

A. SubBytes

In the AES standard, there is just one nonlinear operation, and that is the algorithm's replacement layer transformation. A change is applied to each member of the state matrix (referred to as S-BOX). Galois Finite Field (GFf28g) S-BOX inversion using irreducible polynomial x8+x4+x3+x+1 and affine mapping. I.eno byte maps to either itself or its own reverse value. As a result, the transformation becomes more complicated while also avoiding fixed and opposing places. The S-BOX transformation is often carried out using lookup tables or arithmetic operations in finite fields (LUTs).

B. ShiftRows

Changing the rows in the state matrix causes this change. All succeeding rows are rotated one byte to the left, even if the first row remains unchanged. As a consequence, the last row is three bytes to the left of where it originally was.

C. MixColumns

The state matrix's columns may be seen as discrete units of change. In mathematics, this is equivalent to multiplying polynomials over the finite field GFf28g. Each column is represented by polynomial coefficients multiplied by the constant polynomial 03x3+01x2+01x+02, which is known as the irreducible polynomial x4+1 in the standard. Equation 1 may demonstrate this for a single column. A superscript 0 is appended to a state matrix entry after a change (zero).

$$\begin{bmatrix} S'_{0,j} \\ S'_{1,j} \\ S'_{2,j} \\ S'_{3,j} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \times \begin{bmatrix} S_{0,j} \\ S_{1,j} \\ S_{2,j} \\ S_{3,j} \end{bmatrix}$$

D. Key Expansion

Encryption sub-keys are generated via the use of a recursive approach. Depending on key size, this technique is fast or slow. A 128-bit secret key's primary sub-key and key are the same. The final sub-keywords are calculated using Equation 2, as stated above. For example, using a similar S-BOX for encryption, you may execute circular shift followed by replacements, and then perform an XOR process amongst the maximum substantial bit and an 8-bit continual named RC to keep the value changing every time the programme is performed in a loop.

W[4i] = W[4(i- 1)] + g(W[4i - 1]) (2)

W[4i + j] = W[4i + j - 1] +W[4(i- 1) + j]   (3)

Regardless of the development technique, nearby are two methods near handle overweight keys. Popular the principal method, referred to as precompute, all of the subkeys are generated before any encoding takes place. On the other hand, a dynamic method uses only when they are really needed to generate sub-keys. There is no requirement for parallelism inside the encryption process with the designs in this study since they use a 128-bit secret key and a precompute method.

E. Procedure modes

Plain text delivery to the cypher is controlled by operating modes when the block size is exceeded. NIST's full recommendations describe the most commonly utilised operating modes. Codebook (ECB), Cipher (CFB), OFB, and Counter are all acronyms that should be familiarised with (CTR) ECB mode is used in these systems, which means that each segment of the plain text is encrypted separately, according to this study. Because it keeps the plain text statistics, this mode of operation is insecure when it comes to encryption. Because it's not a feedback mode, we may encrypt a huge number of data blocks at once as a foundation for both opposing modes.

III. DESIGN

This study's designs include three major components: key extension, encryption, and control. The standard's expansion method turns a square key into a round one. Encoding begins as soon as key expansion is complete, thanks to a fsm that controls communication between blocks involved in key expansion and encryption. An output text block notifies the control module that a replacement has arrived, and the output data is now legitimate. . The three designs are referred to as "basic," "pipeline," and "compact," respectively. It's important to note that their blocks are implemented in very different ways.

A. Basic architecture

All of the 128-bit wide internal buses are cycled once every clock cycle. Cryptographic blocks are constructed using the architecture shown in Figure 2, with the result being transmitted back via a register as an input after one general round. To implement AddRoundkey, each byte has its own LUT instance, and each word has its own MixColumns instance. There are a total of 32 LUTs used in this round. The addressing between the SubBytes output and the MixColumns input determines the ShiftRows transformation. There's a multiplexer just before the final match.

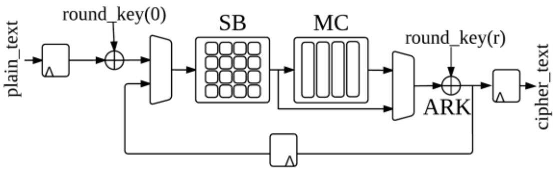


Fig.2: Implementation of an encryption block in the underlying architecture

Since the sub-keys have already been calculated, the key may be expanded more quickly. Basic design compromises between size and performance due to key development block's one sub-key calculation per clock cycle.. The final result is revealed in Fig. 3. Send the encryption block the keys first, and it will store them in an array.

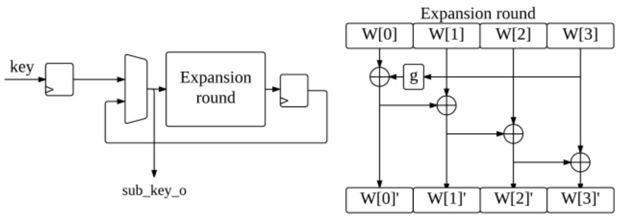


Fig.3: Implementation of the key expansion in the fundamental architecture

B. Pipeline construction

This design's primary objective is to outperform the core module in terms of throughput. The only way you're going to go this far is by going through a lot of rounds at once. Figure 4 shows an encryption block, thus there are a total of nine general rounds and a final round. Four AddRoundkey instances are used in addition to four MixColumns instances for each general round. By using registers to connect the rounds, 10 data blocks may be processed in the same amount of time (fully-pipelined structure). Every clock cycle, a new cypher text block will be produced using the finalised design's basic structure. There is a striking resemblance between this design's primary expansion block and a building block from classical architecture.

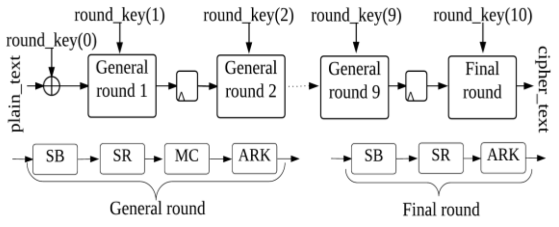


Fig.4: Implementation of an encryption block in a pipeline architectural design

C. Compact architecture

Compact architecture is being utilised to minimise the amount of hardware needed to encrypt data. It is possible to fit a single 128-bit block inside a module by splitting it into four 32-bit blocks. The encrypted transitional consequences are deposited in four memory blocks, one for each row of the national matrix. SubBytes and AddRoundkey instances, as well as a MixColumns, are shown in Figure 5 of the encryption block One bank of memory blocks stores the round input file, the other the round output file. Interpretation a expression since an input bank and dispensation it using combinatorial logic is how encryption is done in rounds. The results will be deposited in the production bank until the succeeding watch series. Before the next cycle begins, the input and output banks exchange places.

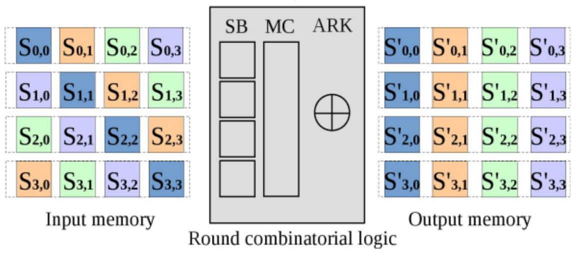

$S_{0,0}$ $S_{0,1}$ $S_{0,2}$ $S_{0,3}$
$S_{1,0}$ $S_{1,1}$ $S_{1,2}$ $S_{1,3}$
$S_{2,0}$ $S_{2,1}$ $S_{2,2}$ $S_{2,3}$
$S_{3,0}$ $S_{3,1}$ $S_{3,2}$ $S_{3,3}$
Input memory
SB MC ARK
Round combinatorial logic
$S'_{0,0}$ $S'_{0,1}$ $S'_{0,2}$ $S'_{0,3}$
$S'_{1,0}$ $S'_{1,1}$ $S'_{1,2}$ $S'_{1,3}$
$S'_{2,0}$ $S'_{2,1}$ $S'_{2,2}$ $S'_{2,3}$
$S'_{3,0}$ $S'_{3,1}$ $S'_{3,2}$ $S'_{3,3}$
Output memory

Fig.5: Compact implementation of the encryption block.

The proper addressing determines how the Shift Rows architecture transforms while reading from memory. A clock cycle's worth of memory reads and writes are shown in Fig. 5 through colour coding.
The 32-bit internal buses of the compact design's main expansion block are also included [9]. As shown in Figure 6, one sub-keyword is computed for each tick of the clock. For each computed word in the encryption block, round keys are kept in four memory blocks (one for every key matrix row). The AddRoundkey transformation allows instant access to these blocks.

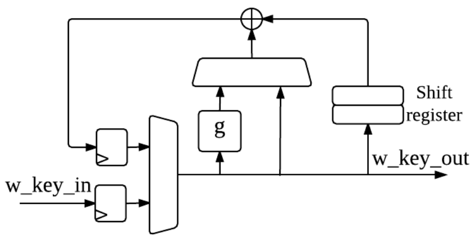


Fig.6: Compact architecture implements a key expansion.

IV. AMBA CROSSING POINT

For the AMBA AHB interface, wrappers were employed. This allowed it to be integrated into a variety of AES designs. Wrappers are used to offer a 32-bit datapath for the smallest and most basic systems (HRDATA and HWDATA buses in AMBA). Pipelines now feature buses with a bit depth of 128 bits. A clear text block may be transmitted to the cypher per clock cycle thanks to the AMBA AHB standard, which supports this bus width. A pipeline design's maximum throughput can never be reached with smaller data paths. Data (plain text and the secret key) is sent to the cypher by AHB Master via registers. Because information is transferred from a datapath to an AES-cypher after the least significant number of word registers are written with datapath wrapping, AHB Master must write registers in a certain order (from most to least important) while using datapath wrapping. In basic and compact architectures, the HREADYOUT signal is set to zero while encryption or key expansion is taking place to indicate that no new data will be received. HREADYOUT is set to a coffee value only while the key expansion procedure is in ongoing since the pipeline structure encrypts several bits of data at once. As soon as encryption is complete, the AMBA AHB interface is updated with wrappers for future AES designs. Basic and small systems may both benefit from 32-bit datapath wrappers (HRDATA and HWDATA buses in AMBA). Pipeline buses now have a bit depth of 128 bits rather than 64 bits. One clear text block may be sent to the cypher per clock cycle under the AMBA AHB standard utilising this bus width. If smaller data channels are utilised, the pipeline architecture's maximum throughput will never be reached. AHB Master utilises registers to send plain text and the secret key to the cypher. AHB Master must first set up the registers in a certain order before sending the data to AES (from most significant word to least). The HREADYOUT signal in basic and compact architectures is set to zero while encryption or key expansion processes are in progress, signalling that no further data will be received. A single value for HREADYOUT is only utilised while the key expansion procedure is in place since pipeline architecture encrypts several bits of data at once. There are three methods to save the cypher text once it has been encrypted: (four 32-bits registers for basic and compact and one 128-bits register for pipeline). Before creating a new cypher text block, the AHB Master should double-check these registers. AMBA In order to avoid data loss and maximise pipeline efficiency while using AHB Master and skimming Slaves at the same time, cypher text blocks must be preserved.

V. SIMULATION RESULTS

The new and enhanced architecture was simulated using Xilinx ISE 14.7. After the key expansion process is complete, the Master directs the underground key also the cypher gets the plane text blocks. At the end of the encryption development, the test bench compares the value returned by the cypher with the conforming cypher transcript in the file.

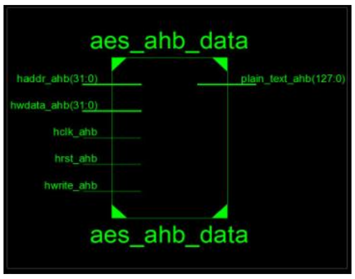


Fig.7: Showing Detailed RTL Schematic of AES AHB DATA

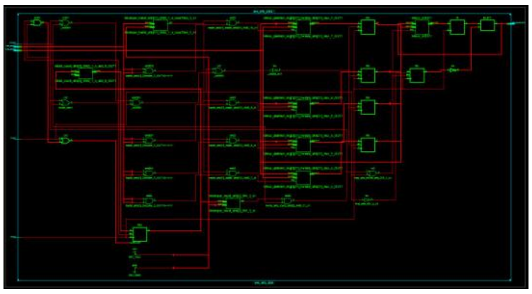


Fig.8: Showing RTL Schematic of AES AHB DATA

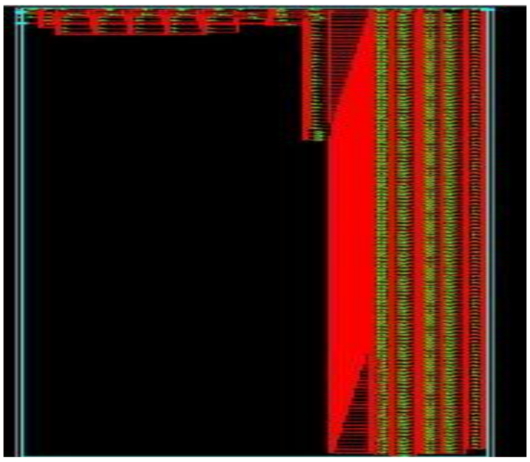


Fig.9: Showing Technology Schematic of AES AHB DATA

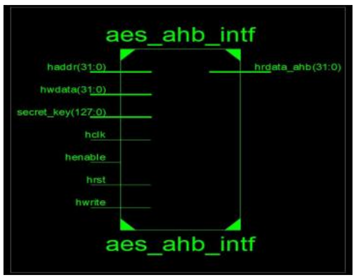


Fig.10: Showing RTL Schematic of AES AHB Interface

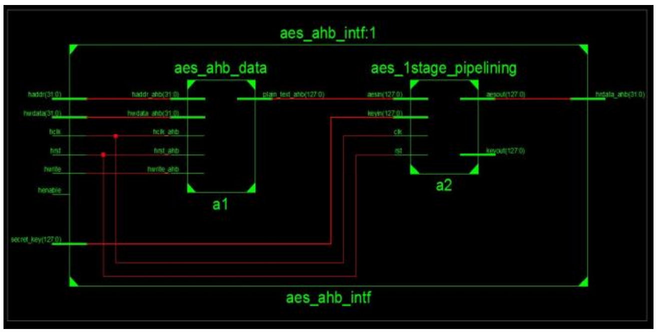


Fig.11: Showing Detailed RTL Schematic of AES AHB Interface

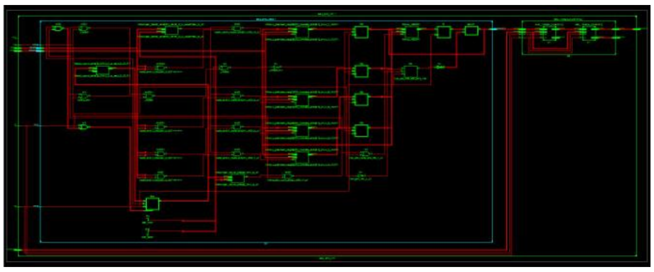


Fig.12: Showing Detailed RTL Schematic of AES AHB Interface_1

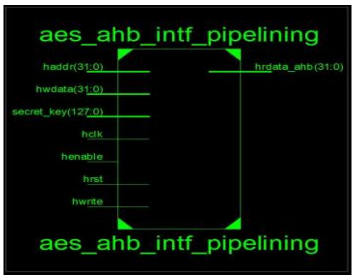


Fig.13: Showing RTL Schematic of AES AHB Interface Pipelining

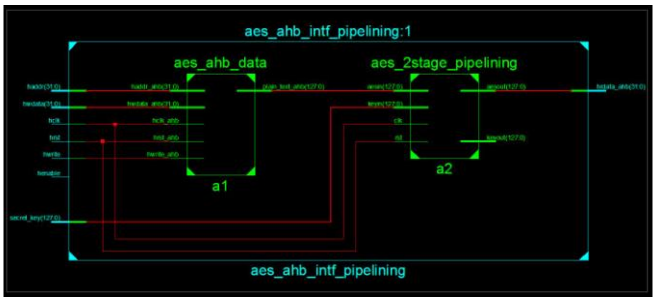


Fig.14: Showing Detailed RTL Schematic of AES AHB Interface Pipelining

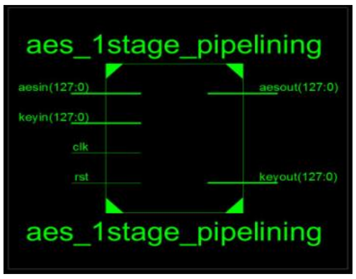


Fig.15: Showing RTL Schematic of AES 1-Stage Pipelining

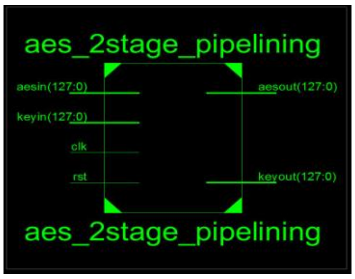


Fig.16: Showing RTL Schematic of AES 2-Stage Pipelining

Fig.17: Showing Simulation waveform of AES AHB DATA

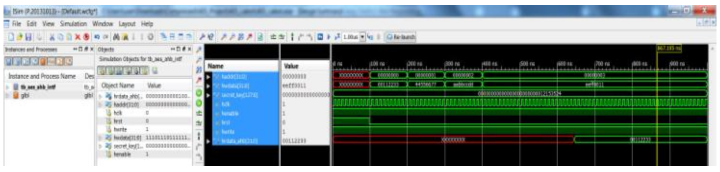


Fig.18: Showing Simulation waveform of AES AHB Interface

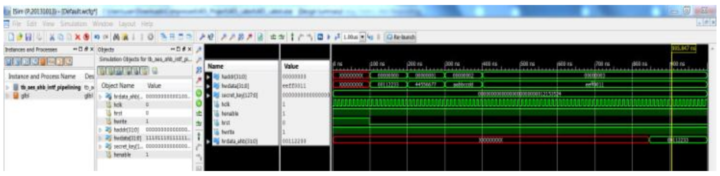


Fig.19: Showing Simulation waveform of AES AHB Interface Pipelining

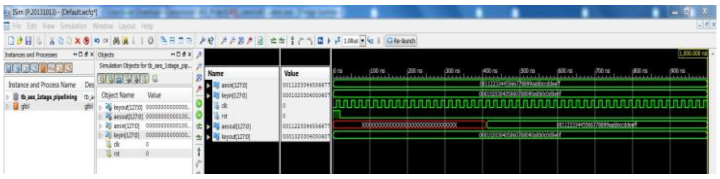


Fig.20: Showing Simulation waveform of AES 1-Stage Pipelining

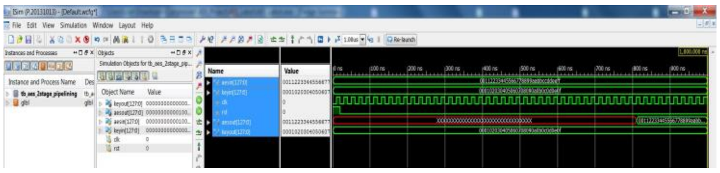
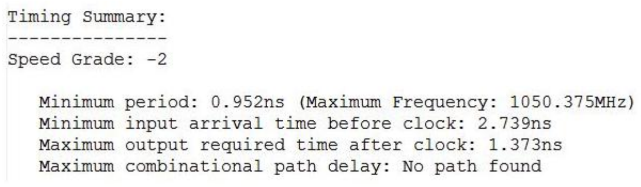


Fig.21: Showing Simulation waveform of AES 2-Stage Pipelining

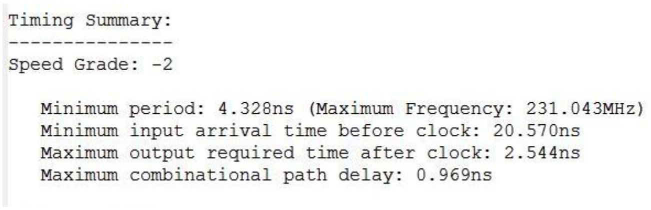
VI. TIMING DELAY SUMMARY



```
Timing Summary:
---------------
Speed Grade: -2

   Minimum period: 0.952ns (Maximum Frequency: 1050.375MHz)
   Minimum input arrival time before clock: 2.739ns
   Maximum output required time after clock: 1.373ns
   Maximum combinational path delay: No path found
```

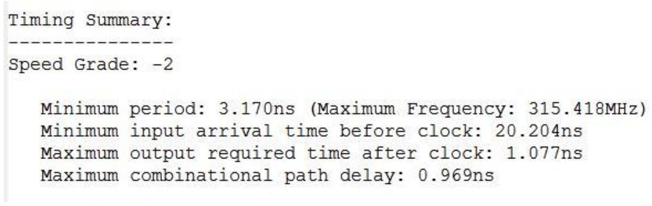
Fig.22: Showing Timing Summary of AES AHB DATA



```
Timing Summary:
---------------
Speed Grade: -2

   Minimum period: 4.328ns (Maximum Frequency: 231.043MHz)
   Minimum input arrival time before clock: 20.570ns
   Maximum output required time after clock: 2.544ns
   Maximum combinational path delay: 0.969ns
```

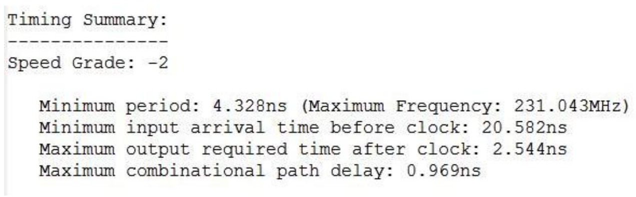
Fig.23: Showing Timing Summary of AES AHB Interface



```
Timing Summary:
---------------
Speed Grade: -2

   Minimum period: 3.170ns (Maximum Frequency: 315.418MHz)
   Minimum input arrival time before clock: 20.204ns
   Maximum output required time after clock: 1.077ns
   Maximum combinational path delay: 0.969ns
```

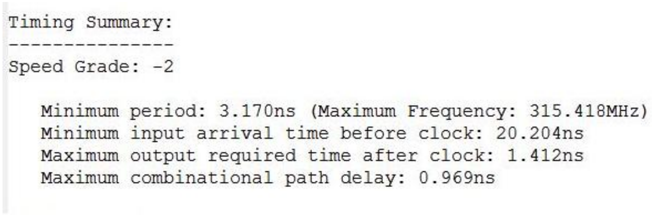
Fig.24: Showing Timing Summary of AES AHB Interface Pipelining



```
Timing Summary:
---------------
Speed Grade: -2

   Minimum period: 4.328ns (Maximum Frequency: 231.043MHz)
   Minimum input arrival time before clock: 20.582ns
   Maximum output required time after clock: 2.544ns
   Maximum combinational path delay: 0.969ns
```

Fig.25: Showing Timing Summary of AES 1-Stage Pipelining

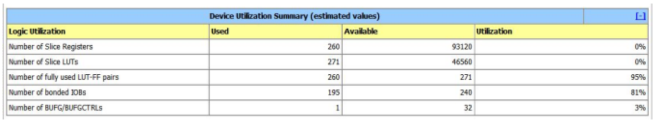


```
Timing Summary:
---------------
Speed Grade: -2

   Minimum period: 3.170ns (Maximum Frequency: 315.418MHz)
   Minimum input arrival time before clock: 20.204ns
   Maximum output required time after clock: 1.412ns
   Maximum combinational path delay: 0.969ns
```

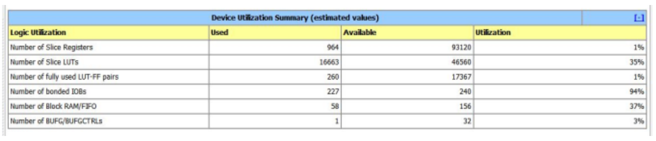
Fig.26: Showing Timing Summary of AES 2-Stage Pipelining

VII. DEVICE UTILIZATION SUMMARY



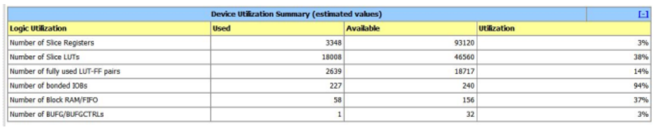
Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	260	93120	0%
Number of Slice LUTs	271	46560	0%
Number of fully used LUT-FF pairs	260	271	95%
Number of bonded IOBs	195	240	81%
Number of BUFG/BUFGCTRLs	1	32	3%

Fig.27: Showing Device Utilization Summary of AES AHB DATA



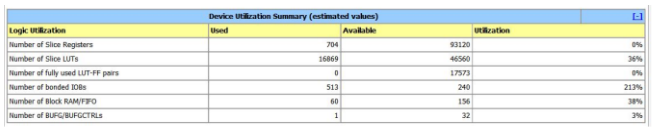
Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	964	93120	1%
Number of Slice LUTs	16663	46560	35%
Number of fully used LUT-FF pairs	260	17367	1%
Number of bonded IOBs	227	240	94%
Number of Block RAM/FIFO	58	156	37%
Number of BUFG/BUFGCTRLs	1	32	3%

Fig.28: Showing Device Utilization of AES AHB Interface



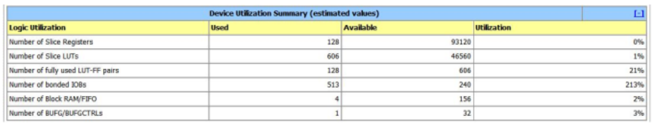
Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	3348	93120	3%
Number of Slice LUTs	18008	46560	38%
Number of fully used LUT-FF pairs	2639	18717	14%
Number of bonded IOBs	227	240	94%
Number of Block RAM/FIFO	58	156	37%
Number of BUFG/BUFGCTRLs	1	32	3%

Fig.29: Showing Device Utilization of AES AHB Interface Pipelining



Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	704	93120	0%
Number of Slice LUTs	16869	46560	36%
Number of fully used LUT-FF pairs	0	17573	0%
Number of bonded IOBs	513	240	213%
Number of Block RAM/FIFO	60	156	38%
Number of BUFG/BUFGCTRLs	1	32	3%

Fig.30: Showing Device Utilization of AES 1-Stage Pipelining



Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	128	93120	0%
Number of Slice LUTs	606	46560	1%
Number of fully used LUT-FF pairs	128	606	21%
Number of bonded IOBs	513	240	213%
Number of Block RAM/FIFO	4	156	2%
Number of BUFG/BUFGCTRLs	1	32	3%

Fig.31: Showing Device Utilization of AES 2-Stage Pipelining

VIII. CONCLUSION

AMBA AHB and other standard interface protocols with enhanced AES provide the proposed architecture more flexibility, enabling it to be used and integrated into additional compound organizations comparable SoCs.

IX. REFERENCES

[1] NIST, "Federal Information Processing Standards (FIPS)
[2] http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf, Nov. 2001.J. Daemen and V. Rijmen, "AES Proposal: Rijndael," Aug. 1998.
[3] P. Chodowiec, "Comparison of the Hardware Performance of the AES Candidates Using Reconfigurable Hardware,".
[4] J. Nechvatal, et. al., Report on the Development of the Advanced Encryption Standard (AES), National Institute of Standards and Technology, October 2, 2000.
[5] A. J. Elbirt, W. Yip, B. Chetwynd, C. Paar, "An FPGA implementation and performance evaluation of the AES block cipher candidate algorithm finalists.
[6] Performance Evaluation of the AES Block Cipher Candidate Algorithm.
[7] A. Menezes, P. van Oorschot, and S. Vanstone, Handbook of Applied Cryptography, CRC Press, New York, 1997, p. 81-83.
[8] Finalists," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 9, no. 4, pp. 545–557, Aug. 2001.

[9] T. Ichikawa, T. Kasuya, and M. Matsui, “Hardware Evaluation of the topology - ASIACRYPT 2001, pp. 239–254, 2001.

[10] X. Zhang And K. K. Parhi, "High Speed Vlsi Architectures For The Aes Algorithm," Ieee Transactions On Very Large Scale Integration (Vlsi) Systems, Vol. 12, No. 9, Pp. 957-967, September 2004.

[11] Sushma R Huddar, Sudhir Rao Rupanagudi, Ramya Ravi, Shikha Yadav & Sanjay Jain―Novel Architecture for Inverse Mix Columns for AES using Ancient Vedic Mathematics on FPGA.‖2013 IEEE.

[12] B. Gladman’s AES http://fp.gladman.plus.com/cryptography_technology.

[13] Salma Hesham, Mohamed A. Abd El Ghany.‖ High Throughput Architecture for the Advanced Encryption Standard Algorithm‖2014 IEEE.es and Practices‖ fourth edition.

[14] Computer Security Objects Register (CSOR): http://csrc.nist.gov/csor

[15] J. Daemen and V. Rijmen, The block cipher Rijndael, Smart Card research and Applications, LNCS 1820, Springer-Verlag, pp. 288-296.