Design of A Pre-Scheduled Data Bus for Advanced Encryption Standard Encrypted System-on-Chips

Xiaokun Yang[1] and Wujie Wen[2]

[1]Department of Engineering, University of Houston Clear Lake
[2]Department of Electrical and Computer Engineering, Florida International University

Abstract— This paper proposes a high efficiency data bus (DBUS) for Advanced Encryption Standard (AES) encrypted system-on-chips (SoCs). Using DBUS, the data sequence can be pre-selected for AES encryption/decryption, so that the state buffering and rescheduling overhead can be reduced. FPGA results show that the DBUS based design lowers the dynamic energy to 66.93%, and achieves up to 1.30 times higher valid throughput compared with the Advanced eXensible Interface (AXI) based implementation.

I. Introduction

To date, Advanced Encryption Standard (AES) has been widely used in multiple areas, such as Internet-of-Things (IoT) [9, 8] and wireless communication [6, 7]. Based on the premise that the plaintext/ciphertext can be input to AES engines immediately, many AES designs have been proposed to improve the circuits' performance. For instance, [17], [14], [10], and [13] decomposed and rearranged AES transformations as new linear and non-linear operations to achieve higher throughput. In addition, some parallelism and pipeline implementations have been used in [11] and [12] to provide a structural support for AES engines.

However, all the previous researches have not considered the data supply efficiency for AES cores using traditional interfaces, such as the AMBA Advanced High-Performance Bus (AHB), Advanced eXensible Interface (AXI), Wishbone, and OCP [1, 2, 3, 4]. Basically, AES is a symmetric block cipher that processes on a 4×4 matrix of bytes, named as AES state, in the cyclically shifted/inverse-shifted column-major order [5]. Since the data sequence in an AES state is non-linear and non-contiguous, two issues should be considered using conventional interfaces: 1) all the states on bus should be buffered then rescheduled before being encrypted/decrypted; 2) additional commands are required for each non-linear boundary operation. The logic of buffers and the scheduler can increase the slice cost, extra commands occupy bus cycles and reduce bus efficiency, and the frequent toggle activities of logics, signals & IOs consume much dynamic power.

To overcome these issues, this paper proposes a high performance data bus (DBUS) that can provide a way to efficiently transfer AES states to and from a circuit, and also backward support data transfer by traditional linear and block, which are presented by our previous work in [15]. Additionally, we compare the performance of DBUS with AXI – the dominant bus protocol widely used in industry today. More specifically, the key contributions are as follows.

- We propose the use of a specific state transfer, which processes 4×4 matrix of bytes in shifted/inverse-shifted column-major order, whereby a single bus command can put multiple pre-scheduled states of AES on DBUS. Since AES states are structured in such a way, the latency and power overhead for data buffering and rescheduling can be reduced. Our results show that the DBUS state transfer increases valid throughput to $\times 1.30$ and reduces dynamic energy to 66.93% compared with AXI.

- We propose a flexible DBUS, which is also backward compatible with the traditional linear and block transfer modes. Using transfer types of data bus separately and efficiently can cut the number of IOs and slices required by the micro-controller and help optimize chips size, power, and performance. For example, the valid bandwidth of DBUS block transfer is increased to $\times 1.20$ of AXI, and the dynamic energy of DBUS is 72.88% compared with AXI.

The remainder of this paper is organized as follows: section II introduces our previous work on DBUS protocol, and section III presents the novel state transfer mode proposed in this article. In section IV, the register transfer level (RTL) design, simulation, synthesis, and power analysis are introduced. The experimental results are shown in section V. Finally, section VI concludes this paper.

978-1-5090-1558-0/17/$31.00 ©2017 IEEE

II. Background

This section briefly introduces the interface or IOs for data communication. Before a master can commence the data bus, it must be granted access to the bus. This process is starting by the master asserting a request (REQ) to the arbiter. Then, the arbiter indicates when the master is granted use of the bus (GNT). A granted master starts a data transfer by driving command signals. These signals provide information on the address (ADDR), direction (WR), and length (LEN[9:0]) of the transfer, as well as the current transfer mode indicated by the most two significant bits of the LEN[11:10] signal. The 4-bit write data valid signal (WD_VLD[3:0]) indicates the corresponding valid byte of the word-size write data (WDATA[31:0]). Finally, slaves must send a data response (RESP[1:0]) within a timeout window, RESP[1] for write and RESP[0] for read.

As a DBUS timing diagram shown in Figure 1, there are two bus operations, one write followed by one read. Each operation consists of two distinct sections: the command phase, involving the bus arbitration and commands, and the data phase which may require several cycles depending on the burst length. In cycle 5 and 6, the write ready signal denoted by RESP[1] is de-assert, so the slave cannot receive the third write data (WD2) immediately, and the master should hold WD2 in the next two clock cycles. Similarly, in cycle 5 and 7, the read ready signal denoted by RESP[0] is de-assert, thus the read data on DBUS is not available, and the master needs to wait another cycle for a valid read data.

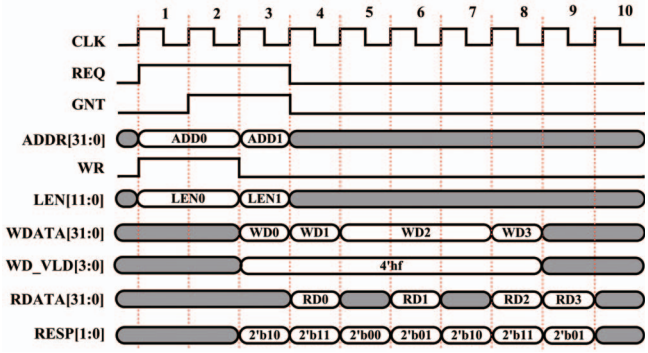


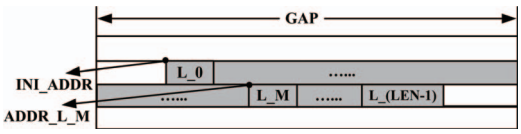
Fig. 1. DBUS Timing Diagram.

III. Proposed State Transfer Mode

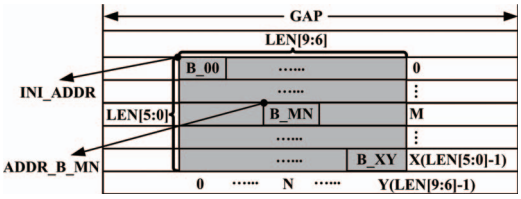
In this section, first, we mainly introduce our proposed state transfer type using the DBUS interface. Then, the timing diagrams of DBUS and AXI are compared.

A. Linear and Block Transfer Types

Before discussing the state transfer mode, we mainly illustrate the traditional linear and block transfer types.



(a) Linear



(b) Block

Fig. 2. Transfer Modes (a)Linear (b)Block

We define a bus transfer as a linear operation when the LEN[11:10] signal is binary 2'b00 and a block operation when 2'b01. As a linear transfer shown in Figure 2(a), the LEN[9:0] signal gives the exact data number in the row-major order. Assume that INI_ADDR, GAP, and DS, respectively, represent the initial address, memory gap between the data of the vertical neighbors, and the data size. The DS values of 0, 1, 2, 3 and 4 indicate byte, half word, word, double word, and quad word, respectively. Hence, the aligned initial address ADDR_0 is:

$$ADDR\_0 = (INI\_ADDR \ll DS) \gg DS. \quad (1)$$

Here, the shift operators "$\ll$" and "$\gg$", respectively, perform left and right shifts of their left operand by the number of bit positions given by the right operand. Furthermore, the Mth data address of a linear transfer can be calculated as:

$$ADDR\_L\_M = ADDR\_0 + (M \ll DS). \quad (2)$$

As a block transfer shown in Figure 2(b), the LEN[5:0] signal denotes the block height and the LEN[9:6] signal denotes the block width. Thus, the Nth column and Mth row data address can be calculated as:

$$ADDR\_B\_MN = ADDR\_0 + (M \times GAP) + (N \ll DS). \quad (3)$$

Since the non-linear addresses are computable by hardware using the DBUS protocol, only the initial address is driven on DBUS for the entire bursts.

B. Proposed State Transfer Mode

In this work, we further propose a new transfer type – state transfer, as an extension of the linear and block modes. As shown in Figure 3, the 128-bit matrix is adopted as the basic unit in the state mode. When the LEN[11:10] signal is binary 2'b10, the current transfer is indicated as a state transfer, and the LEN[9:0] signal represents the number of the states. The address of the Nth

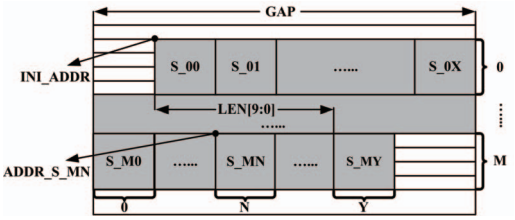


Fig. 3. State Transfer Mode.

state and Mth state-row thus can be formulated as:

$$ADDR\_S\_MN = ADDR\_0 + (4M \times GAP) + (N \ll 2). \quad (4)$$

Basically, each AES round, for both cipher and inverse cipher, consists of four transformations: 1) non-linear byte substitution using a S-Box (SB/ISB), 2) shifting rows of the state array (SR/ISR), 3) mixing the data within each column of the state array (MC/IMC), and 4) adding a round key to the state (AK), while the final round does not have the MC/IMC transformation. Since each column of an AES state should be mixed/inverse-mixed with four shifted/inverse-shifted bytes, the selected data on DBUS is pre-arranged in such a way, whereby the words of the arrays on DBUS are ready-to-use for AES encryption/decryption. The implementations of data buffering and rearrangement thus can be reduced.

As an example shown in Figure 4, the raw state is in the sequence of {'h0, 'h1, 'h2, 'h3}, {'h4, 'h5, 'h6, 'h7}, {'h8, 'h9, 'ha, 'hb}, and {'hc, 'hd, 'he, 'hf} for the first, second, third, and fourth columns, respectively. Using the state read transfer, the read data on DBUS is pre-scheduled in the sequence of {'h0, 'h5, 'ha, 'hf}, {'h4, 'h9, 'hc, 'h3}, {'h8, 'hd, 'h2, 'h7}, and {'hc, 'h1, 'h6, 'hb} for the first, second, third, and fourth words, respectively. Similarly, the write data on DBUS is pre-selected in the sequence of {'h0, 'hd, 'ha, 'h7}, {'h4, 'h1, 'hc, 'hb}, {'h8, 'h5, 'h2, 'hf}, and {'hc, 'h9, 'h6, 'h3}, respectively, for the first, second, third, and fourth words using the state write transfer.

Additionally, notice that a single command (C0) from a master can put two states of AES (S0 and S1) on DBUS at the same time, since the state transfer processes data by 4×4 byte matrices.

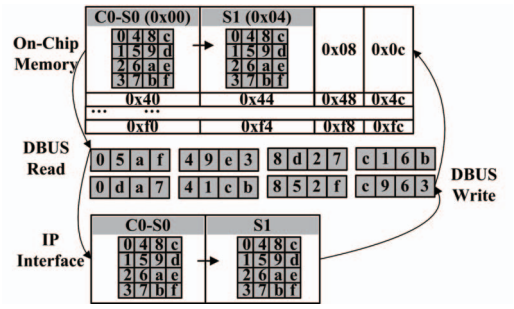
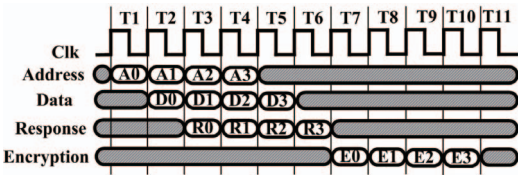


Fig. 4. Two-State Operation using State Mode.

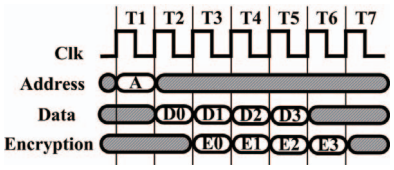
C. Comparison Between AXI and DBUS

In this section, we compare the state feeding efficiency using AXI and DBUS. Using the conventional AXI bus, additional commands are needed when data crossing each memory boundary. As an example shown in Figure 5(a), four addresses (A0, A1, A2, and A3) are needed to access one 4×4 byte matrix due to the non-continuous addresses. In contrast, all the addresses can be computed by hardware using the DBUS state mode, so only the initial address (A) is required as shown in Figure 5(b).

In addition, the data encryption starts at T7 cycle in Figure 5(a), after all the 4 words of a state (D0, D1, D2, and D4) has been stored. In Figure 5(b), however, the pre-shifted plaintext can be immediately encrypted at T3 cycle, since each word (D0) has been pre-scheduled using the state transfer mode.



(a) AXI



(b) DBUS

Fig. 5. Timing Diagrams (a)AXI (b)DBUS

IV. Hardware Design and Simulation

As a case study, we implement AES-encrypted Direct Memory Accesses (DMAs) using AXI and DBUS interfaces, termed A-DMA and D-DMA. Moreover, the 32- and 64-bit bus sizes are also considered in order to provide more design solutions.

A. DMA Design

As an example, Figure 6 shows the AES-encrypted DMA structure. It can be accessed by all the masters located on AXI or DBUS. The arbiter grants requests according to each master's priority. Using the command controller and address mapping modules, the granted bus commands can be converted between the bus side and memory side. The data paths are used to multiplex cipher and non-cipher processing between bus masters and memory. For the non-cipher tests, including linear and block vectors, the AES encryption/decryption will be bypassed. Otherwise, the write data path decrypts the ciphertexts

from the data bus then writes the plaintexts into memory, or the read data path encrypts the plaintexts from memory then drive the ciphertexts to the data bus.

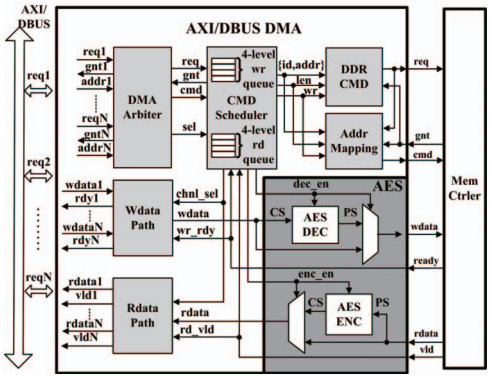


Fig. 6. AES Encrypted DMA Structure

B. AES Engine

In order to evaluate the bus cipher performance, an AES engine is also implemented using the 10-round algorithm with 4-word key and the GF($[(2^2)^2]^2$) construction with ϕ=$\{10\}_2$, λ=$\{1100\}_2$ [16]. Considering the system performance, we rearrange the transformations and split each AES round into two substages. As shown in Figure 7, the SB/ISB transformation is decomposed as a modular inversion over GF(2^4) and four linear functions – A, IA, δ, and $I\delta$. To shorten the SB/ISB critical path, IA is combined with δ ($IA \times I\delta$) in substage1, and $I\delta$ is combined with A ($I\delta \times A$) in substage2. In addition, the SR/ISR, MC/IMC, and AK transformations are also merged in substage2 to obtain approximately equal delay to substage1.

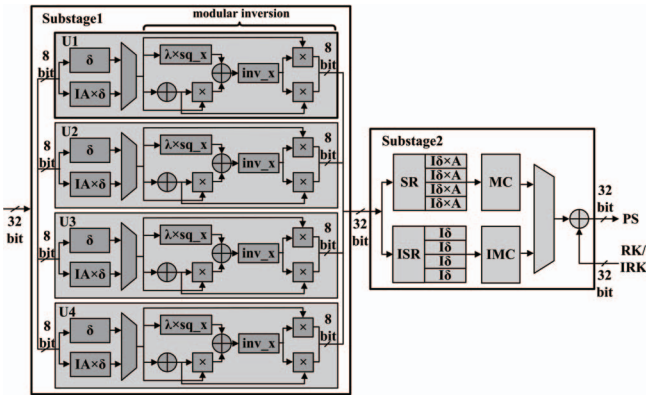


Fig. 7. AES Engine

Using a 32-bit AXI or DBUS, each round of an AES state spends 8 cycles – 4 cycles for substage1 and 4 cycles for substage2. When the data processing is fully pipeline, each round uses 5 clock cycles with 3 cycles overlapped between substage1 and substage2. Similarly, each round costs 3 cycles with 1 cycle overlapped using 64-bit AXI or DBUS.

TABLE I
Resource comparison

Resource Cost	32-bit A-DMA[a]	32-bit D-DMA[b]	64-bit A-DMA	64-bit D-DMA
IOs	533	324	661	460
Slices	15028	13264	13809	11368
MOF[c] (MHz)	167	216	163	207

[a]A-DMA: AXI DMA combined with AES Core
[b]D-DMA: DBUS DMA combined with AES Core
[c]MOF: Maximum Operational Frequency

C. Typical Test Cases

In addition, the typical test cases used in our study are that 40 words, 10×4 words, and 10 AES states, respectively, are written into memory then read out using linear, block, and state transfer modes. As an example of the cipher or state test, two masters initiate 10-state write and read commands sequentially. The design-under-test (DUT) grants the write request first, decrypts the ciphertexts, and then writes the plaintexts into memory. Meanwhile, it encrypts the plaintexts read from memory and drives the ciphertexts on buses.

After each test, a Value Change Dump (VCD) file, including specific switching information (toggle rates, signal rates, and frequency information) that gives the most accurate dynamic power estimation, will be generated.

V. Experimental results

A. Synthesis and Area Cost

In what follows, we synthesize register transfer level (RTL) designs using Xilinx ISE 14.7 with the target device Virtex6xc6vlx550t-2ff1760. As the practical results summarized in Table I, it can be observed that the compact DBUS structure reduces the IO & slice count, and achieves higher operational frequency compared with AXI bus for both 32- and 64-bit buses.

B. Power Analysis

After synthesis, we run design implementation, which comprises the mapping and placement & route steps. Furthermore, we use the XPower Analyzer to estimate the realistic power consumption. Xilinx Power Analyzer does an analysis on the real design Native Circuit Description (NCD) file and the specific simulation activity VCD file. Once all setting are properly set, the summary lists all the relevant power for the design shown in Table II. Static power (SP) results primarily from transistor leakage current in the device. It is almost a constant for the same design with different test vectors.

TABLE II
Power comparison

Test Cases	SP[a] (mW)	DP[b] (mW)	TP[c] (mW)
32-bit AXI Linear	753	476	1229
32-bit DBUS Linear	751	413	1164
32-bit AXI Block	761	532	1293
32-bit DBUS Block	758	457	1215
32-bit AXI Cipher	764	594	1358
32-bit DBUS Cipher	759	518	1277
64-bit AXI Linear	761	451	1212
64-bit DBUS Linear	760	396	1156
64-bit AXI Block	777	563	1340
64-bit DBUS Block	772	482	1254
64-bit AXI Cipher	779	649	1428
64-bit DBUS Cipher	775	575	1350

[a]SP: Static Power
[b]DP: Dynamic Power
[c]TP: Total Power

TABLE III
Experimental Result Comparison

Test Cases	VB[a] (GBPs)	DE[b] (uJ)	SE[c] (KBps/Slice)	DEE[d] (GBps/J)
32-bit AXI Linear	1.04	0.29	69.46	2.19
32-bit DBUS Linear	1.17	0.23	88.26	2.83
32-bit AXI Block	0.98	0.35	65.21	1.84
32-bit DBUS Block	1.17	0.25	88.26	2.56
32-bit AXI Cipher	0.56	0.68	37.15	0.94
32-bit DBUS Cipher	0.73	0.46	54.83	1.40
64-bit AXI Linear	2.00	0.14	144.83	4.43
64-bit DBUS Linear	2.29	0.11	201.07	5.77
64-bit AXI Block	1.92	0.19	139.04	3.41
64-bit DBUS Block	2.29	0.13	201.07	4.74
64-bit AXI Cipher	0.86	0.48	62.07	1.32
64-bit DBUS Cipher	1.33	0.28	117.29	2.32

[a]VB: Valid Bandwidth
[b]DE: Dynamic Energy
[c]SE: Slice Efficiency
[d]DEE: Dynamic Energy Efficiency

Our work focuses on analyzing Dynamic power (DP) shown in the third column, which is associated with clock power and switching events of IOs, Signals, and Logic of the design. First of all, it can be observed that DBUS tests consume less dynamic power compared with AXI tests due to the less toggle activities. In addition, 64-bit bus consumes more dynamic power than 32-bit bus, as shown in the block and cipher results. In the linear tests, however, the dynamic power consumed by 32-bit bus surpasses 64-bit bus. Because the switching activity is very low in the linear tests, and the clock power becomes the dominant factor.

C. Performance Metrics

Traditional bus bandwidths can be formulated as $2 \times 4 \times OF$ for the 32-bit full-duplex bus and $2 \times 8 \times OF$ for the 64-bit full-duplex bus, where OF denotes the operational clock frequency. To concentrate on analyzing data feeding efficiency, we rewrite valid bandwidth (VB) as the valid data without protocol overhead that can be transferred in one clock cycle. Thus, the 32- and 64-bit valid bandwidths can be formulated as:

$$VB = (8 \times OF \times N)/CY. \quad (5)$$

$$VB = (16 \times OF \times N)/CY. \quad (6)$$

Here, CY denotes the number of clock cycles, N denote the number of valid data (word/double-word) transferred on bus.

Furthermore, the slice efficiency (SE) is computed as the valid bandwidth divided by the slice count, in order to estimate the resource efficiency in terms of valid data number that can be transferred per second per slice cost. Similarly, the dynamic energy efficiency (DEE) is calculated as the valid bandwidth divided by dynamic power to evaluate the performance in terms of valid data number that can be transferred per joule consumption. Finally, the dynamic energy (DE) is computed as the multiplication of average dynamic power and time latency.

D. Result Summary

We summarize the experimental results, involving valid bandwidth, dynamic energy, slice efficiency, and dynamic energy efficiency, in Table III. In the second and third columns, it can be observed that DBUS based designs achieve higher valid bandwidth and consume less dynamic energy compared with AXI based implementations. In the third column, notice that the dynamic energy of the 32-bit DBUS increases to 1.64 × of 64-bit DBUS due to the higher bus latency, although the 32-bit DBUS consumes less dynamic power than the 64-bit DBUS.

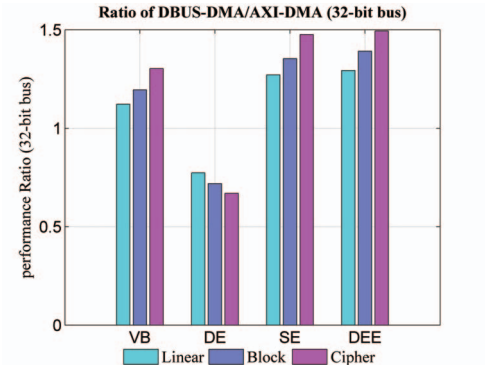
The slice and dynamic energy efficiency are shown in the fourth and fifth columns. As an example of the 32-bit buses, the DBUS and AXI cipher tests, respectively, can transfer 54.83 gigabytes and 37.15 gigabytes per second per slice cost. Moreover, they can transfer 1.40 gigabytes and 0.94 gigabytes per second per watt usage, respectively.

E. Performance Comparison between AXI and DBUS

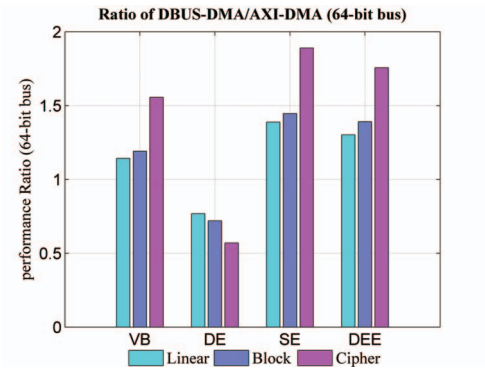
In sum, it is clear to see the difference by analyzing the 32-bit D-DMA/A-DMA ratios in Figure 8(a). All the valid bandwidth ratios are around 1.2 for the linear,

block, and cipher tests, and all the dynamic energy ratios are around 0.7. In this work, we focus on analyzing the performance of the cipher tests. The valid bandwidth of DBUS is 1.30 times compared with AXI for the cipher tests, and the dynamic energy consumption of DBUS is 66.93% compared with AXI. Additionally, the slice efficiency and dynamic energy efficiency of DBUS are 1.48 and 1.49 times compared with AXI.

Similarly, the performance of 64-bit DBUS surpasses 64-bit AXI as shown in Figure 8(b). For the cipher tests, the valid throughput ratio is 1.56 and the dynamic energy ratio is 0.57.



(a) 32-bit



(b) 64-bit

Fig. 8. Performance Comparison (a)32-bit (b)64-bit

VI. Conclusions

In this paper, we improve the data-feeding efficiency for AES systems using a high efficiency DBUS. By pre-scheduling data sequence in shifted/inverse-shifted order, the data on DBUS can be encrypted/decrypted immediately without additional latency and power cost. To extend the usage, DBUS is also backward compatible with the conventional linear and block transfer modes. As a case study, we implement DMA combined with AES engine using DBUS and AXI, involving both front-end and back-end FPGA designs. The results show that the DBUS outperforms AXI in terms of resource cost, valid throughput, and dynamic energy consumption.

References

[1] *AMBA AHB Specification,* Axis. Sunnyvale, CA, USA, 1999.

[2] *AMBA AXI Protocol Specification,* Axis. Sunnyvale, CA, USA, 2003.

[3] *Wishbone BUS,* Silicore Corp., Corcoran, MN, USA, 2003.

[4] *Open Core Protocol Specification,* OCP Int. Partnership, Beaverton, OR, USA, 2001.

[5] *Advanced Encryption Standard (AES),* FIPS-197, Nat. Inst. Of Standards and Technol., Nov. 2001.

[6] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification,* IEEE Standard 802.11-1999, 1999.

[7] *Specification of the Bluetooth System,* Bluetooth SIG, Kirkland, WA, USA, Dec 02, 2014.

[8] T. Xu, J. B. Wendt, and M. Potkonjak, "Security of IoT Systems: Design Challenges and Opportunities," *2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 417–423, Nov. 2014.

[9] J. Wurm, K. Hoang, O. Arias, A. Sadeghi, and Y. Jin, "Security Analysis on Consumer and Industrial IoT Devices," *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 519–524, Jan. 2016.

[10] M. M. Wong, M. L. D. Wong, A. K. Nandi, and I. Hijazin, "Construction of Optimum Composite Field Architecture for Compact High-Throughput AES S-Boxes," *IEEE Trans. Very Large Scale Integr. (TVLSI) Syst.*, vol. 20, no. 6, pp. 1151–1155, June 2012.

[11] M. Kermani and A Masoleh, "Efficient and High-Performance Parallel Hardware Architectures for the AES-GCM," *IEEE Trans. Comput. (TC)*, vol. 61, no. 8, pp. 1165–1178, Aug. 2012.

[12] N. Sklavos and O. Koufopavlou, "Architectures and VLSI Implementations of the AES-Proposal Rijndael," *IEEE Trans. Comput. (TC)*, vol. 51, no. 12, pp. 1454–1459, Dec. 2012.

[13] W. Suntiamorntut and W. Wittayapanpracha, "The Study of GF($(2^4)^2$) AES Encryption for Wireless FPGA Node," *International Journal of Communications in Information Science and Management Engineering*, vol. 3, no. 3, pp. 40–46, June 2012.

[14] C. Hsing Wang, C. Lin Chuang, and C. Wen Wu, "An Efficient Multimode Multiplier Supporting AES and Fundamental Operations of Public-Key Cryptosystems," *IEEE Trans. Very Large Scale Integr. (TVLSI) Syst.*, vol. 18, no. 4, pp. 553–563, April 2010.

[15] X. Yang and J. Andrian, "A High Performance On-Chip Bus (MSBUS) Design and Verification," *IEEE Trans. Very Large Scale Integr. (TVLSI) Syst.*, vol. 23, no. 7, pp. 1350–1354, July 2014.

[16] X. Zhang and K.K. Parhi, "On the optimum constructions of composite field for the AES algorithm," *IEEE Trans. Circuits Syst. II. Exp. Briefs*, vol. 53, no. 10, pp. 1153–1157, Oct. 2006.

[17] Y. Wang and Y. Ha, "FPGA-Based 40.9-Gbits/s Masked AES With Area Optimization for Storage Area Network," *IEEE Trans. Circuits Syst. II.*, vol. 60, no. 1, pp. 36–40, Jan. 2013.