A High Performance Advanced Encryption Standard (AES) Encrypted On-Chip Bus Architecture for Internet-of-Things (IoT) System-on-Chips (SoC)

Xiaokun Yang
Florida International University, xyang001@fiu.edu

FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

A HIGH PERFORMANCE ADVANCED ENCRYPTION STANDARD (AES) ENCRYPTED ON-CHIP BUS ARCHITECTURE FOR INTERNET-OF-THINGS (IOT) SYSTEM-ON-CHIPS (SOC)

A dissertation submitted in partial fulfillment of the

requirements for the degree of

DOCTOR OF PHILOSOPHY

in

ELECTRICAL ENGINEERING

by

Xiaokun Yang

2016

To: Interim Dean Ranu Jung
College of Engineering and Computing

This dissertation, written by Xiaokun Yang, and entitled A High Performance Advanced Encryption Standard (AES) Encrypted On-Chip Bus Architecture for Internet-of-Things (IoT) System-on-Chips (SoC), having been approved in respect to style and intellectual content, is referred to you for judgment.

We have read this dissertation and recommend that it be approved.

Gang Quan

Hai Deng

Deng Pan

Malek Adjouadi

Jean H. Andrian, Major Professor

Date of Defense: March 25, 2016

The dissertation of Xiaokun Yang is approved.

Interim Dean Ranu Jung
College of Engineering and Computing

Andrés G. Gil
Vice President for Research and Economic Development and Dean of the University Graduate School

Florida International University, 2016

DEDICATION

I dedicate this dissertation to my wife and my parents. Without their encourage, understanding, and most of all love, the completion of this work would not have been possible.

ACKNOWLEDGMENTS

First and foremost, I am grateful to my major advisor, Dr. Jean H. Andrian, for being friendly, caring, supportive, and help in numerous ways. Without his support, I could not have done what I was able to do. He was very generous in sharing his experiences on electrical and computer engineering, academic life and beyond. He is not only my adviser, but also, a friend inspiring me for the rest of my life.

Next, I would like to thank the members of my committee, Dr. Gang Quan, Dr. Hai Deng, Dr. Deng Pan, and Dr. Malek Adjouadi, for their support and suggestions in improving the quality of this dissertation. It is truly honored to have such great fantastic and knowledgeable professors serving as my committee members.

I would also like to thank all the lab mates and members at the Wireless Communications and Networking Laboratory for creating an amazing working environment, and thank my friends, Dr. Tianyi Wang and Dr. Nansong Wu, for their assistance on work related to my research.

Furthermore, I would also like to acknowledge the research support provided from the Department of Electrical and Computer Engineering at Florida International University, and the dissertation year fellowship from the graduate school during my dissertation research.

Finally, I want to thank my family for their unconditional love, faith, and encouragement.

ABSTRACT OF THE DISSERTATION

A HIGH PERFORMANCE ADVANCED ENCRYPTION STANDARD (AES) ENCRYPTED ON-CHIP BUS ARCHITECTURE FOR INTERNET-OF-THINGS (IOT) SYSTEM-ON-CHIPS (SOC)

by

Xiaokun Yang

Florida International University, 2016

Miami, Florida

Professor Jean H. Andrian, Major Professor

With industry expectations of billions of Internet-connected things, commonly referred to as the IoT, we see a growing demand for high-performance on-chip bus architectures with the following attributes: small scale, low energy, high security, and highly configurable structures for integration, verification, and performance estimation.

Our research thus mainly focuses on addressing these key problems and finding the balance among all these requirements that often work against each other. First of all, we proposed a low-cost and low-power System-on-Chips (SoCs) architecture (IBUS) that can frame data transfers differently. The IBUS protocol provides two novel transfer modes the block and state modes, and is also backward compatible with the conventional linear mode. In order to evaluate the bus performance automatically and accurately, we also proposed an evaluation methodology based on the standard circuit design flow. Experimental results show that the IBUS based design uses the least hardware resource and reduces energy consumption to a half of an AMBA Advanced High-Performance Bus (AHB) and Advanced eXensible Interface (AXI). Additionally, the valid bandwidth of the IBUS based design is 2.3 and 1.6 times, respectively, compared with the AHB and AXI based implementations.

As IoT advances, privacy and security issues become top tier concerns in addition to the high performance requirement of embedded chips. To leverage limited resources for tiny size chips and overhead cost for complex security mechanisms, we further proposed an advanced IBUS architecture to provide a structural support for the block-based AES algorithm. Our results show that the IBUS based AES-encrypted design costs less in terms of hardware resource and dynamic energy (60.2%), and achieves higher throughput ($\times 1.6$) compared with AXI.

Effectively dealing with the automation in design and verification for mixed-signal integrated circuits is a critical problem, particularly when the bus architecture is new. Therefore, we further proposed a configurable and synthesizable IBUS design methodology. The flexible structure, together with bus wrappers, direct memory access (DMA), AES engine, memory controller, several mixed-signal verification intellectual properties (VIPs), and bus performance models (BPMs), forms the basic for integrated circuit design, allowing engineers to integrate application-specific modules and other peripherals to create complex SoCs.

TABLE OF CONTENTS

CHAPTER PAGE

LIST OF TABLES

TABLE PAGE

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The Internet of Things (IoT) represents a major departure in the history of the Internet, as connections move beyond computing devices, and begin to power billions of everyday things, such as Google glasses, Apple watch, Fitbit body scale, Philips smart lights, and the Nike wristband. Basically, IoT is a term that describes a system where the Internet is connected to the physical world via embedded chips and ubiquitous sensors. By sensing our surrounding environment, IoT will create many practical improvements in our world, increasing our convenience, health, and safety, while at the same time improving energy efficiency and comfort.

The full vision of IoT will become a reality when billions of tiny devices can securely and intelligently connect and interoperate, and can be simply and securely accessed by applications and cloud services. In the future, we expect most new physical objects to have some sort of chips and sensors implanted with them. It will change the world in an even more profound way than has the Internet. As the prediction shown in Figure 1.1, it is estimated by Cisco's Internet Business Solutions Group (IBSG) that by 2020 as many as 50.1 billion devices of all types, shapes, and sizes will be wirelessly connected to the Internet. Billions of things will be going online soon, thanks to IoT, and new bus architectures that are high-performance and high-security - with low cost and low power system on chips - will be needed to power them.

1.1 High Performance On-Chip Bus Architecture

With industry expectations of billions of new smart Internet-connected things, commonly referred to as IoT, we see a growing demand for highly customized embedded chips with two attributes: low-cost and low-power [145, 43, 66, 89, 111].

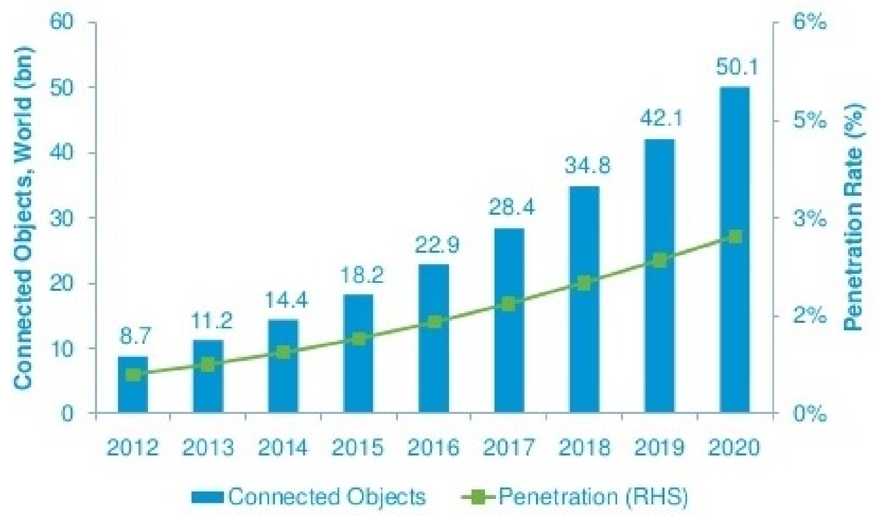


Figure 1.1: Number of Internet-Connected Things [11].

For the energy-limited IoT and wearable devices, low power technologies to prolong the life of the battery becomes the important constraint of the embedded chip design. Since more than 50% of the total dynamic power dissipation in a processor is due to interconnection [133], the on-chip bus becomes one of the main design issues, which dominates the power consumption and degrades the performance due to its poor scalability. Therefore, a reduced interface complexity and minimal power consumption on-chip bus architecture is necessary to the smart devices.

However, nearly all the existing bus protocols, such as AMBA Advanced High-Performance Bus (AHB) [1] and Advanced eXensible Interface (AXI) [5] from ARM Holdings, Wishbone from Silicore Corporation [6], OCP from OCP-IP [4], CoreConnect from IBM [2], and STBus from STMicroelectronics [7], define their standards either from high-throughput communication features and/or using complex structures for a wide variety of controllers, rather than a particular set of IoT circuits. As an example shown in Figure 1.2, a typical AMBA system consists of a number of master and slave devices connected together through some form of interconnection. It supports high-performance, high-frequency system designs, and provides a single

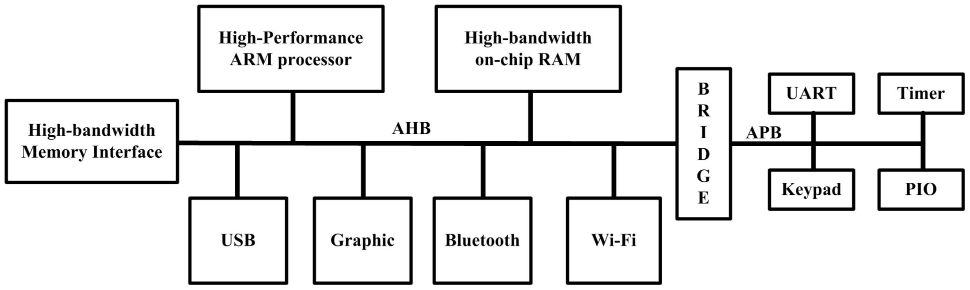

High-Performance ARM processor
High-bandwidth on-chip RAM
High-bandwidth Memory Interface
AHB
B R I D G E
APB
UART
Timer
Keypad
PIO
USB
Graphic
Bluetooth
Wi-Fi

Figure 1.2: A Typical AMBA Architecture [1].

interface definition, for the interfaces between a master and the interconnection, between a slave and the interconnection, and between a master and a slave. The interface definition supports a variety of different interconnection implementations. Hence, the protocol:

- is suitable for high-bandwidth and low-latency designs.
- meets the interface requirements of a wide range of components.
- provides flexibility in the implementation of interconnect architectures.

Because of the highly flexible nature and the complex structure, such buses can incur significant area and power penalties, and thus are not well-suited to the small-scale and resource-limited IoT circuits.

In this context, we propose a compact and power-efficiency IoT bus architecture (IBUS) that can balance performance with cost and implement the features required for low-power and high-throughput. In brief [186], IBUS is a dual-bus structure providing two novel transfer modes, block and cipher transfer types, and also backward compatible with the conventional linear transfer mode. The control bus, termed IC bus, has only one master - the microprocessor. Likewise, the data bus, named as ID bus, has only one slave - the ID Direct Memory Access (DMA). As a typical IBUS System-on-Chip (SoC) architecture shown in Figure 1.3, all the

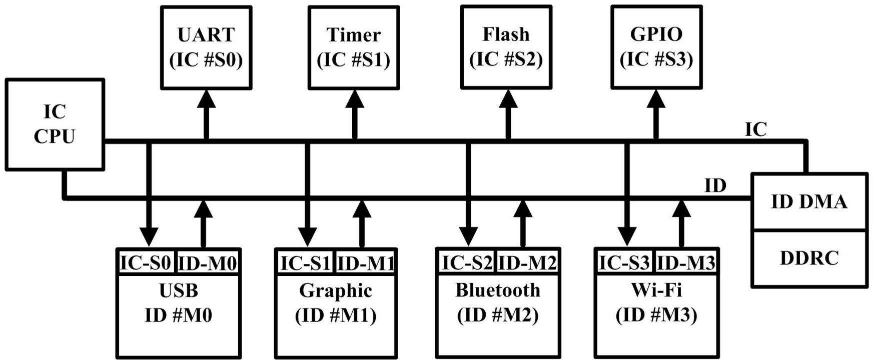


Figure 1.3: A Typical IBUS Architecture.

peripherals, such as UART, Timer, Flash Controller, and GPIO, as well as all the application-specific devices, are IC bus's slaves. They are configured by microprocessor through IC bus directly. In the other side, all the devices, including USB On-The-Go (OTG), Graphic module, Bluetooth, and Wi-Fi Media Access Control (MAC), are the masters of ID bus. They access the only slave memory through ID bus. Experimental results from both the analytical models and the practical tests show that the latency of IBUS is close to 63% of the AXI, and the energy consumption of IBUS-based designs can be reduced to a half compared with AXI-based implementations.

1.2 High Security On-Chip Bus Architecture

Another top tier concern of IoT embedded chips is privacy and security. Using insecure Internet-connected circuits to transfer data may suffer from significant risk, resulting in huge loss. One of the most useful methods for chip security is employing a cryptographic system, as the design of cipher algorithms is based on an advanced mathematical theorem. In January 1997, the National Institute of Standards and Technology (NIST) invited proposals for security algorithms for the Advanced Encryption Standard (AES) to replace the old Data Encryption Standard (DES). After two rounds of evaluation on the 15 candidate algorithms, NIST selected the Rijndael as the AES algorithm in October 2000 [19].

Today, many AES algorithm implementations for Rijndael have been proposed and their performance has been evaluated by using application-specific integrated circuit (ASIC) [72, 193, 47] and field programmable gate-array (FPGA) [175, 132]. Moreover, some structural optimization approaches have been widely employed to speed up the AES engines, such as parallel, pipelining, and subpipelining [117, 41, 30, 159, 163]. However, all the previous research focused on analyzing AES performance from the circuit perspective, based on the assumption that the 128-bit states can be input to AES engines immediately. From the system perspective, the bus protocol overhead for data transfers is necessary for all the SoC architectures. Particularly due to limited computing resources in IoT embedded chips, the bus protocol actually plays an important role in advancing the AES-encrypted circuit performance.

Traditional buses, such as AMBA AHB and AXI, only define data transfers by the linear mode. When transferring data by block, additional commands are required for each non-linear boundary operation. When processing data by AES state, a complex bus structure for data scheduling and buffering should be designed by architects, because the AES standard is a symmetric block cipher that processes on a 4×4 matrix of bytes in the column-major order and cyclically-shifted/cyclically-inverse-shifted.

Therefore, IBUS protocol is further upgraded to provide an architectural support for small-scale AES-encrypted circuits, in addition to being optimized for minimal power consumption and compact interface complexity. Figure 1.4 shows a typical IBUS structure with the AES security engine. Using the block transfer mode, the block boundary-crossing addresses can be calculated by the initial command. Us-

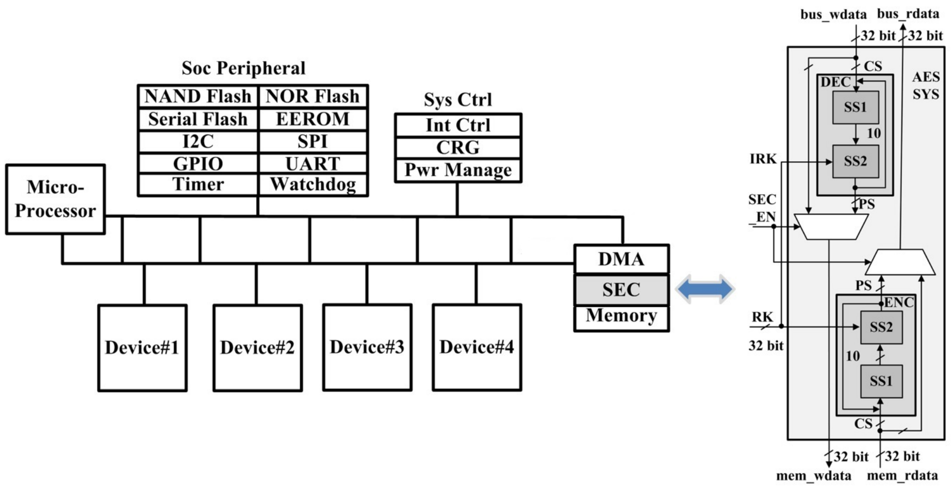


Figure 1.4: A Typical IBUS Architecture with an AES Engine.

ing the AES state transfer mode, IBUS can read/write data in the column-major order and cyclically-shifted/cyclically-inverse-shifted between memory and the AES engine. In this way, the data on IBUS is optimally reordered and prepared for the AES engine, thus the encryption/decryption processing can start immediately, without a scheduling and buffering delay on data bus. Comparing with AXI3, large reduction in IBUS architecture area and power consumption has been achieved.

1.3 Verification And Performance Evaluation Methodology

Discovering problems with system performance and power consumption late in the development cycle can be catastrophic to project schedules and product competitiveness, causing failure in the market. To predict the dynamic system performance and power of multi-function, multi-application SoC architectures, accurate simulation and analysis must be done in the design cycle to accelerate innovation. In earlier work, a lot of efforts have been put in modeling the bus latency [181], bandwidth, and wire efficiency [108], based on some system simplicity and assumption of normal conditions. Actually, it is an unfeasible task to accurately predict latency of a complicated system by static models. Likewise, the power consumption analysis, such as using some high level power analysis tools [131, 180, 133] or modeling power analysis [23, 171, 64, 84, 49, 90], suffers from more inaccuracies without gate-level parameters and switching activities. The high abstraction models are very hard to determine with accuracy in the early development stages.

To overcome the aforementioned issues, a novel performance evaluation methodology is proposed to estimate the system performance [178]. In our work, we extend the standard circuit design flow, and create multiple performance models and mixed-signal verification intellectual properties (VIPs) in both front-end and back-end environment.

Performance Evaluation Flow: Starting with algorithm analysis, several performance metrics are modeled by System Verilog hardware verification language [13]. Two output files are generated during front-end simulation, one is the VCD file with signal switching activities, and the other is the performance file with transfer latency and valid bandwidth. During the synthesis process, a hardware cost file, involving the maximum operating frequency (MOF), the number of IOs, and slice registers & slice LUTs, can be obtained. Furthermore, inputting the fully placed and routed NCD file, the physical constraints PCF file, and a specific simulation VCD file into the XPower Analysis tool, the detailed power consumption information can be generated in a PWR file. Finally, the energy and energy efficiency metrics are computed by the energy estimation model coded by Perl and tcl scripts, and then the final bus performance evaluation report is automatically derived.

Universal Verification Methodology (UVM) Based Environment: In order to verify and evaluate the chip performance, in either register transfer level (RTL) code or gate-level netlist, a Universal Verification Methodology (UVM) [16, 18] ver-

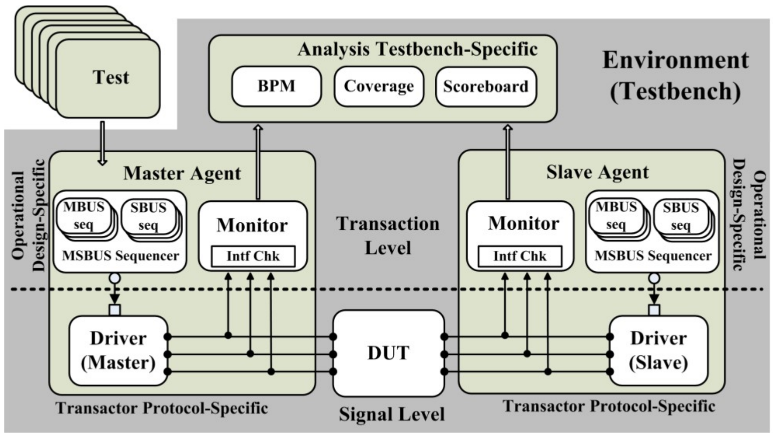


Figure 1.5: Mixed-Signal OVC

ification environment is set up. Apart from the traditional verification methods, such as timing check by assertions [76], a reusable framework to obtain functional and code coverage [82], and constrained random data generation [85], we also utilize several bus performance models (BPM) in this test bench to examine the bus performance in real time. The models are coded by System Verilog language, and are reusable, configurable, and seamlessly compatible with the UVM methodology. As an example shown in Figure 1.5, the verification environment integrates several encapsulated, ready-to-use, and configurable verification agents. Each of them, the master or slave agent, contains three subcomponents: the sequencer, driver, and monitor. The signal level or the physical level includes the design under test (DUT) and each agents' driver. The drivers are used to drive (in a master) or respond to (in a slave) the bus-signal interface. Considering the sequencer and the monitor in the transaction level, all the operations are design-specific and interconnected with test vectors and signal events. The sequencer controls and arranges the flow of sequence items to the driver, and the monitor samples the activities and collects the transactions seen on the signal-level interface and sends them into the analyzer.

Mixed-Signal Open Verification Component (OVC): Simulation speed and a lack of test approaches are the main difficulties for mixed-signal verification. In our work, multiple equivalent high-level Radio Frequency (RF) VIPs are created using the System Verilog language and integrated into a mixed-signal UVM environment. Such models can be executed on a digital simulator, which is dramatically faster than the traditional methods using an analog solver. Traditional verification approaches on digital side, such as constrained random data generation, assertion-based verification, coverage-driven verification, and verification methodology manual (VMM)/UVM methodologies, can seamlessly be used in our proposed mixed-signal environment.

1.3.1 Design Automation

Creating a highly tailored SoC is very complicated even for the most experienced design teams, especially when the on-chip bus architecture is based on protocols that are new or otherwise unfamiliar to the team, such as the IBUS proposed in this dissertation. Deciphering intellectual properties (IPs) under different bus protocols, modeling reusable VIPs to build up a test bench, and accurately evaluating the chip performance to meet the design specifications are huge development effort requiring deep technical knowledge. It will prolong time-to-market, adds complexity and costs, and reduces chip reliability. More specifically, the barriers include:

- A long design cycle using the new protocol is susceptible to delays. Especially the industrial standard IP integration, such as AMBA, Wishbone, and OCP based IPs, can directly impact the project release.
- The lack of easy-to-use new protocol VIPs makes verification environment setup and test vector design a tedious chore. Engineering asks the industry and

academia for ideas as to how the cost of verification for these bus architectures could be significantly reduced.

- The accurate architectural performance is extremely hard to predict using an unfamiliar bus protocol. It introduces the tape-out risk.

Therefore, we points out that this will force the industry and academia to look for more standardization and scalability across the entire platform that will be used for whole classes of embedded chips for the IoT. To tackle the issues, we proposed a configurable and synthesizable IBUS architecture for integrating third-party IPs, a solution for enhancing SoC performance, shortening the design cycle, and reducing the chip tape-out risk. Specifically, our work includes the follows.

- Derived from the IBUS structure, a flexible dual-bus architecture is presented to improve AES-encrypted chip performance. In this architecture, we provide some customizable silicon-proven modules - DMA, AES encryption/decryption engine (ENC/DEC), and memory controller - well-suited to the IBUS structure, so that designers can focus on their application-specific designs.
- In order to integrate the third-party IPs from multiple chip vendors, we provide several configurable bus wrappers for the IBUS structure. Configurable protocol transformations, sizes of asynchronous Fist-in, First-out (FIFO), and clock domain crossing solutions are presented to facilitate adapting the IBUS interconnection of different industrial protocols.
- The trade-offs among resource cost, speed, and power consumption of the IBUS integration are considered and analyzed. As a case study, we demonstrate that the auto-generation algorithm can be effectively used to create an IBUS structure. Comparing with AXI3 implementations, the generated IBUS-based

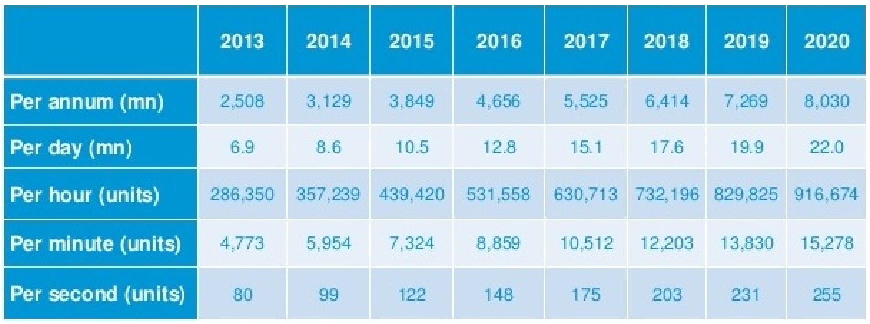
designs achieve higher valid bandwidth and consume less dynamic energy with less slice usage.

1.4 The Research Problems And Our Contributions

Since IoT is a new and emerging area, there is very little research on the bus protocol for IoT embedded chips. Previous work has been performed mainly on the SoC features of transfer speed, throughput, and power consumption. However, since most of the smart devices are small scale and battery-powered, the trade-offs among the silicon utilization, energy consumption, valid bandwidth, and security algorithm complexity must be considered. Therefore, our research mainly addresses four problems:

- Efficient IoT chip designs must balance a host of requirements that often work against each other. Low cost is important, but often supporting all the key features required by the application increases micro-controller IO number and slice count - two components that work against low cost. Low power is also important for IoT applications where battery operation is necessary. Adding features and performance can increase the power requirement, however. Clearly finding the right balance between all these requirements can be a problem, but that's just the type of challenge engineers expect from cutting-edge designs. One of the most effective ways to cut this design gordian knot is to look for a low-cost and power-efficiency architecture that can frame the data transfer problem differently. Using transfer types of data bus separately and efficiently, for example, can cut the number of IO and slice required by the micro-controller and help optimize chip size, power, and performance.

- As the connection speed shown in Figure 1.6, in 2016, 148 things-per-second will connect to the Internet, and by 2020 that number will reach 255 things/sec. The rapid rise in Internet-connected things has the potential to profoundly and positively affect our daily lives, however, it comes with a downside, as it opens up users to security vulnerabilities that did not previously exist. As IoT advances, the gap between small-scale chip performance and the security algorithm complexity widens. The resource limitation highlights that current bus architectures are not capable of keeping up with the computational demands of security processing, and the power constraint emphasizes that the energy consumption overhead of supporting security on power-limited embedded chips is very high.
- As the industry reaps the benefits of Moore's Law and technology nodes and chip designs increase in complexity, we emphasize integration of IP cores and building automatic design flows. Today, chip design is so complex that it is rapidly becoming unaffordable, design automation exploration is becoming intractable with the advent of new protocol and multiple IP integration, and technology nodes are nearly impossible to develop and use due to increasing variability. Meanwhile, studies have shown that verification continues to consume up to 70% of the development cost in each advanced node. UVM verification methodology combined with a System Verilog testbench improves verification of complex ASIC and FPGA designs with the potential benefits - high flexibility, randomization, re-usability, and higher levels of abstraction. Taking advantage of these benefits requires familiarity with object oriented programming (OOP), central to UVM and so essential to properly setting up a verification environment. These prerequisites can be time consuming, and even if engineers are familiar with OOP and UVM, developing from scratch



	2013	2014	2015	2016	2017	2018	2019	2020
Per annum (mn)	2,508	3,129	3,849	4,656	5,525	6,414	7,269	8,030
Per day (mn)	6.9	8.6	10.5	12.8	15.1	17.6	19.9	22.0
Per hour (units)	286,350	357,239	439,420	531,558	630,713	732,196	829,825	916,674
Per minute (units)	4,773	5,954	7,324	8,859	10,512	12,203	13,830	15,278
Per second (units)	80	99	122	148	175	203	231	255

Figure 1.6: Connecting Speed of Things [11].

a multi-layer verification environment with lots of connections often gives engineers too many chances to make mistakes. Too often, engineers spend lots of time just on start-up, finding and fixing errors, instead of debugging the design and performing other verification tasks. Because the UVM verification environment is well structured and test bench building blocks are defined by the standard, there exists the possibility of automatically generating many elements of the verification environment.

- SoC interconnection fabrics are categorized according to trade-offs among latency, throughput, speed, and silicon area, and the correctness and performance of these fabrics in FPGA applications are assessed through experimentation and simulation. Another strategy of this dissertation is to develop efficient evaluation methodologies and design flows that enable the design team to accomplish more with less resource without compromising the quality of results, enable evaluation of design options for complex chips using reliability metrics, analyze and mitigate variability to enhance parametric yield, and parallelize EDA tool applications to fully leverage front-end and back-end operations.

Toward these problems, we have made the following contributions in this dissertation:

- First of all, we propose a low-cost and low-power IBUS architecture to improve IoT chip performance and the capabilities to provide efficient architectural support for the AES-encrypted SoCs. In the IBUS protocol, not only do we define a compact and high efficient interface so as to reduce both resource cost and bus toggle rate, but we also create two novel bus transfer modes - the block mode to access data by matrix, and the state mode to enhance data supplying efficiency for the AES algorithm. Considering the tradeoffs among resource cost, data throughput, and energy consumption together, we also provide alternative architectural designs using different bus sizes. Based on the available resource, structural efficiency demands, and circuit performance requirements, engineers can choose different solutions to fulfill the constraints of different applications. Finally, we show that the block and state transfer modes can be efficiently and effectively used in the IBUS architecture, and then evaluate the AES-encrypted SoCs' performance using the evaluation methodology.

- We propose a performance evaluation methodology with a verification model factory (VMF), including several reusable IBUS-interfaced mixed-signal VIPs and ready-to-use control tasks, as well as some performance evaluation models for system verification. Several performance metrics, including slice count (SC), time cost (TC), wire efficiency (WE), bandwidth and valid data bandwidth (BW and VDB), static and dynamic power consumption (SP and DP), dynamic energy (DE), slice efficiency (SE), and dynamic energy efficiency (DEE), are performed to evaluate DUTs' performance. They are collected and computed during the standard circuit design flow, involving RTL design, UVM-based verification, synthesis, place & route, and power & energy analysis. The methodology solves the most critical dilemma facing chip makers today, how to begin developing performance models for a chip and using software to determine how a chip performs in the real world before the chip actually exists. This easy-to-use evaluation methodology helps SoC design with confidence and reduce risk while improving time to market.

- We propose several configurable and synthesizable bus wrappers underlying IBUS protocol that lend flexibility to the IBUS-based IP integration. Furthermore, we show that the IBUS protocol can be efficiently used to develop SoCs to meet the chip design requirements of high-performance and high-security, leverage limited resource of small-scale chips and overhead costs of complex security mechanisms, and shorten time to market due to its configurability. Using the evaluation methodology, we also present the power-area-throughput results under a variety of tests compared with AXI architectures. We found that our proposed IBUS costs less in terms of hardware resource and achieves higher throughput than the AXI-based design, and the dynamic energy consumption of IBUS is reduced to 66.2% compared with the AXI cipher test.

1.5 Structure Of The Dissertation

The rest of this dissertation is organized as follows. In Chapter 2, we introduce background to this dissertation and discuss related works that are close to our research problems. In Chapter 3, we propose a high performance on-chip bus architecture with consideration of reduced interface complexity, minimal power consumption, and high efficient data transfers. In Chapter 4, we propose a performance evaluation methodology based on the standard circuit design flow. In Chapter 5, we further propose an advanced IBUS protocol to leverage the overhead cost of complex AES algorithm and limited resource of IoT embedded chips. In Chapter 6, we present a configurable and synthesizable IBUS architecture for IP integration, in order to reduce the chip design cycle and tape-out risk. Finally, in Chapter 7, we conclude this dissertation and discuss possible future work.

CHAPTER 2
BACKGROUND AND RELATED WORK

In this chapter, we introduce the fundamentals on high-performance and high-security bus architectures for tiny size embedded chips. We then discuss the related research that deals with integrated circuit design flow and design automation, verification methodology, and chip performance evaluation.

2.1 Related Work On High Performance On-Chip Bus Architectures

As IoT advances, the increasing importance of low-cost and low-power on-chip architectures results in numerous designs and optimization in bus architectural topology and protocols, in both industry and academia.

2.1.1 High Performance Bus Protocols

A complicated IoT SoC today can contain dozens of processing engines: microprocessors, a verity of high-speed external interfaces, wireless communication modules, graphic processors, and enormous amounts of memory. These blocks are connected by either a proprietary or industry-standard buses, for instance, AMBA from ARM [1, 5], Wishbone from Silicore Corporation [6], OCP from OCP-IP [4], CoreConnect from IBM [2], STBus from STMicroelectronics [7], and others [102, 104, 114, 157, 168, 177, 115]. All of their data transfers require handshaking between a master and a slave, the performance is thus somewhat limited as this requirement does not allow one transfer to start unless the previous one has completed. For the advanced bus standards, such as AXI or OCP, all the address, data, control signals are channel-based or user-defined. This scheme efficiently improves the bandwidth because of the paralleled channel operations. However, a large number of wires and internal logic, such as multiplexers for different layer data conversion and buffers for data flow control, are necessary to form several sets of bus signals.

To avoid excessive wire usage, the phase-based optimization of on-chip network protocol has been introduced in [108, 88, 97]. Three-bit phase signals are used to distinguish the information of the shared channels. The phase signals facilitate the reduction of the communication time with phase interleaving and phase omission-restoration among successive transactions. But for the requirement of phase handshaking, especially for the case of intensive boundary-crossing transfers, either the latency or the wire efficiency is still degraded. A single bus architecture, named as SAMBA-Bus proposed in [141], enables multiple compatible bus transactions to be performed simultaneously with only one access grant from the bus arbiter as long as the bus destinations are uncommon. Otherwise, the bus communication conflicts and has to wait for the arbitration winner. In addition, the scheduler design of single winner with multiple transactions is more costly in terms of both complexity and area.

Other choices for bus performance improvement are multi-bus [167, 59, 98, 100] and multi-layer SoC structures [135, 107, 147]. While the bus bandwidth can be improved when most of the communications occur in the same bus level or the same bus layer, the latency could be even worse as the time consumption of any bridge crossing would be added in addition to the time of each individual bus or layer. Moreover, the bridges for bus interconnection and signal synchronization require a number of hardware resource, such as multiplexers for different layer data switching, arbiters for different layer command arbitration, and buffers/FIFOs for data flow control. All of these circuits significantly increase the hardware cost and communication latency.

As such, all of the existing buses are limited in the ability to support small-scale IoT chips in terms of both area and energy efficiency. As the IoT devices continue to advance at a fast pace, the desire for low-cost and high-efficiency bus protocol rapidly increases.

2.1.2 Low Power Technologies

The growing market of wearable and IoT devices demands embedded chip design with ultra low power dissipation. However, as the integration, size, and complexity of the chips continue to increase, the difficulty in providing adequate cooling might either add significant cost or limit the functionality of the computing systems which make use of those integrated circuits. Generally, the power dissipation in circuits can be classified into three categories as described below.

Dynamic Power Consumption: Dynamic power consumption is mainly due to the logic transitions causing logic gates to charge/discharge load capacitance. To understand how architectural strategies can provide high performance for perception applications at low power levels, it is necessary to look at the CMOS circuit dynamic power consumption equation.

$$P\_dy = A \times C \times V^2 \times F. \tag{2.1}$$

In this equation, P_dy is the dynamic power consumed, A is the activity factor, i.e., the fraction of the circuit that is switching, C is the switched capacitance, V is the supply voltage, and F is the clock frequency [189, 109, 183]. If a capacitance of C is charged and discharged by a clock signal of frequency F and peak voltage V, then the charge moved per cycle is CV and the charge moved per second is CVF. Since the charge packet is delivered at voltage V, the energy dissipated per cycle, or the power, is CV^2F. The data power for a clocked flip-flop, which can toggle at most once per cycle, will be $\frac{1}{2}CV^2F$. When capacitances are clock gated or when flip-flops do not toggle every cycle, their power consumption will be lower. In Equation 2.1, a variable called the activity factor ($0 \leq A \leq 1$) is used to model the average switching activity in the circuit. It is obvious that a high-efficiency bus protocol can drastically reduce the dynamic power consumption by lowering the activity factor for data transfers.

Short-Circuit Current: In a CMOS logic P-branch and N-branch are momentarily shorted as logic gate changes state resulting in short circuit power dissipation. In static CMOS circuits, the component of power due to short circuit current is about the 10% of the total power consumption. In dynamic circuits, however, we do not come across this problem, since there is no any direct DC path from supply voltage to ground.

Leakage Current: Leakage current is the power dissipation that occurs when the system is in standby mode or not powered. There are many sources of leakage current in MOSFET, diode leakages around transistors and n-wells, subthreshold leakage, gate leakage, tunnel currents etc.

Power optimization in a processor can be achieved at various abstract levels [87, 153, 188]. System, algorithm, and architecture levels have a large potential for power saving even these techniques tend to saturate as we integrate more functionality on an integrated circuit. So optimization at circuit and technology levels is also very important for miniaturization of circuits. An integrated low power methodology requires optimization at all design abstraction layers as mentioned below [10, 155].

System: At the system level, hardware reuse [25], power management [31], and clock gating technologies [149, 184, 160] are extensively used.

Algorithm: Low power algorithms, such as a hardware accelerator [35, 57] and address compression and encoding [136, 21, 148] technologies, are used at algorithm level.

Architecture: In system architecture, parallelism, pipelining, and redundancy can reduce power significantly [116, 38, 86]. In addition, multi-voltage [22] and power gate technologies, including the unified power format standard (UPF) which is supported by Synopsys [17, 173] and the common power format specification (CPF) which is supported by Cadence [12], are commonly used in industry nowadays.

Circuit Logic: Transistor resizing can be used to speed up circuit and reduce power [125, 166], and sleep transistors can be used effectively to reduce standby power [45, 143].

Technology: Dynamic power varies as V^2, so lowing the supply voltage dramatically reduces power dissipation. Dynamic/Switching power is due to charging and discharging of load capacitors driven by the circuit. Supply voltage scaling has been the most adopted approach to power optimization, since it normally yields considerable power savings due to the quadratic dependence of switching/dynamic power switching on supply voltage V [92, 128]. However, lowering the supply voltage affects circuit speed which is the major short-coming of this approach. So both design and technological solutions must be applied to compensate the decrease in circuit performance introduced by reduced voltage. Also threshold reduction, multi-threshold devices, and selective frequency reduction techniques can be used to reduce dynamic power [28, 50].

2.2 Related Work On Hardware Security

Internet connectivity varies among different applications, in general however, the security needs are all common. Security in networking is based on cryptography, the science and art of transforming messages to make them secure and immune to attack [62]. Encryption is one of the principal means to guarantee security of information. Encryption algorithms perform various substitutions and transformations on the plaintext and transforms it into ciphertext. Many of them have been widely available and used in information security. Generally, they can be classified into two groups: symmetric-key and asymmetric-key encryption [158].

Symmetric key encryption is a form of cryptosystem in which encryption and decryption are performed using the same key. In contrast, asymmetric encryption, also known as public-key encryption [162, 58], is a form of cryptosystem in which encryption and decryption are performed using different keys, one public key and one private key. The generation, modification, and transportation of keys have been done by the encryption algorithm, which is also named as cryptographic algorithm.

2.2.1 Cryptosystems

There are many cryptographic algorithms available in the market to encrypt data. The strength of encryption algorithm heavily relies on the computer system used for the generation of keys [58, 158, 54]. Some important encryption algorithms are discussed here:

Rivest-Shamir-Adleman (RSA): RSA is designed by Ron Rivest, Adi Shamir, and Leonard Adleman in 1978. It is one of the best known public key cryptosystems for key exchange or digital signatures or encryption of blocks of data. RSA uses a variable size encryption block and a variable size key. It is an asymmetric (public key) cryptosystem based on number theory, which is a block cipher system. It uses two prime numbers to generate the public and private keys. These two different keys are used for encryption and decryption purpose. Sender encrypts the message using receiver public key and when the message gets transmit to receiver, then receiver can decrypt it using its private key [78, 113, 24].

RSA operations can be decomposed in three broad steps: key generation, encryption, and decryption. It has many flaws in its design therefore not preferred for the hardware security use. When the small values of p & q are selected for the designing of key, then the encryption process becomes too weak and one can be able to decrypt the data by using random probability theory and side channel attacks. On the other hand, if large p & q lengths are selected then it consumes more time and the performance gets degraded in comparison with Data Encryption Standard (DES). Further, the algorithm also requires of similar lengths for p & q, practically this is very tough conditions to satisfy. Padding techniques are required in such cases increases the system's overheads by taking more processing time [56, 152, 51].

Data Encryption Standard (DES): For about two decades since 1977, the US government used a cipher called DES to protect sensitive, unclassified information. DES is one of the most widely accepted, publicly available cryptographic systems. It was developed by IBM in the 1970s but was later adopted by NIST, as Federal Information Processing Standard 46 (FIPS PUB 46). DES is a block cipher, which is designed to encrypt and decrypt blocks of data consisting of 64 bits by using a 64-bit key [123, 65]. Although the input key for DES is 64 bits long, the actual key used by DES is only 56 bits in length, because the least significant (right-most) bit in each byte is a parity bit. The algorithm goes through 16 iterations that interlace blocks of plaintext with values obtained from the key. It transforms 64-bit input in a series of steps into a 64-bit output. The same steps, with the same key are used for decryption. There are many attacks and methods recorded till now those exploit the weaknesses of DES, which made it an insecure block cipher. Despite the growing concerns about its vulnerability, DES is still widely used by financial services and other industries worldwide to protect sensitive on-line applications [105]. Since DES was proven to be insecure, however, prompting the government to look for a replacement.

Advanced Encryption Standard (AES) : This led to a standardization process that attracted 15 competing encryption designs, which included, among others, MARS from IBM [40], RC6 from RSA Security [142], Serpent [139], Twofish [37, 36], and Rijndeal [55]. It was Rijndael, designed by two Belgian cryptographers, Joan Daemen and Vincent Rijmen, that eventually became the standard and henceforth acquired the title AES.

The standard selection process was very stringent, taking 5 years to complete. Although the cipher's strength against various attacks was a major consideration in choosing the standard, other factors like speed, versatility, and computational requirements were likewise given importance. The government wanted an encryption standard that was not just strong, but also fast, reliable, and easily implemented in both software and hardware, even those with limited CPU and memory.

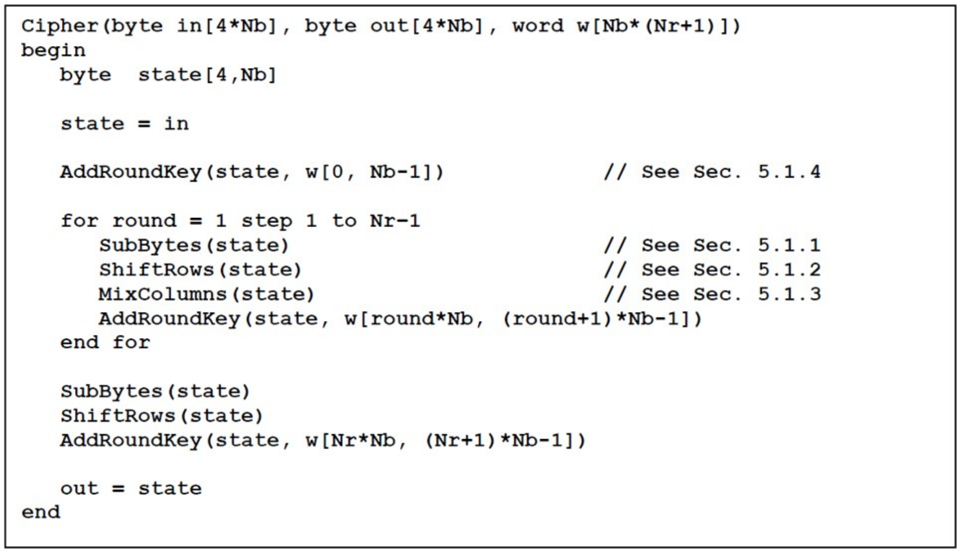
Basically, AES is a block cipher that processes data as a 4×4 matrix of bytes called a state [19, 93]. Figure 2.1(a) and 2.1(b), respectively, show the AES cipher and inverse cipher procedures with pseudo code. Each 128-bit information is operated by four primitive transformations, SubBytes (SB), ShiftRows (SR), MixColumns (MC), and AddRoundKey (AR) for encryption, and InvSubBytes (ISB), InvShiftRows (ISR), InvMixColumns (IMC), and AR for decryption. The length of the input block, the output block, and the state is 128 bits. This is represented by $N_b = 4$, which reflects the number of 32-bit words (number of columns) in the state.

During the encryption/decryption process, the four primitive transformations are executed iteratively in rounds, where the value represented by N_r will be 10, 12, or 14, depending on which key size is selected.

Although the other encryption algorithms were also very good, the Rijndael cipher was ultimately selected and declared a Federal Information Processing Standards or FIPS standard by the NIST in 2001. It was approved by the Secretary of Commerce and then recognized as a federal government standard the following year. Some of those ciphers are also widely used today but understandably do not enjoy the same level of acceptance as AES. The rise of AES did not end there. In 2003, the government deemed it suitable for protecting classified information. In fact, up to this day, the National Security Agency (NSA) is using AES to encrypt even top secret information. That should explain why AES has gained the confidence of various industries. If it is good enough for the NSA, then it must be good enough for IoT industry.

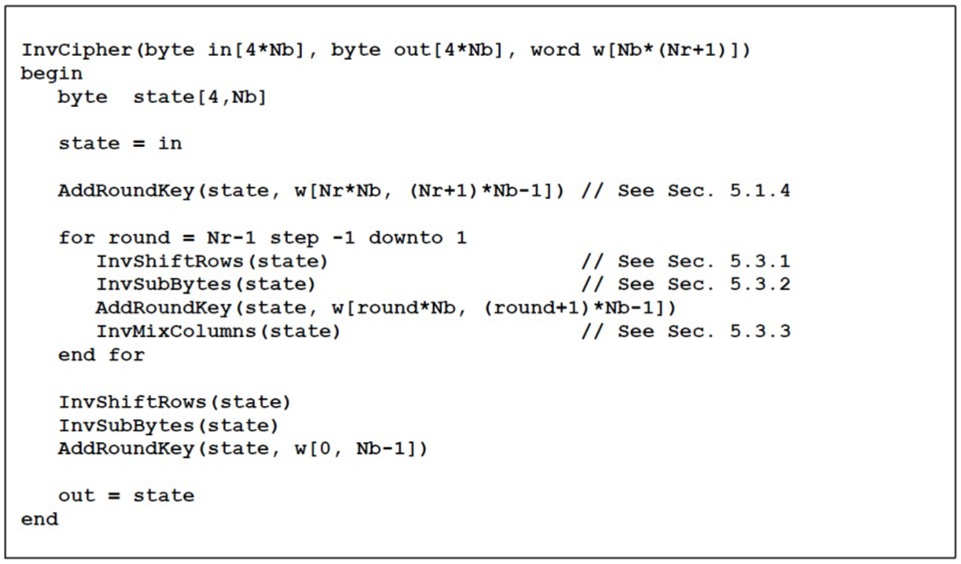
2.2.2 AES Engine Optimization

The NIST selected the Rijndael algorithm for AES because it offers a combination of security, performance, efficiency, ease of implementation, and flexibility. Specifically, Rijndael appears to be consistently a very good performer in both hardware and software across a wide range of computing environments regardless of its use in feedback or non-feedback modes. Its key setup time is excellent, and its key agility is good. The very low memory requirements of the Rijndael algorithm make it very well suited for restricted-space environments, in which it also demonstrates excellent performance. The Rijndael algorithm operations are among the easiest to defend against power and timing attacks. Additionally, it appears that some defense



```
Cipher(byte in[4*Nb], byte out[4*Nb], word w[Nb*(Nr+1)])
begin
   byte  state[4,Nb]

   state = in

   AddRoundKey(state, w[0, Nb-1])            // See Sec. 5.1.4

   for round = 1 step 1 to Nr-1
      SubBytes(state)                        // See Sec. 5.1.1
      ShiftRows(state)                       // See Sec. 5.1.2
      MixColumns(state)                      // See Sec. 5.1.3
      AddRoundKey(state, w[round*Nb, (round+1)*Nb-1])
   end for

   SubBytes(state)
   ShiftRows(state)
   AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1])

   out = state
end
```

(a) Pseudo Code for the AES Cipher



```
InvCipher(byte in[4*Nb], byte out[4*Nb], word w[Nb*(Nr+1)])
begin
   byte  state[4,Nb]

   state = in

   AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1]) // See Sec. 5.1.4

   for round = Nr-1 step -1 downto 1
      InvShiftRows(state)                    // See Sec. 5.3.1
      InvSubBytes(state)                     // See Sec. 5.3.2
      AddRoundKey(state, w[round*Nb, (round+1)*Nb-1])
      InvMixColumns(state)                   // See Sec. 5.3.3
   end for

   InvShiftRows(state)
   InvSubBytes(state)
   AddRoundKey(state, w[0, Nb-1])

   out = state
end
```

(b) Pseudo Code for the AES Inverse Cipher

Figure 2.1: Pseudo Code for the AES Algorithm

can be provided against such attacks without significantly impacting the algorithm's performance. Finally, the algorithm's internal round structure appears to have good potential to benefit from instruction-level parallelism.

As the resource-limited IoT chips advance, AES has become the dominant symmetric-key cryptosystem and the key building for the Internet-connected circuits, leading up to a significant share of resource cost. For decades, multiple implementations of Rijndael have been presented, targeting a wide range of circuit design technologies. These implementations use specic Galois Field fixed constant multipliers based on the constant matrix of the associated algorithm, thus resulting in either logic equations or look-up tables being generated to perform the multiplication [91, 27, 164, 129]. Implementations based on logic equations are optimized for area and require a moderate number of logic levels. Implementations based on look-up tables are optimized for speed at the cost of additional logic resources, although the performance of these implementation is highly dependent on the memory system and cache organization and size. These all focus on performance improvement of one round of the AES algorithm without pipelining and parallel structures.

Subsequently, numerous AES core optimizations are proposed and their performance are evaluated [126, 110, 29, 190, 122, 96]. Based on the finite field arithmetic, several high performance designs are proposed using combinational logics [193, 47, 60]. For example, the arithmetic is employed in the computation of the multiplicative inversion in the SB/ISB transformation, and a deep sub-pipelining approach is enabled for the AES algorithm in [193]. Assume that each 128-bit state can be obtained immediately in each round, the proposed encrypter in [193] can achieve 1.52 GBps or 12.56 Gbps throughput on a Xilinx XCV1000 e-8bg560 device with seven substages (SS) in each round unit. In [47], the construction procedure to implement a two-stage pipelining S-Box by using combinational logics is presented.

As compared to the typical ROM based LUT, the presented implementation is capable of higher operational frequency (72.155 MHz) and small size in terms of area occupancy.

Furthermore, since the composite field can be constructed by using different irreducible polynomials, [192] and [117] analyze and compare the complexity of the SB implementation with different constructions. In [192], how the coefficients of the field polynomials affect each block in the composite filed implementation of the SB transformation is illustrated, and then an optimum construction for the AES algorithm is presented. The brief [117] explores all the possible isomorphic mapping for each of the composite field constructions, and employs a new common subexpression elimination (CSE) algorithm to derive the most optimum isomorphic and inverse isomorphic mapping (δ and $I\delta$) with affine and inverse affine (A and IA) transformations.

Different from the research above focusing on AES cores' implementation, the AES performance is further considered on the circuit structural level in [41, 146, 30, 159, 77, 163]. For instance, the four primitive transformations are decomposed, rearranged, and regrouped as new linear and non-linear operations in [41] to provide 1.28 Gbps throughput for 128-bit keys. In [146], the transformations A/IA, SR/ISR and MC/IMC are combined into a single function unit A/SR/MC or IMC/ISR/IA, and the substructure sharing algorithm is applied to reduce the area cost.

In our survey, however, all the previous performance evaluations are based on the assumption that the AES states can be input to the encrypter/decrypter column-by-column immediately, without considering any on-chip bus overhead [81, 68, 67]. In general, the AES algorithm is based on the column-major state processing, but the conventional bus protocols are in the linear-major order and very low-efficient to supply rectangular arrays of bytes. Therefore, the on-chip bus efficiency becomes one of the bottlenecks for the AES algorithm from the system perspective: the speed and power consumption of cipher/inverse cipher processing are influenced by the bus size and transfer mode, the number of hardware instances such as SB/ISB and MC/IMC modules is determined by architectural degree of parallelism, and the delay cycles and pipelining levels are depended on the throughput requirements and the desired clock frequency.

2.3 Related Work On Verification and Performance Evaluation Methodology

How to determine the trade-off between hardware cost and performance in terms of silicon area, bandwidth, and energy consumption is another critical issue for SoC architects. In earlier work, a lot of efforts were put in modeling the bus performance such as transfer latency [181], bandwidth, and wire efficiency [108]. These kinds of models are based on system simplicity and assumption of normal conditions. Actually, it is an unfeasible task to accurately predict the latency of a complicated system by static analysis. For instance, the handshaking or response signal is usually not immediate in a complicated system and the abnormal transfer statuses such as ERROR and RETRY are normally occur in a real on-chip bus. Likewise, the power consumption analysis, such as using some high level power analysis tools [131, 180, 133] and modeling power analysis [90, 23, 49], suffers from much more inaccuracy without the gate-level parameters and toggle activities of internal logic, signals, and IOs. It is very hard to determine with accuracy in the early circuit design stage using high abstraction models.

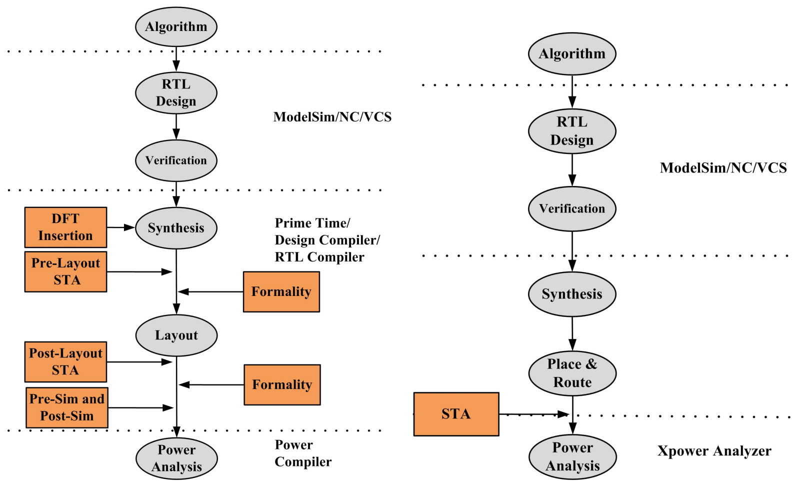
In this context, we create a UVM-based [16, 18] performance evaluation methodology to estimate the chip performance automatically and accurately. This methodology extends the standard integrated circuit design flow, and backward compilable with traditional verification methods, including Open Verification Methodology (OVM), VMM, and UVM verification methodologies, and constrained-random verification, assertion-based verification, and coverage-driven verification methods, and real-data based mixed-signal verification models.

2.3.1 Integrated Circuit Design Flow

FPGAs and ASICs provide different values to designers, and they must be carefully evaluated before choosing any one over the other. AISCs used to be selected for low unit cost and small form factor designs, while FPGAs used to be selected for low speed and low volume designs. The advantages of FPGA based design include:

- The time to market of FPGA development is faster than ASIC without layout, masks, and other manufacturing steps.
- FPGA implementation is better than ASIC implementation when building low-volume production circuits.
- FPGA development cycle is more predictable since its design flow eliminates potential re-spins, wafer capacities, etc, of the project.
- Since FPAG is reprogrammable and reusable, it is low-cost for very low volume industrial devices.

Basically, IoT devices can be sub-divided into two different sectors of application: consumer and industrial. For the consumer IoT and wearable applications, it is essential to study the performance for the ASIC implementation. For the industrial side of things, such as the Internet-connected devices applied in the intelligent industrial system, the machine to machine (M2M) communication system, and the



(a) ASIC Design Flow (b) FPGA Design Flow

Figure 2.2: Integrated Circuit Design Flow.

smart grids for intelligent energy supply, smart health for tele-medicine and remote diagnosis, smart mobility and smart factory, the FPGA based design is commonly used.

The ASIC and FPGA design flows are shown in Figure 2.2(a) and Figure 2.2(b). It can be observed that the FPGA design flow eliminates the complex and time-consuming floorplanning, place and route, timing analysis, and mask/re-spin stages of the project, since the design logic is already synthesized to be placed onto an already verified, characterized FPGA device.

In our work, a set of EDA licenses to support the FPGA design are used, such as Mentor Graphic ModelSim, Xilinx ISE14.6, Virtex5 xc5vlx110t-2ff1136 FPGA device, and Xilinx XPower Analyzer.

2.3.2 Verification Methodology

Design productivity growth continues to remain lower than complexity growth - but this time around, it is verification time, not design time, that poses the challenge.

A recent statistic shows that 60-70% of the entire product cycle for a complex logic chip is dedicated to verification tasks. Verification of complex functions that we can build using new design tools poses a challenge to reduce the total circuit design time.

System Verilog [13, 8] was started to merge a number of disjoint verification languages such as Vera and e that were built as a layer on top of Verilog and VHDL. Each of these languages has their own proprietary methodologies (RVM and eRM) that provide a re-useable framework to construct, configure, and execute tests [112]. Once System Verilog became established, it needed its own methodology. Then, Mentor Graphic created the Advanced Verification Methodology (AVM) in 2006 that was derived from concepts in SystemC [9]. Synopsys converted their Vera-based Reuse Verification Methodology (RVM) library to System Verilog and called it VMM [39, 83]. Then, Mentor Graphic and Cadence joined together and created the OVM in 2008 [70], which was the merging of the existing AVM with concepts from eRM. Finally by 2011, Mentor Graphic, Cadence, and Synopsys joined together through Accellera and created the UVM. More specifically, UVM, OVM, and VMM are introduced as follows.

UVM: UVM is an open source System Verilog library allowing creation of flexible, reusable verification components and assembling powerful test environments utilizing constrained random stimulus generation and functional coverage methodologies. It is a combined effort of designers and tool vendors, based on the successful OVM and VMM methodologies. Its main promise is to improve test bench reuse, make verification code more portable, and create new market for universal and high-quality VIPs [137].

OVM: OVM is the library of objects and procedures for stimulus generation, data collection, and control of verification process. Available in System Verilog and SystemC, OVM allows easy creation of directed or random test utilizing transaction-level communication and functional coverage. As the first System Verilog-based verification library available on multiple simulators, OVM contributed significantly to the development of its successor, UVM.

VMM: VMM is the first successful and widely implemented set of practices for creation of reusable verification environments in System Verilog. Created by Synopsys, one of the strong proponents of System Verilog, VMM harnesses language features, such as object-oriented programming, randomization, constraints, functional coverage to enable both novices and experts to create powerful verification environments. VMM contribution is an important factor in creation of UVM.

In sum, UVM, which is the latest standard ratified by Accellera, is popularly used today in digital integrated circuit design flow. It is based on the System Verilog standard and provides the flexibility and ways to connect the legacy VMM/OVM components. Not only the major EDA vendors participated in the UVM group, but several users from competing companies, such as AMD, Cisco, Free scale and Intel, collaborated.

2.3.3 Verification Methods

In addition, there are also some verification methods widely used in industry, such as coverage-driven verification, assertion-based verification, and constraint-random verification.

Coverage-Driven Verification: The term "functional coverage" is used to describe a parameter that quantifies the functional space that has been covered, as opposed to code coverage that quantifies how much of the implemented design has been covered by a given test suites [46, 95]. Directed simulation can then be used to cover corner test space at the end of the verification cycle. When using coverage-based verification, engineers typically want an estimate of the functional space covered and captured in quantifiable terms [103, 134]. These include:

- Line Coverage: the number of lines of code that were verified.
- Expression Coverage: the number of the logical expressions were tested.
- (FSM Coverage: the number of states in a FSM design were reached.
- Toggle Coverage: the number of ports and registers that were toggled both ways during a simulation run.
- Path Coverage: the number of logical paths in the design code that were covered.

Assertion-Based Verification: Designers use assertions as placeholders to describe assumptions and behavior associated with a design [76]. Assertions get triggered during a dynamic simulation if the design meets or fails the specification or assumption. Assertions can also be used in a formal or static functional verification environment [34]. Assertions and properties can be made to work in the background during static functional verification at the module level and can be reused in a dynamic simulation environment at both module and system level. They are also useful if the module is going to be turned into IPs because the assertions will constantly check the IPs' properties when it is reused [161].

Constraint-Random Verification: Constraint-random verification offers a highly effective way to deal with the challenges of SoC verification [33, 170]. These challenges are overwhelming for many reasons: complex instruction sets, multiple pipeline stages, in-order or out-of-order execution strategies, instruction parallelism, fixed- and floating-point scalar/vector operations, and other features that create a seemly never-ending list of corner cases to exercise. Because constraint-random verification can automatically generate a large number of test cases within the parameters specified by the verification team, it can hit corner cases that neither the design nor verification engineers would have ever anticipated. Without constrained-random stimulus, the bugs lurking in these corners hide until late in the development cycle, or are not found at all until customer usage [80].

2.3.4 Mixed-Signal Verification

Today, there are various efficient, reusable, and reliable functional verification methodologies available for digital circuits. Verification done using these methodologies ensures 99.99% functional correctness of digital design, but same does not hold true when it comes to analog and mixed-signal SoCs. Due to the increasing in mixed-signal chips, there is a potential need for methodology or flow to provide similar confidence on functional verification and constraint-random tests as seen for digital SoCs. Some flows or methodologies being used to verify analog or mixed-signal designs are mentioned below.

Low Level Non-Functional Behavioral Models

- Digital Verification: using current standard verification methodologies.
- Analog Verification: verification using circuit simulators [172, 154].
- Mixed-Signal Verification: connectivity testing between digital and analog designs using very low level analog behavioral models [156, 32, 106].

Analog Functional Behavioral Model Developed in Verilog, VHDL, or Verilog AMS Language

- Digital Verification: using current standard verification methodologies.

- Analog Verification: using spice or fast spice analog circuit simulators. Analog functional verification done using behavioral model developed in Verilog or Verilog AMS using analog mixed-signal simulators [150, 165].
- Mixed-Signal Verification: verification using behavioral analog models [182, 42, 42].

The earlier verification methods sperate analog environment on the left and digital environment on the right, depending on which engineering group was responsible for final assembly. But today's mixed-signal designs have multiple feedback loops, complex modeling requirements, and higher performance targets, meaning it is no longer possible to de-construct designs into separate analog and digital functions. Engineers must embrace the digital-centric metric-driven verification methodologies into their mixed-signal verification flows.

More important, engineers also need an integrated mixed-signal verification environment that focuses on performance and reliability [119, 26]. Verification planning translates into coverage, assertion-based checking, and score-boarding, as well as appropriate stimuli generation for both digital and analog components. The planning process is always important for verification, but for mixed-signal it is critical. The sheer range of operating parameters and complex interaction of analog and digital units requires a clear definition of relevant metrics defined and measured for all units.

The mixed-signal verification system proposed in [185] leverages real-number modeling so that users can perform top-level verification of their analog or mixed-signal designs using discretely simulated real number models. Real-number based system provides the digital equivalent models of analog blocks, enabling engineers to verify a full-chip SoC using only a digital simulator. It eliminates relatively slow analog simulation and convergence issues, allowing for nightly regression runs of the mixed-signal SoCs. It can also integrate with other advanced verification technologies, such as assertion-based verification and metric-driven verification without having to interface with the analog engine or defining new semantics to deal with analog values [185, 174]. Using the mixed-signal verification system, engineers can greatly enhance the top-level verification performance of the overall verification process.

2.4 Related Work On Computer Aided Design Automation

In order to meet the tight time-to-market constraints and to effectively handle the design complexity, it is essential to provide a computer-aided design methodology support for automating this task. In this dissertation, advances in the IoT-style bus protocol is proposed in [186] and [178] for the AES-encrypted circuits, in order to enhance the system performance with the overhead cost of security applications. However, three new challenges are created for a new era of IBUS based design: the third-party IP integration, a flexible UVM based verification environment, and automatic performance evaluation.

2.4.1 IP Integration

Today, designers are increasingly incorporating third-party standards-based IP in their designs, but are still facing several challenges, such as lost protocol expertise, connecting the PHY and Controller, issues around clock and reset, debug and testability, verification and implementation, and integration of performance evaluation models. With the increasing number of IP included in the design and with each of the IP becoming more and more complex, the effort to integrate all of the IPs is similar to the cost of IP, as shown in Figure 2.3. Especially for a new archi-

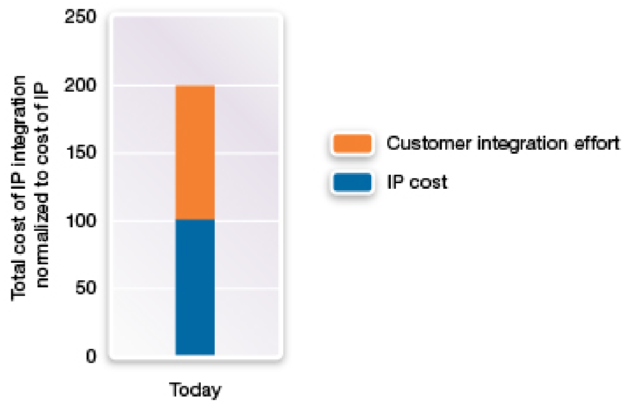


Figure 2.3: The effort to integrate is similar to the cost of IP. [73]

tecture, creating SoCs with IPs under different specifications is a big challenge to system performance and chip complexity, leading to long design cycle and high chip tape-out risk, and even large chip area and power overheads. This is one reason why standard buses are still the predominant architecture of choice in many IoT embedded chips.

Generally, IP integration can be implemented using static bus bridges [44, 75, 118]. Since the static approaches are inherently non-scalable and limited in the ability to provide high performance in cases where the traffic characteristics vary dynamically, a number of automatic design approaches are further proposed to dynamically optimize the bus-based architectural topology [98, 52, 151]. In addition, design methodologies and design flows for customizing these bus architectures to adapt to traffic characteristics, are also studied [79, 48, 191]. While many of these research aim at exploiting theories on system level, they do not adequately address the realization in specific integrated circuit design flow.

2.4.2 Flexible Verification Environment

Verification remains the most significant bottleneck in getting advanced SoCs to market. The development of an independent verification plan, protocol expertise, and efficient use of VIPs are keys to minimizing functional bugs. Creating a reconfigurable verification environment with UVM-compliant building blocks takes less time, eases cross-site collaboration, and maximizes test bench reuse for future projects. It includes test bench generation, constraint random test cases, VIP initialization tasks, integration of protocol-specific VIPs, and integration of performance evaluation models.

Additionally, when looking at the design and verification methodologies in place today, verification is a prime candidate for closer inspection. Conventional verification languages, such as System Verilog and SystemC [13, 82, 9], verification methodologies, such as OVM, VMM, and UVM, [70, 39, 83, 16, 18], and verification methods, such as coverage-driven verification [103, 69], assertion-based verification [76, 34], and constraint-random verification [85], can be certainly reused in the IoT chip design. Moreover, the cost of verification has been rising faster than design and it has been identified as one of the areas in which new solutions may be appropriate for the types of design seen on the edge of the IoT. For example, the pre-verified mixed-signal VIPs and software-controlled initial process can be easily plugged in the UVM/VMM environment to help engineers reduce verification efforts [178, 185].

2.4.3 Automatic Performance Evaluation

Third, the bus architecture for the application-specific designs should closely match the chip performance requirements. Hence, designers need a coherent and systematic approach to quickly evaluate a system with selected IPs and communication architectures.

An earlier stage performance evaluations, such as transfer latency [181] and wire efficiency models [108, 88], and several high-level power analysis tools [133, 64, 90, 171, 84], allow engineers to design circuits more efficiently, reducing the time costs and risk of error involved in building circuit prototypes. For these mathematical analysis models, large system complexity is a challenge, making the enumeration of the complete design logic difficult, sometimes an unfeasible task. Additional problems arise from the high abstraction level, making it hard to achieve high accuracy, in early system development stages. In order to help architects to estimate the chip performance accurately and automatically, a performance evaluation methodology is also proposed in our work [178, 179]. By modeling and collecting several performance metrics, including bus latency, bandwidth, valid bandwidth, power and energy consumption, and slice and energy efficiency using this methodology, it enhances fidelity of the performance analysis and evaluation.

To our best knowledge, no previous works have addressed how to accelerate, optimize, and runtime monitor the entire SoC design flow from front-end to back-end. We combine and augment two previously published works [186] and [178], putting them into context with one another and presenting much more contributions. It focuses on co-exploration of configurable SoC design and system evaluation in real time, as well as the speedup of design cycles by using the IBUS architecture and VIP factory. The platform generates the bus structure with detailed algorithms, involving wrapper insertion, bus width, burst size of a transfer, frequency, and arbitration. Then, we concentrate on providing much higher level of VIPs, entire test bench, and verification environment. 90% to 95% of the device is standardized so that designers can concentrate on the value added functionality and its interactions with the other pieces. This combination delivers a complete IoT chip design solution, from the RTL design to performance estimation, that enables design and verification teams to get a handle on skyrocketing verification costs and bring compelling products to market faster than ever.

2.5 Summary

In this chapter, we introduce the fundamentals of our research and discuss about the related works from a variety of perspectives. We first give a brief introduction on standard on-chip bus architectures commonly used in industry. Traditional bus structures are not suitable for the future IoT circuits and there is a great need to study IoT chip security and privacy. Hence, we present a novel low-cost and low-power bus protocol, and further consider the structural optimization for the security algorithms. Next, we discuss about the related works that have been conducted on IoT embedded chips in standard circuit design flow, including SoC verification, performance evaluation, and design automation from the perspective of integrated circuit design objectives. Finally, design automation based on two top-tier design objectives are presented, IP integration and mixed-signal modeled verification methodology.

In this dissertation, our objective is to develop efficient and effective SoC architecture for high performance and high security IoT embedded chips based on industrial standard circuit design flow, such that the designs can be optimized, fully verified, evaluated, and auto-generated. In the following four chapters, i.e. Chapter 3, 4, 5, and 6, respectively, we present our contributions. In Chapter 7, we conclude this dissertation.

CHAPTER 3

IBUS ARCHITECTURE

This chapter presents a high performance IoT SoC bus architecture termed IBUS. Considering the inevitable trade-off among area, throughput, and energy efficiency, the control bus (IC bus) is developed as a low-cost and low-power bus, and the data bus (ID bus) is created as a high-throughput full-duplex bus with the feature of block data transfer. In order to evaluate the bus performance, we create four analytical models, including transfer time consumption, wire efficiency, valid data bandwidth, and dynamic energy efficiency. Then, the AHB-, AXI- and IBUS-based DMA are developed as a case study. It is observed that IBUS DMA costs less hardware resource and achieves higher performance, especially in the block transfer mode. For instance, the results from both the analytical models and the practical tests show that the time consumption of IBUS is close to 63% of the AXI, the wire efficiency and valid data bandwidth of IBUS are almost 2.3 and 1.6 times of the AXI respectively, and the energy consumption is a half of AXI in the block transfer mode.

3.1 Related Work

Today, the reduced interface complexity and low energy on-chip bus is receiving lots of attention. In industry, the most commonly used bus protocols are the AMBA Advanced High-Performance Bus (AHB) [1] and Advanced eXensible Interface (AXI) [5] from ARM Holdings, Wishbone from Silicore Corporation [6], OCP from OCP-IP [4], CoreConnect from IBM [2] and STBus from STMicroelectronics [7]. All of these buses transfer data linearly, however, in some specific applications such as image processing, computer vision and wireless communication, data processing is usually based on the relationship of data neighbors, adjacency, connectivity, regions and boundaries [71, 3], and block data load and store [15]. In these cases, we prefer data transfers by matrix or block rather than by linear burst. Moreover, for some advanced bus structures such as multi-bus [98, 100, 135] and multi-layer architectures [107, 147], the bandwidth can be improved when most of the communications occur in the same bus level or the same bus layer. However, a large number of wires and internal logic such as multiplexers for different layer data conversion, and buffers or FIFOs for data flow control are necessary, which are more costly in terms of both area and energy consumption.

To overcome the aforesaid limitation, a low-cost and low-energy bus is proposed for limited resource IoT embedded chips in this chapter. It balances performance with cost and implements the features required for low-power and high-throughput. In addition, the bus-performance features including valid data bandwidth and dynamic energy efficiency proposed in this chapter, along with the conventional metrics of transfer time consumption [181], wire efficiency [108, 88], dynamic power and energy consumption [23, 49] are performed as analytical models in this chapter. They are used to analyze the bus performance and compare to the experimental results after the hardware implementation. The rest of the chapter is organized as follows: the section 3.2 introduces the IBUS protocol, based on the definitions of bus performance metrics, section 3.3 presents the analytical models used to evaluate bus standards. In section 3.4, the register transfer level (RTL) design, simulation, synthesis and power analysis are illustrated. Both the experimental and modeling results are compared in section 3.5. Finally, section 3.6 concludes this chapter.

3.2 IBUS Protocol

IBUS is composed of a control bus (IC bus) and a data bus (ID bus). IC bus stands for Master-bus with a single masterthe microprocessor, and ID bus stands for Slave-bus with a single slavethe memory controller.

3.2.1 IBUS Structure

This section presents an on-chip data communication standard for designing low-power and high-speed microcontrollers. In what follows, the architectural performance is statically analyzed.

As shown in Figure 3.1, a typical IBUS architecture consists of a high-performance data bus (ID), able to sustain the memory bandwidth, on which the micro-processor, application-specific devices, DMA and memory reside. ID bus provides a high-bandwidth interface between the elements that are involved in the majority of transfers. The role of the DMA in this architecture is to control which master has access to ID bus and to operate the data transfers between masters and memory. Also located on the architecture is a lower bandwidth control bus (IC), where all the functional register configuration modules, such as SoC peripherals, system control modules, and all the application-specific devices, are located.

IC bus mainly takes charge of low-speed and low-bandwidth functional register operations with a low-cost interface and minimal power consumption. Key features of IC bus include: a) a reduced interface complexity (69 wires for IC bus, 103 wires for AMBA 3 APB protocol, and 139 wires for AHB), b) a single-master bus only used as functional register configuration (multi-master for AHB and AXI), c) SINGLE transfer mode with at least one-cycle command and one-cycle data, and un-pipelined protocol (BURST transfer mode and pipelined protocol for AHB and AXI), d) a

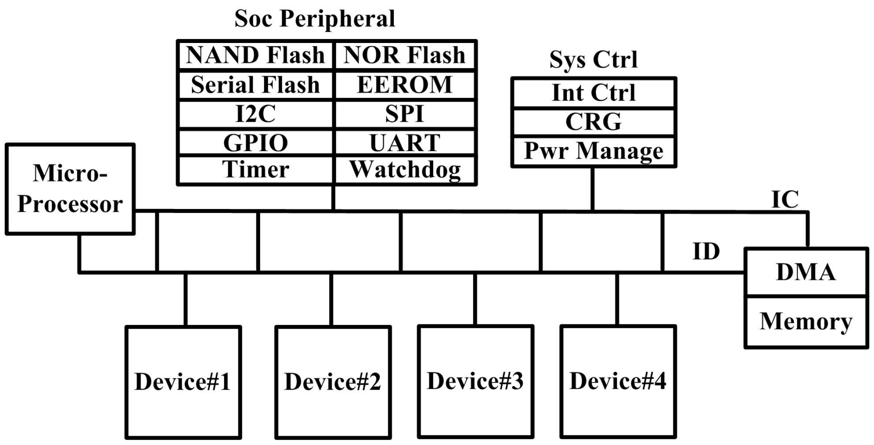


Figure 3.1: IBUS Architecture.

Table 3.1: 32-bit IC Bus Signal

Name	Source	Description
ic_en	Micro-processor	When high it indicates that the micro-processor sends a IC bus command.
ic_wr	Micro-processor	When high it indicates a write transfer and when low a read transfer.
ic_addr_wdata[31:0]	Micro-processor	It indicates address at the command stage, or write data used to transfer data from masters to slaves at the write data stage.
ic_rdata[31:0]	IC bus slaves	It is used to transfer data from slaves to masters during read operations.
ic_vld	IC bus slaves	When high it indicates that a data transfer has finished. It may be driven low to extend a transfer.

half-duplex bus with low band-width and low power consumption (a half-duplex bus for AHB, and a full-duplex bus for AXI). All the IC bus signals prefixed with "ic_" are described in Table 3.1. Notice that the "ic_addr_wdata" signal is created as a shared bus with write address, read address, and write data information. It increases wire usage efficiency and simplifies the hardware interconnection.

ID bus is mainly responsible for high-throughput data transfers. It provides a novel block transfer mode, and backward supports the existing linear mode as well. As an example, a full description of each of the 32-bit data bus signals prefixed with

Table 3.2: 32-bit ID Bus Signal

Name	Source	Description
id_req_x	ID Masters	When high it indicates that the master requests ID bus occupation.
id_gnt_x	DMA	When high it indicates that the request has been granted by DMA.
id_addr[31:0]	ID Masters	The 32-bit address of DBUS.
id_wr	ID Masters	When high it indicates a write transfer and when low a read transfer.
id_len[11:0]	ID Masters	The id_len[11:10] signal determines the transfer mode, and the id_len[9:0] gives the transfer size.
id_wdata[31:0]	ID Masters	It is used to transfer data from masters to DMA during write operations.
id_wdata_vld[3:0]	ID Masters	When high each bit indicates the related valid byte of the write data.
id_rdata[31:0]	DMA	It is used to transfer data from DMA to masters during read operations.
id_resp[1:0]	DMA	When high the id_resp[1]/id_resp[0] signal indicates that a write/read data transaction has finished. It may be driven low to extend a transfer.

"id_" can be found in Table 3.2. Every ID bus master has a pair of "id_req_x" and "id_gnt_x" interfaces to the DMA arbiter to ensure that only one master has access to the bus at any one time. The DMA arbiter performs this function by observing a number of different requests to use the bus and deciding which is currently the highest priority master requesting the bus. The write data channel includes "id_wdata" and "id_wdata_vld" signals. Each bit of the "id_wdata_vld signal indicates that the related-byte of the write data is signaling valid. The bit width of the "id_wdata_vld" signal is 1, 2, 4, 8, 16, respectively, for the byte, half-word, word, double-word, quad-word write data channel. The "id_resp[1:0]" signal indicates that the slave is ready to accept the command and associated data, "id_resp[1]" for write and "id_resp[0]" for read.

As well as the transfer type each transfer will have a number of command signals that provide additional information about the transfer. The "id_addr" signal gives the address of the first data in a transfer, and the “id_wr” signal indicates the transfer direction, logic one for write and logic zero for read. Two transfer modes, linear and block, are supported by the data bus in our work, which are differentiated by the two most significant bits of the data size signal “id_len”. The transfer mode is indicated as the linear mode when the “id_len[11:10]” signal is binary logic “2’b00”, and the block mode when logic “2’b01”. In the linear mode, the signal “id_len[9:0]” gives the exact number of transactions in the row-major order. Notice that the number of transactions in a linear transfer is not the number of data bytes. The total amount of data bytes in a linear transfer is calculated by multiplying the number of transactions by the bus width (in bytes). Let DS denote the bus size parameter. The DS values of 0, 1, 2, 3, and 4 represent the bus width as byte, half word, word, double word, and quad word, respectively. Then, the total number of data bytes in a linear transfer is

$$NDB\_L = id\_len[9:0] \ll DS. \tag{3.1}$$

Here, the shift operators “≪” and “≫” perform left and right shifts of their left operand by the number of bit positions given by the right operand, respectively. For a block transfer, the “id_len[5:0]” signal represents the block height and the “id_len[9:6]” signal represents the block width in the row-major order. Hence, the total number of data bytes in a block mode is

$$NDB\_B = (id\_len[9:6] \ll DS) \times id\_len[5:0]. \tag{3.2}$$

3.2.2 IBUS Transfer Modes

As a control bus for functional register configuration, IC bus defines the SINGLE transfer mode with at least one-cycle command and one-cycle data. It is optimized

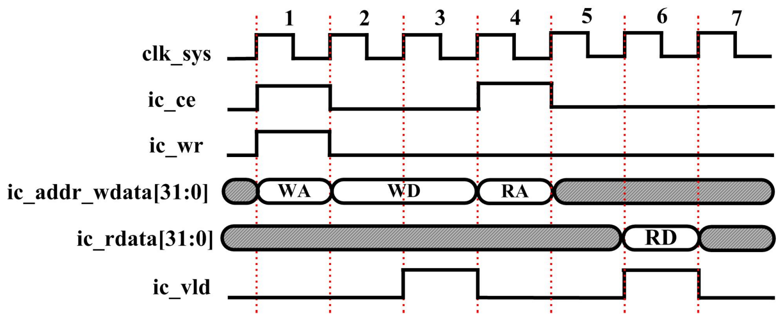


Figure 3.2: IC Bus Protocol.

for minimal power consumption and reduced interface complexity. As shown in Figure 3.2, “ic_addr_wdata” is created as a shared bus with write address, read address, write data information. It increases wire usage efficiency and simplifies the hardware interconnection. Second, IC bus does not require arbitration due to the single-master structure, so the command stage takes only one master cycle. Third, the valid signal (“ic_vld”) used to acknowledge the request is necessary to synchronize signals crossing between master and slave clock domain and avoid command FIFO overflows. Finally, a response delay timer is defined in the IC bus protocol to detect command errors. If the current response is a timeout, the command is indicated as “ERROR” and must be “RETRIED” or “DISCARDED” by the master.

Figure 3.3 shows an example of the timing diagram of ID bus. A pair of handshake signals, “id_req” and “id_gnt”, ensure that there is only one master occupying the write or read channel at the same time. The other ID bus signals are categorized into five packets: command, write data, write data mask, read data, and response. The command packet includes transfer direction, size, and initial address information. The write data mask packet indicates the valid byte of the current word-unit write data, and the respond packet indicates that the current write data is ready or the read data is valid.

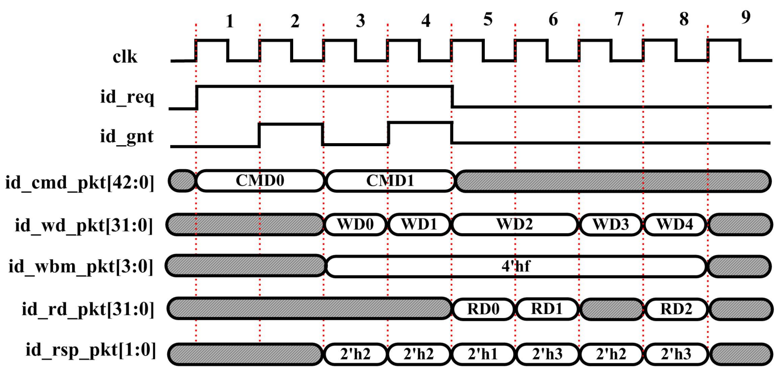


Figure 3.3: ID Bus Protocol.

Notice that ID bus provides the command preprocessing scheme in order to avoid time consumption of multiple command stages. As an example shown in Figure 3.3, the second read data command (CMD1) is granted and preprocessed before all the write data finish. It shares the same cycles with the first and second write data (WD0 and WD1). In addition, ID bus is created as a full-duplex bus according to the time-division multiplexing (TDM) method. As shown in Figure 3.3, the third and last write data (WD2 and WD4) overlap with all the read data (RD0, RD1 and RD2) cycles. Moreover, in cycle 5, the signal of write data valid indicated by "id_rsp_pkt[1]" is de-assert, so the slave cannot receive the third write data (WD2) immediately and the master should hold the write data in the next clock cycle. In cycle 7, the read valid signal indicated by "id_rsp_pkt[0]" is de-assert, so the read data on ID bus is "INVALID" or "DO NOT CARE", the master needs to wait another cycle for a valid read data.

3.2.3 Linear and Block Transfer Modes

Apart from traditional linear data transfer, ID bus supports data transfer by block, which is indicated by the most significant bit of data size signal "id_len[10:0]". If "id_len[10]" is logic 1, the current transfer is a block transfer, "id_len[9:6]" denotes

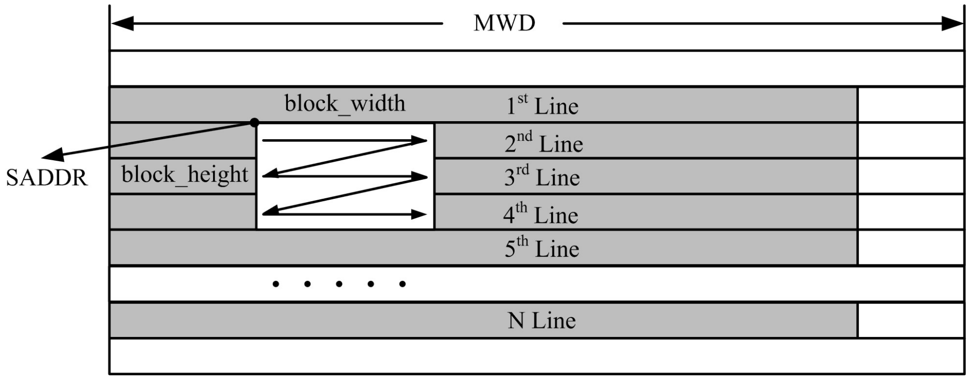


Figure 3.4: ID bus Block Transfer.

the width of the block and "id_len[5:0]" denotes the height of the block. Otherwise, it is a linear transfer and all the other 10-bit signals denote the total transfer size.

As an example of the ID bus block transfer shown in Figure 3.4, SADDR, MWD, and DS parameters are defined in the ID bus protocol. SADDR represents the initial address of this block, MWD denotes the address gap between the data of the vertical neighbors, and DS indicates the ID bus transfer size. The DS value of 0, 1, 2 and 3 represent data transfer as byte, half word, word, and double-word, respectively. Therefore, the initial line start address "line1_addr" is:

$$line1\_addr = (SADDR \ll DS) \gg DS. \tag{3.3}$$

Likewise, the start address of the Nth line "lineN_addr" is:

$$lineN\_addr = [(SADDR + n \times MWD) \ll DS] \gg DS. \tag{3.4}$$

For conventional linear buses, additional commands are required for each non-linear boundary operation of memory. However, ID bus defines all the block boundary-crossing addresses and transfer size with the initial command, thus only one command stage is required for each ID bus block transfer.

3.3 Analytical Models

Below are the bus performance metrics time consumption, wire efficiency, valid data bandwidth, and EE as formulated in this chapter.

3.3.1 Transfer Latency

Transfer time consumption is defined as the total data transfer cycles multiplied by the clock period or the reciprocal of clock frequency denoted by f hereafter. In order to focus on the bus structure, assume that the response to any bus transfer is always available immediately. In addition, request, grant, and address transactions can be overlapped between two back-to-back transfers. Let N and HS denote the number of data and burst size respectively, and P denote the probability of the pipelined transfers. The AHB linear transfer latency denoted by TC_{HL} is

$$TC_{HL} = \{[3 \times ceil(\frac{N}{HS}) + N] - 3P \times ceil(\frac{N}{HS})\} \times \frac{1}{f}. \tag{3.5}$$

In this equation, the "ceil(x)" function means that rounds fraction up. Moreover, the transfers in the same line of a block can be considered as linear transfers. However, in the case of memory boundary-crossing, the address phases as well as bus arbitration are necessary. Let the functions N(ln) and P(ln) denote the data number and the pipelined transfer probability of each line respectively, and HT denote the height of the block. Thus, the communication time of the AHB block transfer denoted by TC_{HB} is

$$TC_{HB} = \langle \sum_{ln=1}^{HT} \{3 \times ceil[\frac{N(ln)}{HS}] + N(ln) - 3P(ln) \times ceil[\frac{N(ln)}{HS}]\}\rangle \times \frac{1}{f}. \tag{3.6}$$

AXI, which is the advanced bus protocol of AHB, has two independent channels of data, handshake, and address, one for read and one for write. Based on this full-duplex bus structure, simultaneous data transferring on both channels is possible.

Therefore, the AXI linear transfer latency denoted by TC_{XL} is

$$TC_{XL} = \langle max(P_r, P_w) \times \{[3 \times ceil(\frac{N}{XS}) + N] - 3P \times ceil(\frac{N}{XS})\}\rangle \times \frac{1}{f}. \tag{3.7}$$

Where P_r and P_w, respectively, denote the data read and data write probabilities. Likewise, the total latency of AXI block transfers denoted by TC_{XB} is

$$TC_{XB} = \langle max(P_r, P_w) \times \sum_{ln=1}^{XT}\{3 \times ceil[\frac{N(ln)}{XS}] + N(ln) - 3P(ln) \times ceil[\frac{N(ln)}{XS}]\}\rangle \times \frac{1}{f}. \tag{3.8}$$

Since the ID bus command stage integrates arbitration and the address phase, and the data stage integrates data transfers and slave-drive responses, ID bus consumes only two-cycle protocol overhead with an immediate grant signal. Let IS denote the data size of the current transfer. Due to provision of the block transfer mode, the total ID bus transfer latency TC_I is the same for both the block and linear mode, that is

$$TC_I = \langle max(P_r, P_w) \times \{[2 \times ceil(\frac{N}{IS}) + N] - 2P \times ceil(\frac{N}{IS})\}\rangle \times \frac{1}{f}. \tag{3.9}$$

3.3.2 Wire Efficiency

As a bus-efficiency metric, wire efficiency is defined as the average number of data bytes per clock cycle divided by the number of bus wires [16]. Let W denote the basic wire number. The wire efficiency is formulated as:

$$WE = \frac{[\frac{N}{(TC \times f)}]}{W} \tag{3.10}$$

3.3.3 Valid Data Bandwidth

Assume the frequencies and bus width (DW) of AHB, AXI, and ID bus are the same, thus the regular bandwidth of ID bus (BW_I) and AXI (BW_X) are the double of the AHB bandwidth (BW_H). All of them are formulated as:

$$BW_H = f \times DW \tag{3.11}$$

$$BW_I = BW_X = f \times 2 \times DW \tag{3.12}$$

Different from the conventional bus throughput, valid data bandwidth is defined as the valid data without protocol overhead that can be transferred in one cycle. Therefore, all of them are modified as:

$$VDB_H = BW_H \times \frac{N}{TC_H \times f} \tag{3.13}$$

$$VDB_X = BW_X \times \frac{N}{TC_X \times 2 \times f} \tag{3.14}$$

$$VDB_I = BW_I \times \frac{N}{TC_I \times 2 \times f} \tag{3.15}$$

3.3.4 Dynamic Energy and Dynamic Energy Efficiency

Basically, energy is the integral of power. Let $P_dy(t)$ denote the dynamic power and T denote the total transfer time, thus dynamic energy consumption is formulated as:

$$DE = \int_{t=0}^{T} P_{dy} \times dt. \tag{3.16}$$

In this equation, if $P_dy(t)$ is constant or average power, the dynamic energy consumption can be simplified as the product of average power and transfer time. Apart from the traditional power and energy estimation, in our work we create a novel parameter, dynamic energy efficiency, to evaluate the correlation between the valid throughput and power consumption, which is formulated as:

$$DEE = \frac{VDB}{P_{avgdy}} \tag{3.17}$$

Here, P_avgdy is the average dynamic power. dynamic energy efficiency provides a performance metric in terms of valid data number can be transferred per second per watt, or valid data number can be transferred per joule.

3.4 Hardware Implementation

This section presents the hardware implementation targeted to accurately estimate the bus performance and verify the validity of the analytical models.

3.4.1 RTL Design and Verification

As shown in Figure 3.5, the AHB-, AXI-, and IBUS-based DMA are designed at the RTL level. As the only slave of ID bus, the ID bus DMA supports both the linear and block transfers, and provides the command preprocessing scheme. There are two separate queues of the ID bus DMA, one write queue and one read queue, and the depth is 4 of each. Thus up to 8 ID bus commands can be preprocessed and pushed into the command queues after winning the arbitration. The commands can be popped as long as the queues are not empty and the Memory Controller interface is idle.

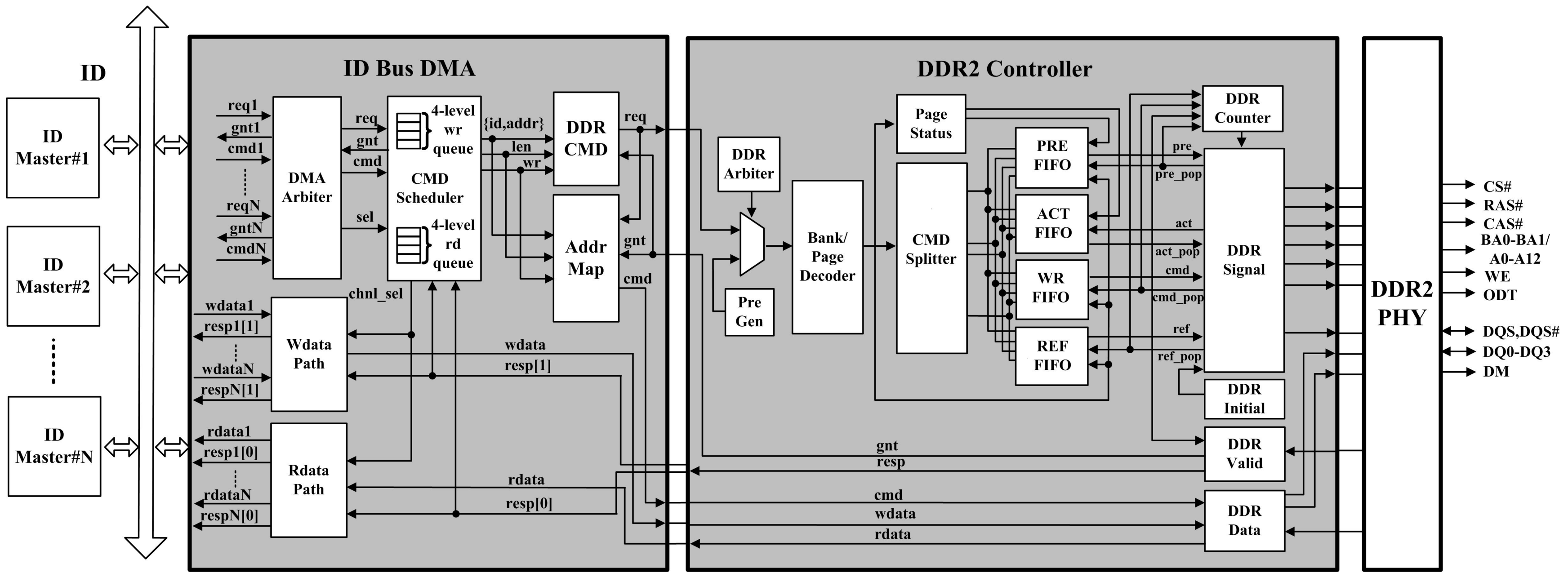


Figure 3.5: DMA and DDR2 Controller.

In order to verify all the bus-based DMA design which can be both RTL-level and gate-level netlist, a UVM [16, 18] environment is set up. As an example of IBUS-based SoC shown in Figure 3.6, several encapsulated, ready-to-use and configurable verification agents are integrated in the test bench. All the six peripherals including NON Flash, UART, I2C, SPI, GPIO, and Timer controllers are IC bus's slaves. They are configured by the microprocessor through IC bus directly. On the other side, all the four application-specific models including Wi-Fi Mac, Bluetooth 4.0 Controller, USB 2.0 Host Controller, and Security module are the slaves of IC bus and the masters of ID bus. They are controlled by the microprocessor through IC bus and access the only slave memory through ID bus. To examine and compare bus performance, we pick two typical test cases as the experimental vectors: one is two 128-word linear write and read operations and the other is two 16×32-pixel block write and read operations. All the data are from a 1024×1024 -pixel picture. In each case, two ID bus master agents request data bus at the same time, and the initial addresses are 0x0000 which is from the top left of the picture and 0x83f0 which is from the bottom right of the picture, respectively. All the simulation information, involving multiple switching activities of signals, logic, and IOs, is captured and saved in VCD files.

To simplify the waveform, only the latencies of the CPU, Wi-Fi, and USB Host Controller operations are shown in Figure 3.7. For IBUS-based SoC signals, notice that the arbitration and commands are preprocessed, the accesses of IC and ID buses are paralleled, and the read and write operations of IBUS are overlapped. All the transfers are finished at 3702ns, which is denoted as the T1 cursor in this figure. For AHB-based SoC signals, the CPU requests are granted firstly due to the highest priority. After register access finishes, the DMA data transfers begin. The write and read data operations are serial on AHB bus, the bus access order is from the

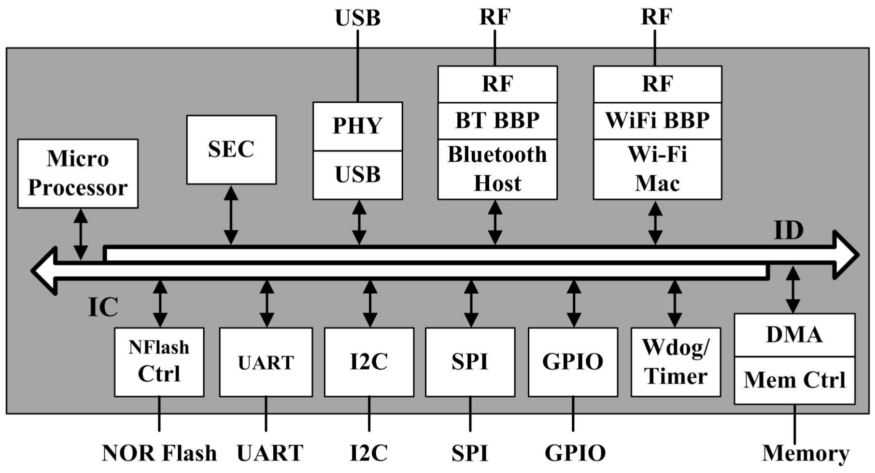
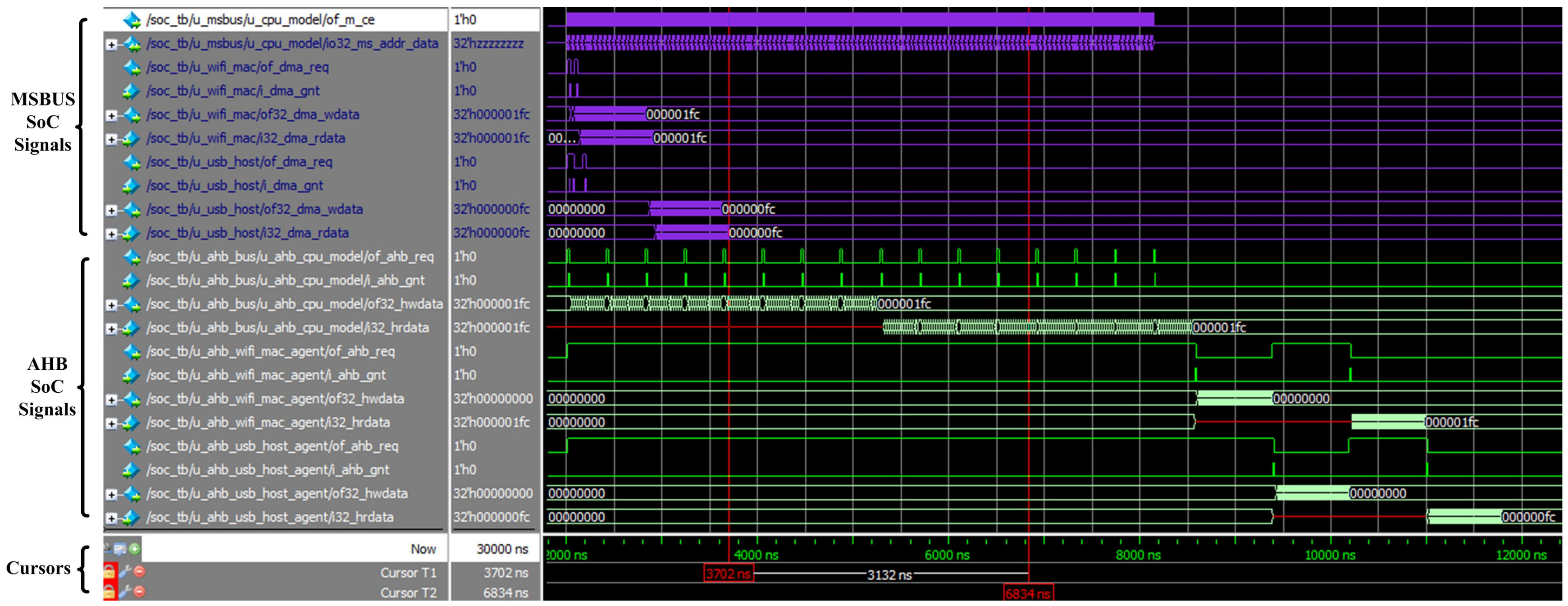


Figure 3.6: A Typical IBUS-Based SoC.

Wi-Fi MAC data write to the lower priority data access, the USB Host Controller data read. All the data transfers are completed at 6934 ns, which are indicated as the T2 cursor.



```
/soc_tb/u_msbus/u_cpu_model/of_m_ce                     1'h0
/soc_tb/u_msbus/u_cpu_model/io32_ms_addr_data           32'hzzzzzzzz
/soc_tb/u_wifi_mac/of_dma_req                           1'h0
/soc_tb/u_wifi_mac/i_dma_gnt                            1'h0
/soc_tb/u_wifi_mac/of32_dma_wdata                       32'h000001fc
/soc_tb/u_wifi_mac/i32_dma_rdata                        32'h000001fc
/soc_tb/u_usb_host/of_dma_req                           1'h0
/soc_tb/u_usb_host/i_dma_gnt                            1'h0
/soc_tb/u_usb_host/of32_dma_wdata                       32'h000000fc
/soc_tb/u_usb_host/i32_dma_rdata                        32'h000000fc
/soc_tb/u_ahb_bus/u_ahb_cpu_model/of_ahb_req            1'h0
/soc_tb/u_ahb_bus/u_ahb_cpu_model/i_ahb_gnt             1'h0
/soc_tb/u_ahb_bus/u_ahb_cpu_model/of32_hwdata           32'h000001fc
/soc_tb/u_ahb_bus/u_ahb_cpu_model/i32_hrdata            32'h000001fc
/soc_tb/u_ahb_wifi_mac_agent/of_ahb_req                 1'h0
/soc_tb/u_ahb_wifi_mac_agent/i_ahb_gnt                  1'h0
/soc_tb/u_ahb_wifi_mac_agent/of32_hwdata                32'h00000000
/soc_tb/u_ahb_wifi_mac_agent/i32_hrdata                 32'h000001fc
/soc_tb/u_ahb_usb_host_agent/of_ahb_req                 1'h0
/soc_tb/u_ahb_usb_host_agent/i_ahb_gnt                  1'h0
/soc_tb/u_ahb_usb_host_agent/of32_hwdata                32'h00000000
/soc_tb/u_ahb_usb_host_agent/i32_hrdata                 32'h000000fc
                                            Now         30000 ns
                                      Cursor T1         3702 ns
                                      Cursor T2         6834 ns
```

Figure 3.7: An Example of AHB and IBUS Transfers.

Table 3.3: Resource Comparison

Resource Cost	AHB	AXI	IBUS
IOs	331	511	302
Slice Registers	10365	11768	10521
Slice LUTs	25272	17494	17300
MOF	344.705	246.875	300.501

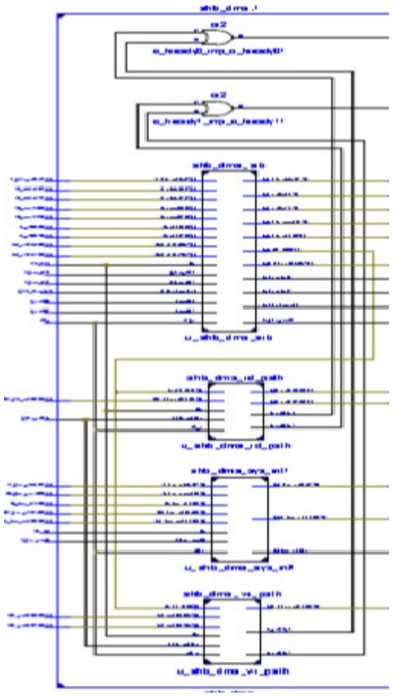
3.4.2 Back-End Power Analysis

Back-end power analysis is a standard step providing average power consumption for the reference implementation. First, a fully placed and routed NCD file and a physical constraint PCF file are generated by Xilinx ISE 14.6 in our work. Figure 3.8 shows the synthesis schematics, and Figure 3.9 shows the slice count of AHB, AXI, and IBUS-based designs with the target device Virtex5 xc5vlx110t-2ff1136 FPGA.

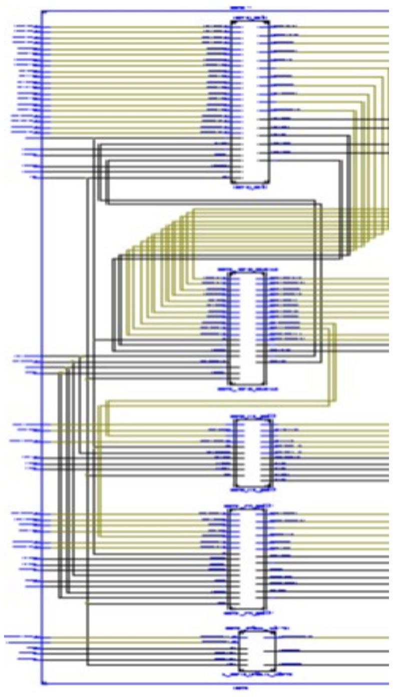
More specifically, Table 3.3 summarized the synthesized results including the maximum clock frequency, IO number, and resource utilization. It is obvious that IBUS-based design costs less IOs, register count, and LUT count than AXI. Notice that the register utilization of IBUS DMA is more than that of AHB DMA, but the LUT count of IBUS DMA is far less than AHB DMA. In addition, the maximum clock frequency of IBUS-based design is 300.501MHz, which is between the maximum clock frequencies of AHB- and AXI-based design. To focus on the bus protocol efficiency, we set the bus frequency to be 100MHz for all the three bus-based design.

Then, by importing the NCD and PCF files, along with the VCD files from the simulation into the XPower Analyzer tool, the detailed average power report (PWR file) is obtained. Figure 3.10 shows the experimental power reports, and Table 3.4 summarizes the power consumption based on different design and test vectors.

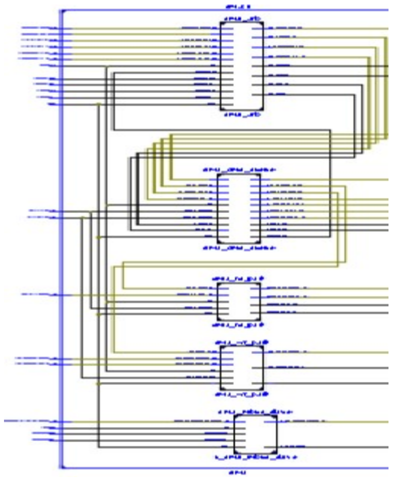
Since the static power is mostly determined at the circuit level, it is almost a constant for different test cases for a given NCD file. Thus our work focuses on analyzing DP. As shown in the third column of Table 3.4, more DP is consumed in



(a) AHB-Based Design

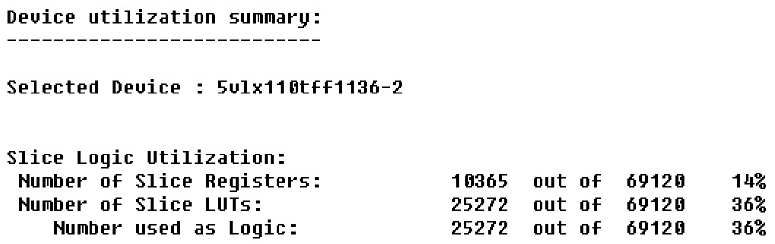


(b) AXI-Based Design



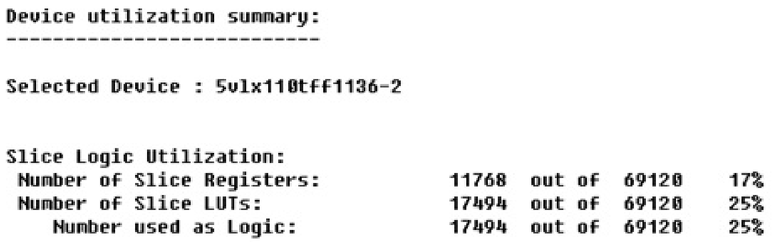
(c) IBUS-Based Design

Figure 3.8: Synthesis Schematics.



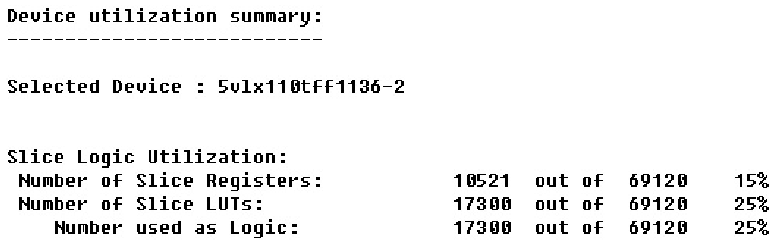
```
Device utilization summary:
---------------------------

Selected Device : 5vlx110tff1136-2


Slice Logic Utilization:
 Number of Slice Registers:          10365  out of  69120    14%
 Number of Slice LUTs:               25272  out of  69120    36%
    Number used as Logic:            25272  out of  69120    36%
```

(a) AHB-Based Design



```
Device utilization summary:
---------------------------

Selected Device : 5vlx110tff1136-2


Slice Logic Utilization:
 Number of Slice Registers:          11768  out of  69120    17%
 Number of Slice LUTs:               17494  out of  69120    25%
    Number used as Logic:            17494  out of  69120    25%
```

(b) AXI-Based Design



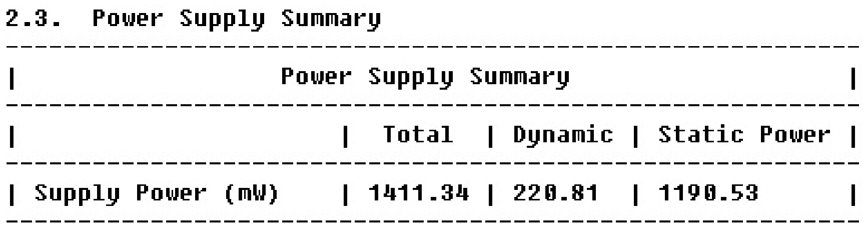
```
Device utilization summary:
---------------------------

Selected Device : 5vlx110tff1136-2


Slice Logic Utilization:
 Number of Slice Registers:          10521  out of  69120    15%
 Number of Slice LUTs:               17300  out of  69120    25%
    Number used as Logic:            17300  out of  69120    25%
```

(c) IBUS-Based Design

Figure 3.9: Resource Costs.

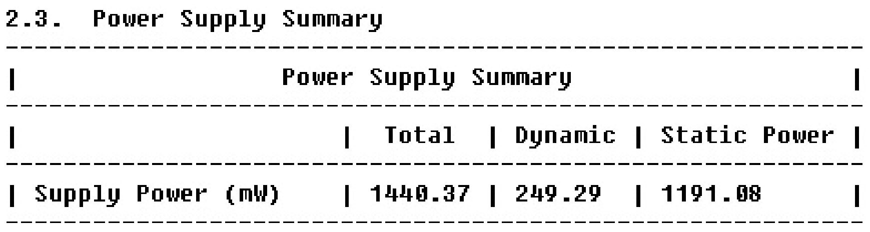
Table 3.4: Power Comparison

Test Cases	SP (mW)	DP (mW)	TP (mW)
HL	1191	221	1411
XL	1195	459	1654
IL	1194	382	1575
HB	1191	249	1440
XB	1195	499	1695
IB	1194	405	1600



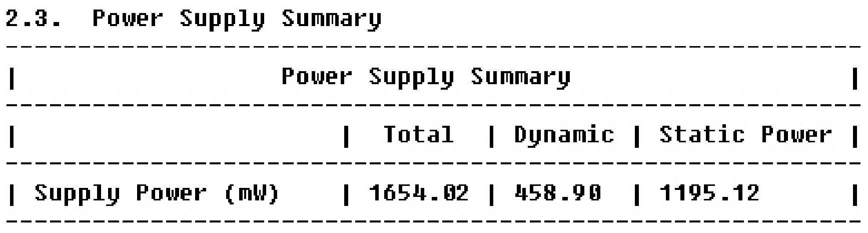
```
2.3.  Power Supply Summary
-----------------------------------------------------------
|                   Power Supply Summary                  |
-----------------------------------------------------------
|                       |  Total   | Dynamic | Static Power |
-----------------------------------------------------------
| Supply Power (mW)     | 1411.34 | 220.81  | 1190.53      |
-----------------------------------------------------------
```

(a) Linear Test of AHB-Based Design



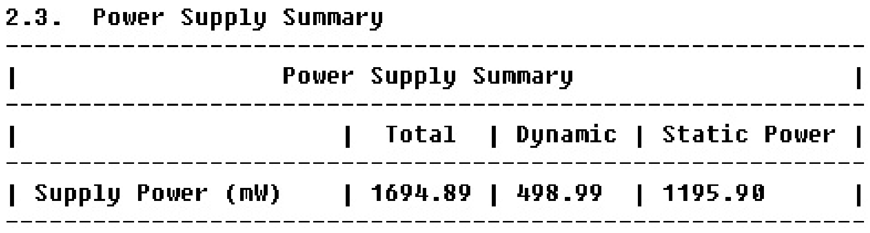
```
2.3.  Power Supply Summary
-----------------------------------------------------------
|                   Power Supply Summary                  |
-----------------------------------------------------------
|                       |  Total   | Dynamic | Static Power |
-----------------------------------------------------------
| Supply Power (mW)     | 1440.37 | 249.29  | 1191.08      |
-----------------------------------------------------------
```

(b) Block Test of AHB-Based Design



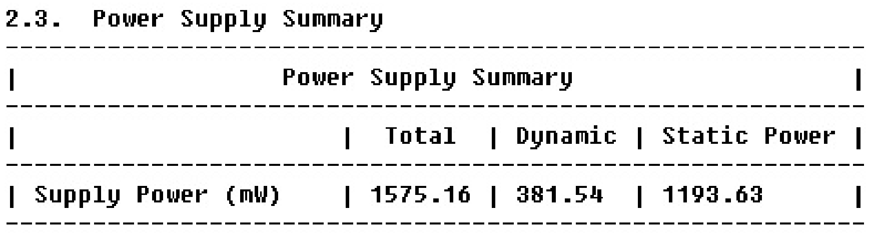
```
2.3.  Power Supply Summary
-----------------------------------------------------------
|                   Power Supply Summary                  |
-----------------------------------------------------------
|                       |  Total   | Dynamic | Static Power |
-----------------------------------------------------------
| Supply Power (mW)     | 1654.02 | 458.90  | 1195.12      |
-----------------------------------------------------------
```

(c) Linear Test of AXI-Based Design



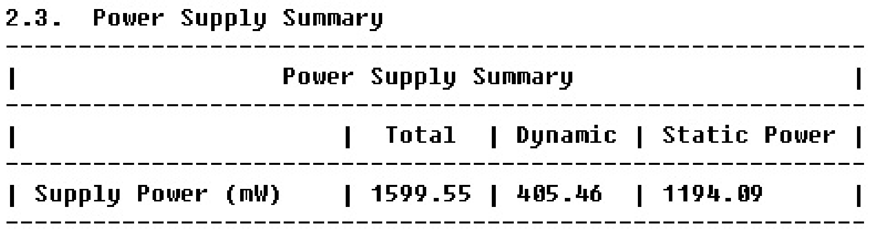
```
2.3.  Power Supply Summary
-----------------------------------------------------------
|                   Power Supply Summary                  |
-----------------------------------------------------------
|                       |  Total   | Dynamic | Static Power |
-----------------------------------------------------------
| Supply Power (mW)     | 1694.89 | 498.99  | 1195.90      |
-----------------------------------------------------------
```

(d) Block Test of AXI-Based Design



```
2.3.  Power Supply Summary
-----------------------------------------------------------
|                   Power Supply Summary                  |
-----------------------------------------------------------
|                       |  Total   | Dynamic | Static Power |
-----------------------------------------------------------
| Supply Power (mW)     | 1575.16 | 381.54  | 1193.63      |
-----------------------------------------------------------
```

(e) Linear Test of IBUS-Based Design



```
2.3.  Power Supply Summary
-----------------------------------------------------------
|                   Power Supply Summary                  |
-----------------------------------------------------------
|                       |  Total   | Dynamic | Static Power |
-----------------------------------------------------------
| Supply Power (mW)     | 1599.55 | 405.46  | 1194.09      |
-----------------------------------------------------------
```

(f) Block Test of IBUS-Based Design

Figure 3.10: Power Consumption.

Table 3.5: Experimental Performance Metrics

Test Cases	TC (ns)	WE	VDB (GBps)	DE (uJ)	DEE (GBps/J)
HL	6105	0.0040	2.68	1.35	12.14
XL	2845	0.0045	5.76	1.31	12.55
IL	2695	0.0077	6.08	1.03	15.91
HB	8985	0.0032	1.82	2.24	7.32
XB	4245	0.0033	3.86	2.12	7.73
IB	2695	0.0077	6.08	1.09	15.01

the block case than in the linear case for the same NCD due to much more toggle activities of IOs, internal logic, and signals. In each case, IBUS consumes less DP than AXI. However, because of the density and high throughput transfer features, the DP of IBUS is much more than that of AHB.

3.5 Experimental and Modeling Results

We measure time consumption using the UVM test bench introduced in subsection 3.4.1, and estimate the wire efficiency, valid data bandwidth, dynamic energy consumption, and dynamic energy efficiency in this section. Table 3.5 summarizes the results of all these metrics.

The difference between these bus protocols is made clear through the analysis of the metric ratios of IBUS to AXI and IBUS to AHB shown in Figure 3.11. All the time consumption ratios are lower than 1. Especially in the block mode, the time consumption of IBUS is reduced to about 30% of AHB and 63.5% of AXI. Considering the wire efficiency next, both IBUS to AXI and IBUS to AHB ratios are more than 1.5 in linear cases and more than 2 in block cases, which is an evidence of the high-efficiency interconnection of IBUS. Although the BW of IBUS and AXI bus are the same, the IBUS to AXI ratio of valid data bandwidth is more than 1 in the linear case and more than 1.5 in the block case. Finally, it is observed that

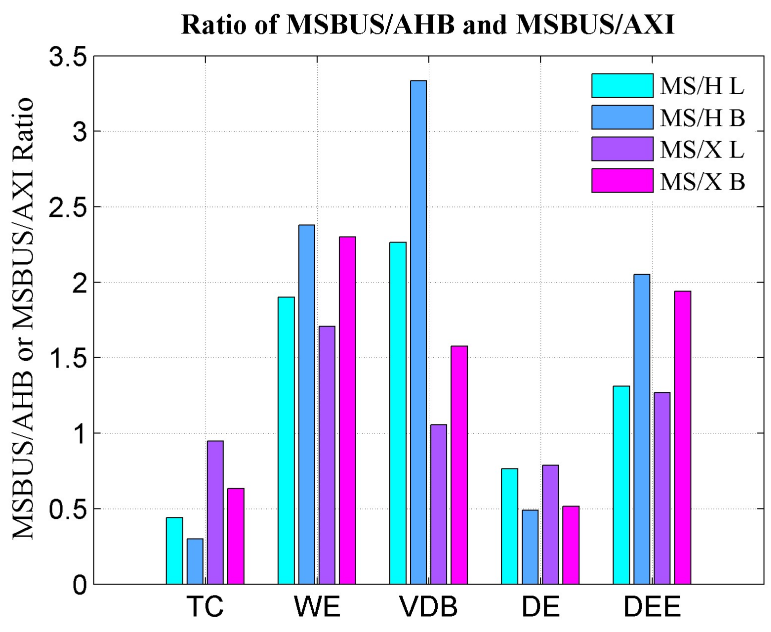


Figure 3.11: Performance Ratio of IBUS/AHB and IBUS/AXI.

Table 3.6: Model Performance Metrics

Test Cases	TC (ns)	WE	VDB (GBps)	DE (uJ)	DEE (GBps/J)
HL	6080	0.0041	2.69	1.34	12.10
XL	2880	0.0041	5.69	1.32	12.39
IL	2600	0.0082	6.30	0.99	16.50
HB	8960	0.0032	1.83	2.23	7.34
XB	3840	0.0032	4.27	1.92	8.55
IB	2600	0.0082	6.30	1.05	15.56

the dynamic energy consumption of IBUS is only a half of AHB or AXI, and the dynamic energy efficiency is close to the double of AHB or AXI in the block mode. In other words, IBUS can transfer twice as much data as AHB and AXI with the same time and power consumption in the block transfers.

For the validity of the performance models, we investigate and compare all the metrics of the three different bus protocols. Table 3.6 shows all the data resulted from the analytical models.

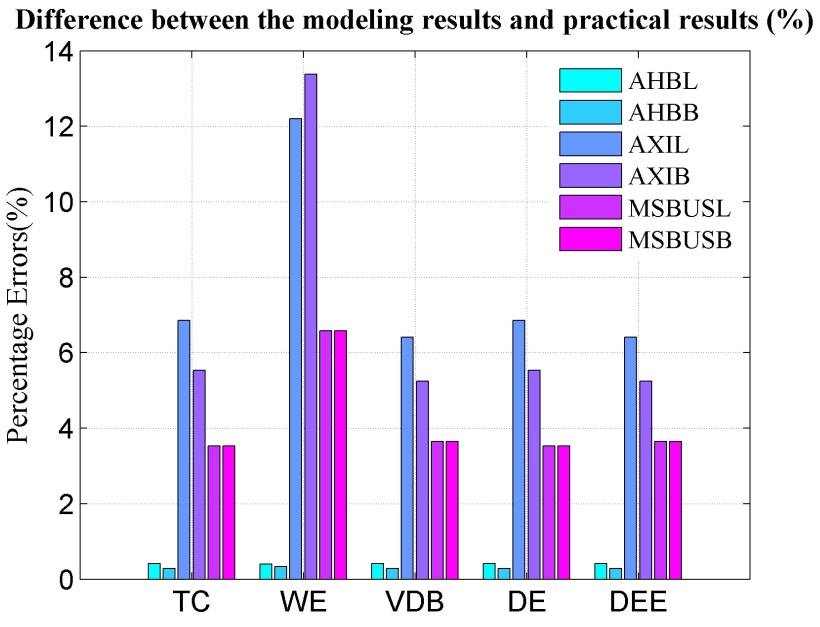


Figure 3.12: Performance Model Error.

Figure 3.12 illustrates the average percent errors of each sample. In the worst case, the estimation error compared with the experimental result is about 13%. Furthermore, the average estimation error is around about 4%. Through the experiments, we verify with sufficient accuracy that all the proposed models can be used effectively to evaluate the bus performance.

3.6 Summary

In this chapter, we propose a high performance bus with reduced interface complexity, minimal power consumption, and high bandwidth. Moreover, we evaluate its performance with four estimation models, such as time consumption, wire efficiency, valid data bandwidth, and dynamic energy efficiency, during the hardware implementation flow automatically and accurately. Comparing AHB, AXI, and IBUS DMA implementations as a case study, both the static analysis and the real hardware results demonstrate that IBUS achieves higher performance especially in the block transfer mode: the wire efficiency of IBUS is 2.3 times of AXI and 2.4 times of AHB, and the dynamic energy efficiency is close to the double of AXI or AHB. The single-processor and multi-client bus structure of IBUS reduces resource utilization and energy consumption, and limits the complexity of circuits. Therefore, the IBUS protocol is very desirable for small scale embedded systems with requirements of a low-cost interface and high energy efficiency.

CHAPTER 4

PERFORMANCE EVALUATION METHODOLOGY

This chapter proposes an innovative on-chip bus transfer mode, the AES state transfer (AS), and a performance evaluation methodology to estimate the transfer performance. By modeling and collecting several performance metrics, including bus latency, bandwidth, valid bandwidth, power, and energy consumption, using the methodology, it enhances fidelity of the performance analysis and evaluation. As a case study, we formally complete the hardware implementation flow on AHB, AXI4, and AS bus (ASBUS) DMA, and demonstrate high estimation accuracy by comparing all the experimental results. Both static analysis and hardware implementation results show that the data transfer latency is close to 29% of AHB and 58% of AXI4 by using the AS transfer. Moreover, it is observed that this high-efficiency transfer mode of ASBUS helps to enhance the valid data bandwidth to around 3.4 times that of AHB and 1.7 times that of AXI4, and the energy consumption of ASBUS is only a half of AHB and AXI4. Furthermore, the proposed evaluation methodology is effectively used with sufficient accuracy (the average estimation error: 3.3%) in the design flow.

4.1 Related Work

Today, much attention has been placed on the IoT technologies. IoT refers not only to personal computers and smart phones connected through the internet, but also to the wireless interconnections of all the billions of things through the Internet or local area network. The main features and requirements of these things are small-scale, power-efficient, high speed, and security. The traditional bus-based SoC architectures [1, 5, 6, 4, 2, 7], such as the AMBA AHB and AXI4 buses from

ARM Holdings, Wishbone from Silicore Corporation, or OCP from OCP-IP, are not suitable for these kind of devices, because they use much more IOs and signals, turning out to be complicated structures and much more power consumption. In addition, all of these buses transfer data linearly, however, in the fields such as image processing, wireless communication, and cryptology, data processing is usually based on the relationship of block neighbors and boundaries [71, 3, 15, 93]. For example, the dominant symmetric-key cryptosystem, AES [19], which is issued by the NIAS in 2001 and widely used in various protocols, is a cipher/inverse cipher algorithm based on the 4×4 matrix of bytes named AES state. A state is operated by 10, 12, or 14 rounds of transformations with key length equal to 128, 192, or 256 bits, respectively. Each round except the final round contains four transformations: SB or S-Box, SR, MC, and AK for the encryption, and ISB, ISR, IMC, and AK for the decryption. Since the state is shifted in each round, we prefer data transfers by interleaving matrix to linear transmission, which is very low-efficiency using conventional buses, such as AHB and AXI4. To overcome these problems, in this chapter, we propose a novel bus transfer mode, the AES state transfer (AS). Furthermore, using the high-efficiency and compact IBUS platform [186], a user-defined bus protocol termed as ASBUS providing the AS transfer mode is proposed in this chapter.

In order to evaluate the AS transfer performance, the widely used industrial bus protocols AHB and AXI4, and the proposed ASBUS are analyzed and compared using static models and implemented as AHB, AXI4, and ASBUS DMA. In earlier work, the AHB bus latency was formulated as the number of clock cycles consumed in transferring data, including request, address, data, and response phases [181]. Together with the wire efficiency proposed in [108], these two parameters are used to estimate AHB and SNP bus performance [108]. Moreover, since power consumption is quickly becoming a first-class concern of circuit design, much effort is put into achieving power dissipation based on either analytical or empirical technologies, such as using some high-level power analysis tools [131, 180, 133], or modeling power analysis [23, 171, 84]. For these mathematical analysis models, large system complexity is a challenge, making the enumeration of the complete design logic difficult, sometimes an unfeasible task. Additional problems arise from the high abstraction level, making it hard to achieve high accuracy, in early system development stages. Although most existing algorithms are shown to be quite effective under their own experimental scenarios, the performance results are inaccurate without real back-end netlist information. In addition, there is limited work on analyzing the correlations among all these bus parameters, and very few studies on the entire bus structure. To address the above issues, this chapter proposes a bus performance evaluation methodology, and presents comparison between results from AHB, AXI4 [1, 5], and ASBUS DMA implementations. In our work, not only traditional bus performance parameters, such as transfer latency, bandwidth, and power, but also valid data bandwidth and dynamic energy consumption are defined and applied to the hardware implementation flow.

In addition, a typical IoT SoC today is inherently analog in dealing with Radio Frequency (RF) signals with the ability to transfer data wirelessly. To set up a verification environment of such a complex mixed-signal system is a big challenge for verification engineers. It not only needs to simulate in a reasonable amount of time, but also maintain an acceptable level of accuracy. To date, the industry has offered several co-simulation tools, such as SPICE and AMS [101, 94]. Since simulations run at the transistor level, these tools are often very time-consuming compared with digital RTL simulation. Aiming at alleviating low-speed problems, two analog and mixed-signal languages, Verilog-AMS and VHDL-AMS, are proposed. They are partial solutions to the problem of simulation speed, but very weak on verification methods, involving coverage driven verification, constraint random test, assertion based verification, and UVM/VMM verification methodologies, in order to reach acceptable standards. Currently, some EDA companies, such as Cadence and Synopsys, provide several analog IPs which are designed by wreal and real data. However, such mixed-signal IPs impose additional amount of work and constraints on how these IPs should be seamlessly integrated and verified at SoC level [176, 76, 69, 69].

As mentioned above, traditional mixed-signal verification technologies are done at the lower level of abstraction. This ensures accuracy and functional correctness, with the idea of higher level language to describe the functional model just emerging. At the SoC level, a test plan is focused on connectivity and functional integration, where simulation speed is really critical. Thus, one of the contributions we have been trying to make is to achieve an optimum trade-off between performance and accuracy at the system level verification, that is, model analog activities to become more like digital, and reuse digital verification environment, technologies and methodologies. Using floating-point real data to describe analog activities, this research brings a high-level abstraction of the analog model by System Verilog verification language (IEEE 1800) [130, 13]. As an extension of Verilog, System Verilog is easier to adopt a modular approach for integrating analog models into an existing pure digital environment. Moreover, being an integrated part of the simulation engine, it also eliminates the need for external verification tools and interfaces, and thus ensures optimal performance. We therefore demonstrate the integration of a mixed-signal SoC, involving both digital and analog components, reasonable and verifiable.

Key contributions of this chapter include:

- Propose and implement a low energy and high throughput AS transfer mode on ASBUS.

- To estimate the AS transfer performance, we propose a high accurate evaluation methodology derived from the standard FPGA design flow.
- Formulate a set of performance parameters, involving transfer time consumption, valid bandwidth, and dynamic energy consumption, which can be seamlessly compatible with the UVM environment [13, 16, 18].
- Model-based design of analog activities by using System Verilog verification language.
- Employ digital verification methods and methodologies in mixed-signal environment, such as coverage driven verification, constraint random test, assertion based verification, and UVM/VMM verification methodologies.

The rest of this chapter is organized as follows: section 4.2 gives a brief introduction on the AS transfer mode and the ASBUS protocol. Section 4.3 defines the time consumption, valid data bandwidth, and dynamic energy models, then analyzes and compares the performance among AHB, AXI4, and ASBUS statically. The performance evaluation methodology based on the hardware implementation flow is presented in section 4.4. Finally, section 4.5 describes and analyzes all the experimental results from AHB, AXI4, and ASBUS DMA designs, and section 4.6 draws the conclusion of this research.

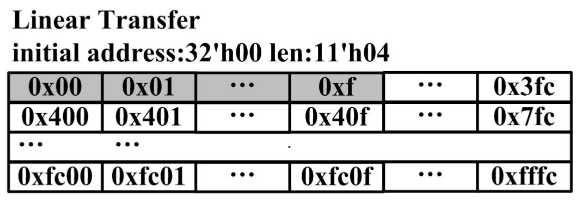
4.2 State Transfer Introduction

In this section, we introduce the features of the AS transfer. Based on this novel transfer mode, then the ASBUS protocol is created and illustrated.

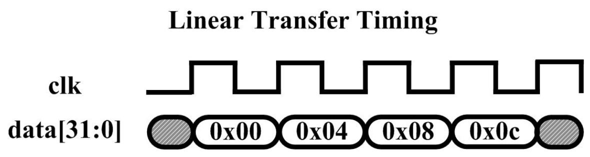
AES is a widely used specification for the cipher security of electronic data established by the U.S. NIAS in 2001 [19, 93]. For both encryption and decryption processing, the AES algorithm operates on a 128-bit or 4×4-byte block, which is called a state. The state is shifted in each round: The first row is not shifted; the second, third, and fourth rows are left shifted one, two, and three bytes for the SR transformation, and right shifted one, two, and three bytes for the ISR transformation, respectively. Since the cyclic rotation does not affect the regrouping result, ASBUS defines the interleaving AS transfer to speed up the AES state transfer and encryption/decryption process.

By redefining the transfer size "len" signal, ASBUS supports both linear and AS transfer modes. When the most significant bit of "len[10]" signal is logic zero, the current transfer is in the conventional linear mode, otherwise it is in the AS mode. In the linear mode, all the bits from nine to zero of "len[9:0]" signal indicate the transfer size. As an example shown in Figure 4.1(a), the linear transfer mode is indicated by the logic zero "len[10]" signal, and the transfer size is 4-word, which is indicated by the "len[9:0]" signal. Thus, the memory access is from address hexadecimal "0x00" to "0x0f" for the byte-unit memory. Figure 4.1(b) shows the timing diagram of a linear data transfer. As a word-size bus, the data shown on ASBUS are in the row-major order from memory address hexadecimal "0x00" to "0x0c".

In the AS mode, bits from nine to six of the "len[9:6]" signal represent the block width, and bits from five to zero of the "len[5:0]" signal represent the block height. As an example shown in Figure 4.2(a), the memory access is a AS transfer, which is indicated by the most significant bit of the "len[10]" signal as logic one, and the transfer size is one interleaving 1×4-word matrix or one interleaving AES state, which is indicated by the "len[9:6]" and "len[5:0]" signals. Thus, the accessing addresses are hexadecimal "0x00", "0x01", "0x02", and "0x03" for the 1kB memory and word-size ASBUS. The timing diagram of AS write and read operations are



(a) Memory Access

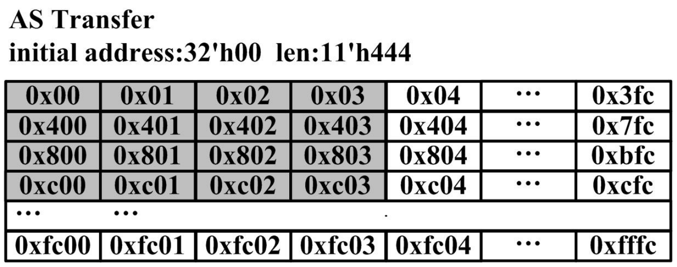


(b) Timing Diagram

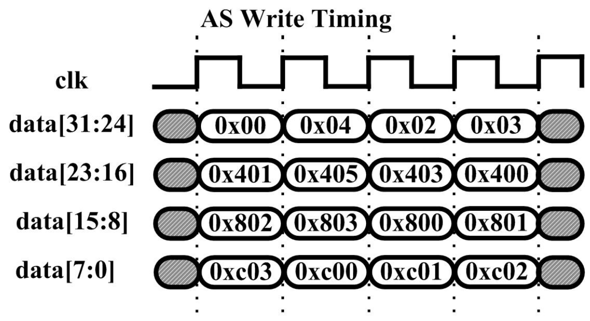
Figure 4.1: An Example of the Linear Transfer

shown in Figure 4.2(b) and Figure 4.2(c), respectively. The write operation requires AES decryption processing before the plaintext can be written into memory, while the read operation requires AES encryption processing before the ciphertext can be transferred on ASBUS. It is observed that both encryption and decryption operations are non-linear. The write data on ASBUS is in the column-major and byte de-interleaving order, while the read data on ASBUS is in the column-major and byte-interleaving order.

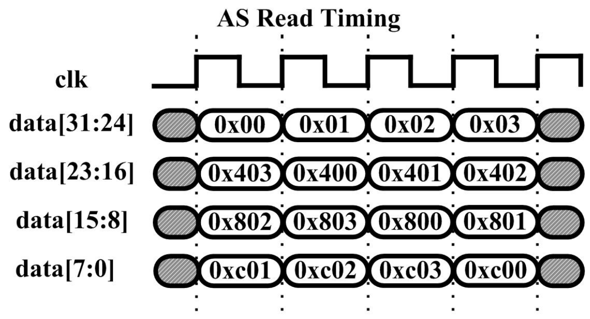
In Figure 4.2(b), notice that the first word-size write data is combined from the byte-unit data addressing from hexadecimal “0x00”, “0x403”, “0x802”, and “0xc01”. Similarly, the second, third and fourth write data on ASBUS are addressed from hexadecimal “0x01”, “0x400”, “0x803”, and “0xc02”, “0x02”, “0x401”, “0x800”, and “0xc03”, and “0x03”, “0x402”, “0x801”, and “0xc00”, respectively. In this way, the ciphertext states are reordered and row-shifted before loaded into the AES decrypter. Thus, the time consumption of the inverse cipher transformations is reduced. As the encryption process shown in Figure 4.2(c), the word-size read data should be in the column-major and byte-interleaving order before loaded into the AES encrypter, which are addressed from hexadecimal “0x00”, “0x401”, “0x802”,


AS Transfer
initial address:32'h00 len:11'h444
0x00 0x01 0x02 0x03 0x04 ··· 0x3fc
0x400 0x401 0x402 0x403 0x404 ··· 0x7fc
0x800 0x801 0x802 0x803 0x804 ··· 0xbfc
0xc00 0xc01 0xc02 0xc03 0xc04 ··· 0xcfc
··· ···
0xfc00 0xfc01 0xfc02 0xfc03 0xfc04 ··· 0xfffc

(a) Memory Access


AS Write Timing
clk
data[31:24] 0x00 0x04 0x02 0x03
data[23:16] 0x401 0x405 0x403 0x400
data[15:8] 0x802 0x803 0x800 0x801
data[7:0] 0xc03 0xc00 0xc01 0xc02

(b) Timing Diagram of Write


AS Read Timing
clk
data[31:24] 0x00 0x01 0x02 0x03
data[23:16] 0x403 0x400 0x401 0x402
data[15:8] 0x802 0x803 0x800 0x801
data[7:0] 0xc01 0xc02 0xc03 0xc00

(c) Timing Diagram of Read

Figure 4.2: An Example of the AS Transfer

and “0xc03”, “0x01”, “0x402”, “0x803”, and “0xc00”, “0x02”, “0x403”, “0x800”, and “0xc01”, and “0x03”, “0x400”, “0x801”, and “0xc02” for the first, second, third, and fourth read data, respectively. The latency of cipher transformations is reduced using this interleaving column-major transfer mode.

4.3 Static Performance Analysis

In order to evaluate the bus performance, time consumption, valid data bandwidth, and dynamic energy are defined in this section. Then ASBUS is compared with AHB and AXI4 using the three metrics to estimate the transfer performance.

4.3.1 Transfer Latency

CY denotes the total number of clock cycles of a specific data transfer. In order to focus on the bus structure, assume that response data to any bus transfer is always available immediately. For the AHB standard [1], all slave “ready” signals are asserted and “hresp” signals are OKEY during the data transactions, and all master “htrans” signals are not BUSY. In general, the SINGLE transfer type is used for the functional register configuration, and the INCR and WRAP transfer types are used for memory data access. Let HL denote each transfer size. Thus for INCR4 or WRAP4, HL is 4, and for INCR16 or WRAP16, HL is 16. Let N denote the number of data, the AHB burst transfer latency CY_H is

$$CY_H = 3 \times ceil(\frac{N}{HL}) + N. \tag{4.1}$$

In this equation, the “ceil(x)” function means that rounds a number to the next larger integer. For example, assume that every burst length HL is 4-beat, and the total burst transfer data is 7-beat. After rounded-up, these 7-beat data requires 6 cycles for command transmission, and 7 cycles for all the data transfers. Usually, request, grant and address transactions can be overlapped between two back-to-back transfers. Let P denotes the back-to-back transfer probability. The total transfer latency is modified to be

$$CY_H = [3 \times ceil(\frac{N}{HL}) + N] - 3P \times ceil(\frac{N}{HL}). \tag{4.2}$$

Then, the AXI4 data latency is considered. As a full-duplex bus, the number of AXI4 transfer cycles mainly depends on the larger number of read or write operational cycles. Let P_w and P_r, respectively, denote the data write and data read probabilities, the total transfer latency CY_X of burst transfers is

$$CY_X = max(P_r, P_w) \times \{[3 \times ceil(\frac{N}{XL}) + N] - 3P \times ceil(\frac{N}{XL})\}. \tag{4.3}$$

Both AHB and AXI4 transfer data linearly. To access data crossing block boundaries, they need to send new requests to initiate the start addresses and transfer sizes. Hence, the latency is duplicated when crossing each state-boundary. Since ASBUS supports the AS transfer mode, the transfer latency is the same for both linear and AS modes, that is

$$CY_A = max(P_r, P_w) \times \{[2 \times ceil(\frac{N}{BS}) + N] - 2P \times ceil(\frac{N}{BS})\}. \tag{4.4}$$

Here, BS represents the block size of the AS transfer. So far, we formulize the transfer latency by clock cycles. Time consumption is defined as the number of data cycles multiplied by the clock period or the reciprocal of clock frequency denoted by f hereafter, it is thus formulized as: TC=$\frac{CY}{f}$.

4.3.2 Static Performance Analysis

To evaluate the performance of the AS transfer, this section statically compares the ASBUS with AHB and AXI4 by the formulas mentioned above. As a result, Figure

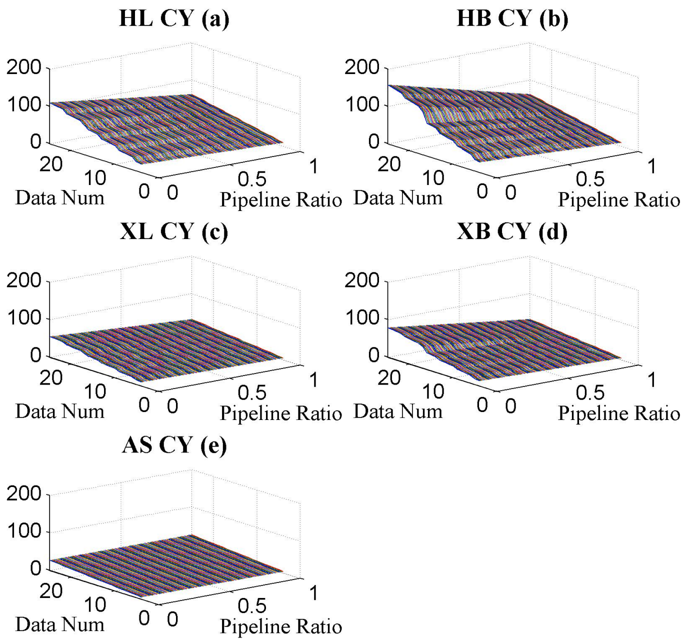


Figure 4.3: CY Comparison.

4.3 shows the linear and block CY of AHB, AXI4, and ASBUS, respectively. In each figure, one horizontal axis represents the transfer pipeline probability ranging from 0 to 1, and another horizontal axis represents the total number of data transfers ranging from 4 words to 25 words as a case study. The vertical axis indicates the CY metric. It can be observed that CY is significantly reduced as the pipeline probability increases in all the figures. In addition, even though CY increases as the data number increases, the data transfers are more efficient on a larger data number than on a smaller data number. Moreover, comparing ASBUS with AHB and AXI4, a further reduction of CY is achieved with the high-efficiency bus transfer features. For instance, using the same pipeline ratio (0.7) and data number (25 words) in all the cases, in the linear transfer mode the CY of ASBUS is reduced to about 35.3% of AHB, and 70.7% of AXI4, and in the block transfer mode the CY of ASBUS is reduced to about 29.7% of AHB, and 59.5% of AXI4.

Figure 4.4 shows the comparison of CY and valid data bandwidth among AHB, AXI4, and ASBUS. To study the impacts of the pipeline probability on system performance, the total data transfer number is fixed at 256 bytes as an example. The horizontal axis represents the pipeline probability ranging from 0 to 1, and the vertical axis indicates the CY in Figure 4.4 (a). It is obvious that ASBUS costs less cycles than AHB and AXI4, especially for the lower pipeline probabilities. As the probability increases, the CY of AXI4 becomes nearly as good as that of ASBUS, but still consumes more time than ASBUS.

As a high-efficiency bus, the ASBUS achieves the highest valid data bandwidth than AHB and AXI4, which is illustrated in Figure 4.4 (b). The horizontal axis represents the pipeline probability ranging from 0 to 1, and the vertical axis represents the valid data bandwidth unit in Gbps. First, all the valid data bandwidth of the three buses increase as the pipeline probability increases. Second, comparing the three buses, ASBUS achieves the highest valid data bandwidth at any pipeline probability. As the probability increases, the valid data bandwidth of AXI4 becomes close to that of ASBUS, but still less than that of ASBUS. As an example, when the pipeline ratio is 0.7, the valid data bandwidth of ASBUS is 3.4 and 1.7 times that of AHB and AXI4, respectively; when the pipeline ratio is 1, the valid data bandwidth of ASBUS is 2.5 and 1.25 times that of AHB and AXI4, respectively.

4.4 Performance Evaluation Methodology

In order to accurately estimate not only the front-end simulation results, such as time latency and signal switching rates, but also the back-end gate-level performance, including the hardware resource cost, the power, and energy consumption, we create

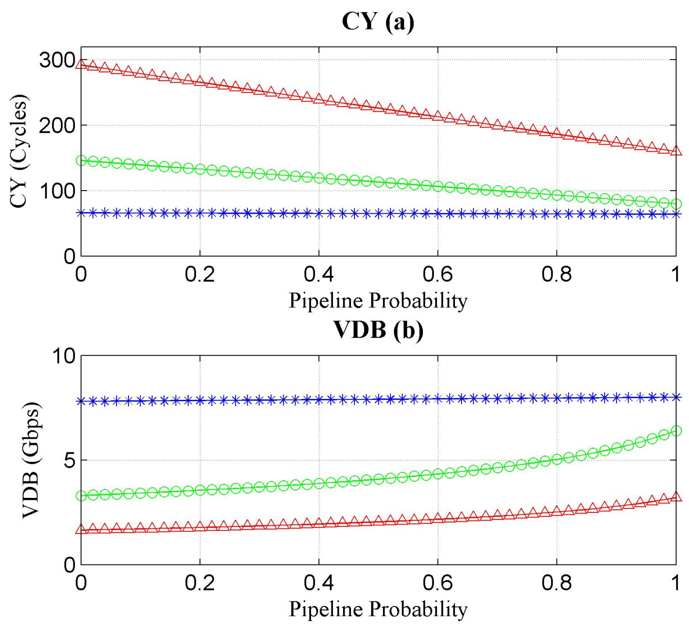


Figure 4.4: Performance Evaluation (AHB($\hat{}$), AXI4(O), ASBUS (*))

an UVM-compatible evaluation methodology based on the standard FPGA design flow.

4.4.1 Methodology Overview

The methodology is shown in Figure 4.5, each ellipse represents an implementation step; each rectangle represents an estimation model; and each rectangle with a wave-like base represents an output file. The flow involving 6 steps is illustrated below:

- Create all the application models at the algorithm stage and determine the hardware architecture considering the trade-offs among hardware costs, throughput, and power consumption.

- Design in RTL and perform simulation using direct test vectors. Moreover, the functional coverage models are created following the design specifications in this step.
- Build up a UVM environment to fully verify the DUT, which can be RTL or gate-level netlist. This step applies a time consumption & valid data bandwidth model to generate two output files for each test case, one is the simulation VCD file with the exact switching activity information, and the other is bus performance results, such as time consumption and valid data bandwidth. The time consumption & valid data bandwidth model is coded by System Verilog Language [13], thus it can be seamlessly integrated into the UVM test bench, and compatible with the traditional verification methods, such as assertion-based verification [76, 34], coverage-driven verification [82], and the constrained random test [85].
- Synthesize and obtain a fully placed and routed NCD file and a physical constraint PCF file by Xilinx ISE.
- Computes power consumption using the back-end FPGA power analysis flow. Inputting the fully placed and routed NCD file, the physical constraint PCF file, and VCD files into the XPower Analyzer tool, a PWR file with detailed power results is generated.
- Match and extract the dynamic power consumption from the PWR file and latency information from the time consumption & valid data bandwidth file by an energy model coded by Perl script. Then, compute the final dynamic energy consumption, and generate the final bus performance evaluation (BPE) report.

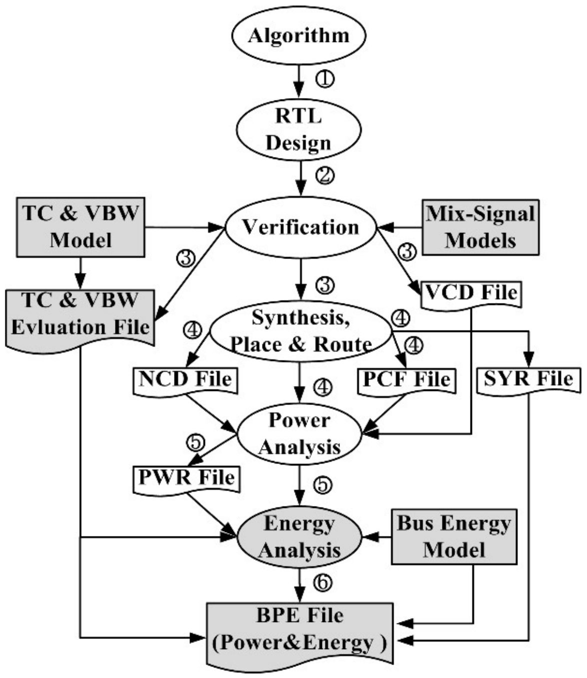


Figure 4.5: Hardware Implementation Flow

4.4.2 Verification Environment

In our work, we set up the verification environment using the UVM methodology [16, 18], employing traditional verification methods, such as assertion verification [76], [82] and the constrained random test [85]. Figure 4.6 shows the UVM-based top module that includes the test class containing the test bench, and the DUT connecting with memory. Many test classes may instantiate the test bench and verification components, and configure them as needed. The test bench is composed of reusable verification components which are applied to the DUT to verify the implementation of the protocol or design architecture. Each verification component is an encapsulated, ready-to-use, configurable verification system for an abstract container called an agent, a scoreboard, an assertion checker, a coverage model, or a performance monitor. All the agents follow a consistent architecture and consist of a complete set of elements for stimulating, checking, and collecting coverage information for a specific protocol or design.

In our test bench there are 6 bus peripheral agents, including Flash controller, I2C controller, UART, SPI controller, Watchdog, and Timer located on the control bus, and 6 application agents, such as the USB2.0 Host Controller and Mixed-Signal Wireless Fidelity (Wi-Fi), located on the data bus (ASBUS). As the UVM methodology, each of the agents, master or slave, contains three subcomponents: the sequencer, driver, and monitor. The driver is an active entity that emulates logic that drives the DUT. It repeatedly receives a data item and drives it to the DUT by sampling and driving DUT signals. The sequencer is an advanced stimulus generator that controls the items that are provided to the driver for execution. The monitor is a passive entity that samples DUT signals but does not drive them. It collects coverage information and perform checking. Moreover, the interface checker inserted between the 12 agents and DUT is combined with several assertions to check the bus protocol timing in real time. After each test complete, the DUT outputs are compared with the golden model results in the scoreboard. As an important part of the overall test effort, the regression tests are used to reduce the risk of bugs in each release. All the functional cover-points are collected by the coverage model.

Apart from the conventional verification approaches, some performance models and VIPs are created and resided into the test bench in our work. Coded by System Verilog language [13], they are reusable, configurable, and seamlessly compatible with the UVM environment. For instance, the Wi-Fi agent contains digital parts of Media Access Controller (MAC) and Baseband Processor (BBP), and some analog VIPs, such as Analog-to-Digital Converter (AD), Digital-to-Analog Converter (DA), Low Noise Amplifier (LNA), Variable Gain Amplifier (VGA), and Power amplifier (PA), as shown in Figure 4.7. Using floating-point real data to emulate these analog

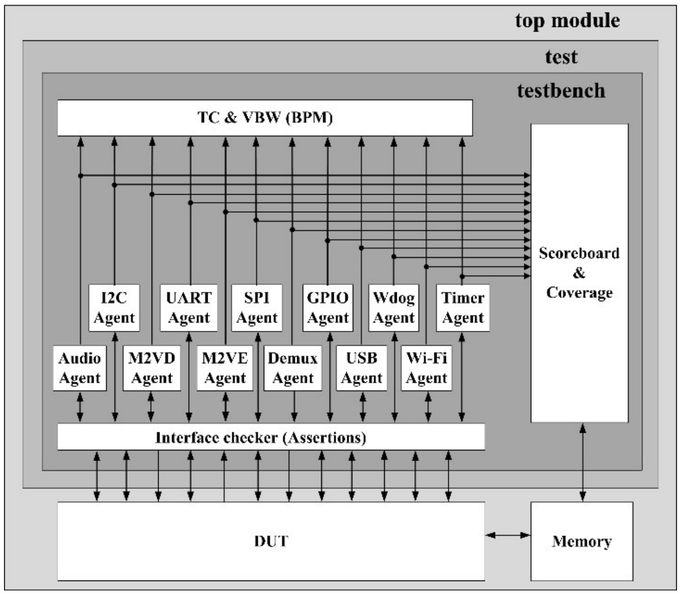


Figure 4.6: Verification Environment

models is easier for adopting a modular approach to integrate analog models into an existing pure digital environment. More specifically, LNA is at the beginning of the receiver, which minimizes the noise contribution in the following VGA. The LNA gain ranges from -3dB to 10dB or 35dB of this system, which is set through the I2C-BUS interface. VGA provides precise input attenuation and interpolation, a linear-in-dB gain-only deviating $\pm 0.5dB$. For the transmitter, PA is another output gain, which is configurable through I2C-BUS. All of these analog models have different approaches to algorithm optimization, in which abstract equation models represent the circuits. It takes analog activities to become more like digital, and reuses the digital verification environment, technologies, and methodologies [187].

Furthermore, notice that the performance model, time consumption & valid data bandwidth, used for collecting and computing time consumption and valid data bandwidth in real time is resided in the test bench. An application programming interface (API) should be configured according to bus standards before it is integrated into the environment. Using these evaluation models, the engineers are

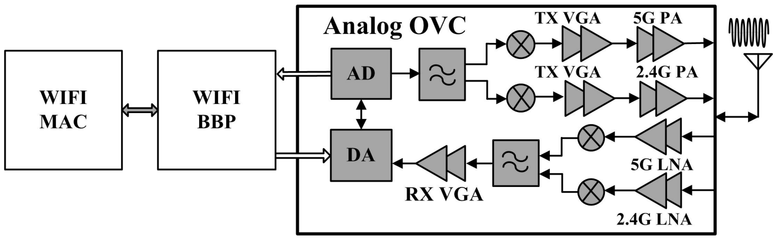


Figure 4.7: Wi-Fi Model

able to record runtime estimate of the performance of their design based on specific test cases.

4.4.3 Back-End Power and Energy Analysis

In this section, we introduce the estimation process of hardware resource costs, power and energy consumption by extending the standard back-end design flow. First, we synthesize the AHB, AXI4, and ASBUS DMA separately to get fully placed and routed files and physical constraint files, which are necessary for the methodology implementation flow. In our work, we use Xilinx ISE14.6 as the synthesis tool, and Virtex5 xc5vlx110t-2ff1136 FPGA as the target device.

The detailed resource costs are shown in Table 4.1. In the second column, it is obvious that ASBUS DMA costs less IOs than AXI4 does, under the context of full-duplex bus-based designs. As a half-duplex bus, AHB uses the least IOs but sacrifices the bandwidth by a half. In addition, ASBUS DMA consumes less slice registers and slice LUTs than AXI4 DMA does due to the high-efficiency structure, which is shown in the third and fourth columns, respectively. Comparing to AHB DMA, even though ASBUS DMA achieves much higher speed and bandwidth, the resource cost is close to that of the AHB-based design. In the fifth and sixth column, it is shown that ASBUS DMA achieves the minimum critical path and the maximum operating frequency (MOF) due to the compact structural design.

Table 4.1: Resource Cost

DUTs	IOs	Slice Registers	Slice LUTs	Critical Path (ns)	MOF (MHz)
HDMA	441	2096	5379	3.845	260.105
XDMA	647	3884	7605	4.392	227.702
ASDMA	460	2456	5761	3.630	275.509

Second, by importing the NCD and PCF files, as well as the specific VCD file with all the transition information into the Xilinx XPower Analyzer, the detailed average power report is obtained and shown in Table 4.2. Basically, the total power consumption shown in the seventh column contains static power and dynamic power, which are shown in the second and sixth columns, respectively. As expected, the static power consumption keeps a near constant to the same design, our work thus only focuses on the dynamic power analysis. For the FPGA-based design, the dynamic power consumption is composed of clock, logic & signals, and IO power. As shown in the third column, the AXI4 DMA consumes the highest clock power due to much more slice register and slice LUT utilization for both the linear and block tests. In theory, the logic & signal power consumption depends on the logic activity and the signal toggle rate. Therefore, the block tests with frequent command stages consume more logic & signal power than the linear tests do, which is shown in the fourth column. Among the three DMAs, ASBUS DMA consumes less logic & signal power than AXI4 does, due to the compact structure and specific state-based design. As a half-duplex low speed bus, AHB achieves the least logic & signals power consumption with very low switching activities. As shown in the fifth column, the IO power consumption shows that it drastically increases as the number of IO increases from AHB to AXI4 and AS transfers, and also increases as the toggle rate increases from linear to block tests. Finally, the dynamic power, which is the sum of clock, logic & signals, and IOs power consumption is shown in the sixth column. It can be observed that ASBUS consumes less dynamic power than AXI4 does for the

Table 4.2: Power Report

Tests	Power Consumption (mW)					
	SP	Clock	Logic & Signals	IOs	DP	TP
AHB L	1190.53	142.64	11.48	126.69	288.81	1471.34
AXI4 L	1195.12	192.12	39.38	227.41	458.90	1654.02
AS L	1193.63	158.47	22.90	200.16	381.54	1575.16
AHB B	1191.08	142.64	11.91	154.74	309.29	1500.37
AXI4 B	1195.90	192.55	55.57	250.88	498.99	1694.89
AS B	1194.81	158.47	23.29	223.70	405.46	1599.55

full-duplex buses. Comparing to the half-duplex AHB bus, ASBUS sacrifices more dynamic power to achieve the bandwidth and transfer speed.

For energy limited devices, the energy consumption is one of the key performance metrics. Although the power consumption of ASBUS is more than that of AHB, a much less energy usage of ASBUS is possible because of the high-speed transfer feature. In our work, the dynamic energy consumption model coded by Perl script is executed to automatically extract the dynamic power information from the PWR file, and the time consumption information from the time consumption & valid data bandwidth evaluation file individually. Then, the dynamic energy can be calculated and written into the BPE file at the end.

4.5 A Case Study: Applying The Methodology To Evaluate AHB, AXI4 and ASBUS performance

In this section, AHB, AXI4, and ASBUS DMA are implemented as a case study. Based on the UVM environment and evaluation methodology, two typical test cases are used to obtain the final results.

4.5.1 ASBUS DMA Structure

Figure 4.8 illustrates the design structure of ASBUS DMA combined with DDR2 Memory. In our work, the DDR2 Memory, including DDR2 Controller and DDR2 PHY, is a VIP instanced in the top module layer of the verification environment shown in Figure 4.6. Generally, the main functions of ASBUS DMA are arbitrating requests from ASBUS masters, scheduling responses, converting commands and data between the on-chip bus side and DDR2 Controller side. Notice that two individual command queues are used in the command scheduler module (CMD Scheduler), write queue and read queue, and the depth is four of each. Thus, up to four commands of each queue can be preprocessed ahead of data transfers. According to the configured priority and the arbitration scheme [169], the commands of higher priority are pushed into the command queues, then all of them can be popped and driven on ASBUS orderly. Using this channel-independent structure, the write and read operations can be full-duplex. Moreover, the main contribution of ASBUS DMA is the high-performance AS transfers. It provides a novel AES state transfer mode to improve the AES cipher/inverse cipher process.

4.5.2 Mixed-Signal Verification System

Leveraging pre-existing digital verification technologies and verification environment on mixed-signal SoCs is another research motivation. It will effectively mitigate the cost of mixed-signal SoC verification procedure. The optimized verification metrics and methodologies, including coverage driven verification, constraint random test, assertion based verification, and UVM/VMM verification methodologies, are briefly introduced as follows.

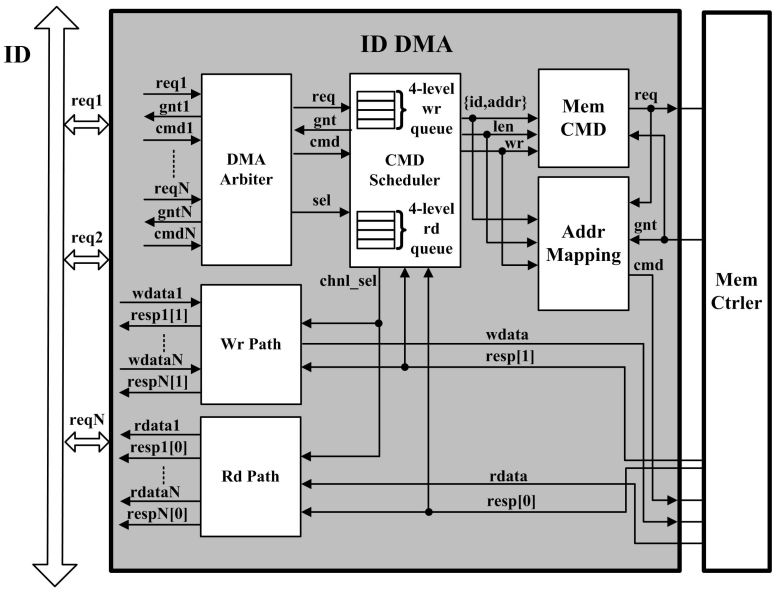


Figure 4.8: ID DMA Structure

UVM/VMM Verification Methodologies: We set up the verification environment using the object-oriented programming (OOP) concept and multi-level hierarchical scheduling by System Verilog language, with reference to the UVM verification methodology.For the signal-level layer, we add the analog models described in Figure 4.7 to digital RTL modules, which is DUT in this test bench. All the analog models are dynamically controlled from digital side, so this test bench keeps the running speed as a pure digital simulation environment. This layer interacts with the mixed-signal modules by manipulating signals through interfaces with integer or real data type.

Constraint Random Verification: Constrained random testing is more effective than directed testing approach. This co-simulation test bench uses constraint-driven test on top of an object-oriented data abstraction that models the data both on integral and real types to be randomized as objects that contain random variables and user-defined constraints. It is also an easier approach to find hard-to-reach corner cases in analog modules.

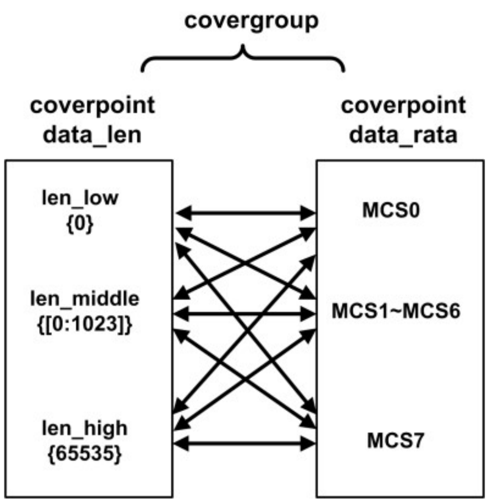


Figure 4.9: Cover Group

Coverage Driven Verification: For system level verification, coverage is used as a metric for evaluating the progress of a verification project, in order to control the number of regression times. Particularly, code coverage is used to tie the verification environment to the design intended or functionality. Functional coverage is a user-defined metric concerning test plans. It is used to describe corner cases and functional points. This test bench includes both code coverage and functional coverage.

Cover point expressions can be described by real or integral data with System Verilog. The example shown in Figure 4.9 illustrates two descriptions of integral valued cover points. The first is transmission data length of 802.11n data frame, which ranges from 0 Bytes to 65535 Bytes; the second is transmission rate, which ranges from MCS0 to MCS7 for 802.11n specification. A covergroup can contain one or more coverage points. The cover point expression takes places when the covergroup is sampled.

In the preceding example, the cross coverage group specifies cross coverage between two coverage points, transmission data length and data rate. Each coverage point includes a set of bins associated with sampled values or value transitions.

Assertion Based Verification: To validate the behavior and timing of a design, this test bench inserts some assertions that state the verification function to be performed.

4.5.3 Test Cases and Simulation

In order to focus on the bus efficiency estimation, the clock frequency is set to 100 MHz and the bus size is set to word unit for all the AHB, AXI, and ASBUS. In our empirical study, we highlight two typical test cases to investigate the bus speed and power characteristics. The first case is a linear test, which is configured to write and read 512-pixel data of a 1024 × 1024-pixel picture between external memory and the on-chip DDR2 VIP. The initial addresses are hexadecimal "0x00" and "0x83f0" for the USB Host agent and the Wi-Fi agent, respectively. The other test is an interleaving block test, which is illustrated in Figure 4.10 that 32 AES state or 512-pixel data from the top left and the bottom right of the same 1024 × 1024-pixel picture are moved between external memory and the internal DDR2 VIP.

Figure 4.11 and Figure 4.12, respectively, show the linear and block tests' simulations of AHB, AXI4, and ASBUS . In our work, we use Mentor Graphic ModelSim as the simulator [53]. For the AHB block test shown in Figure 4.12(a), around 9 ms is required to transfer the 32-state data, because additional software configuration and bus commands are necessary for each non-linear boundary operation. Due to the full-duplex feature, AXI4 DMA reduces the time consumption to be a half of the AHB transfer, which is close to 4.5 ms shown in Figure 4.12(b). Both AHB

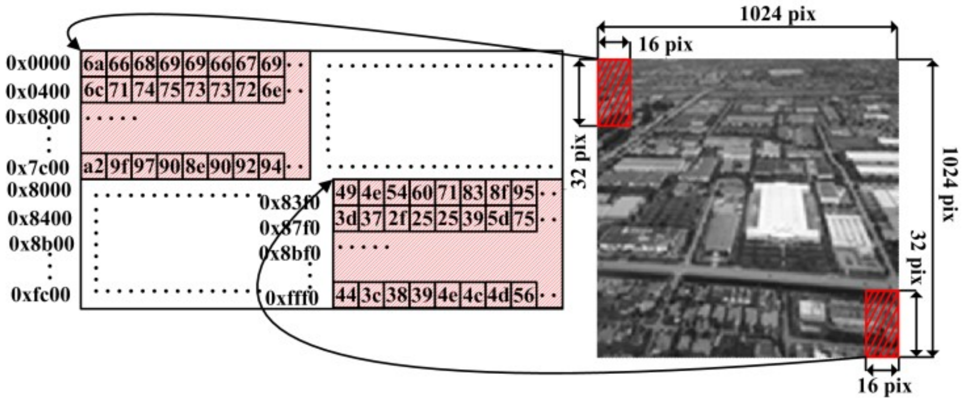
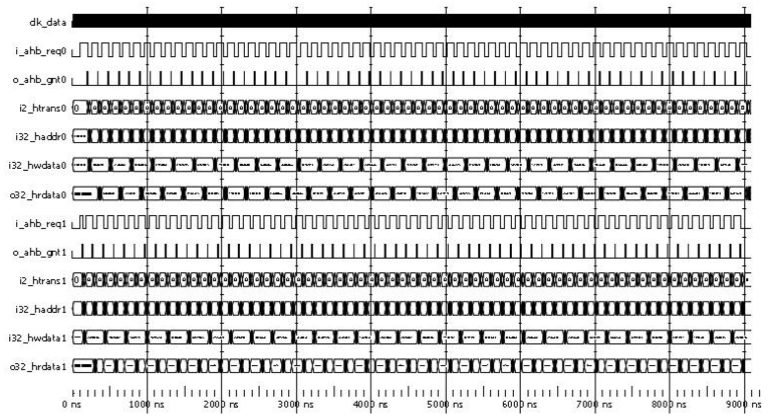


Figure 4.10: Interleaving Block Test Case

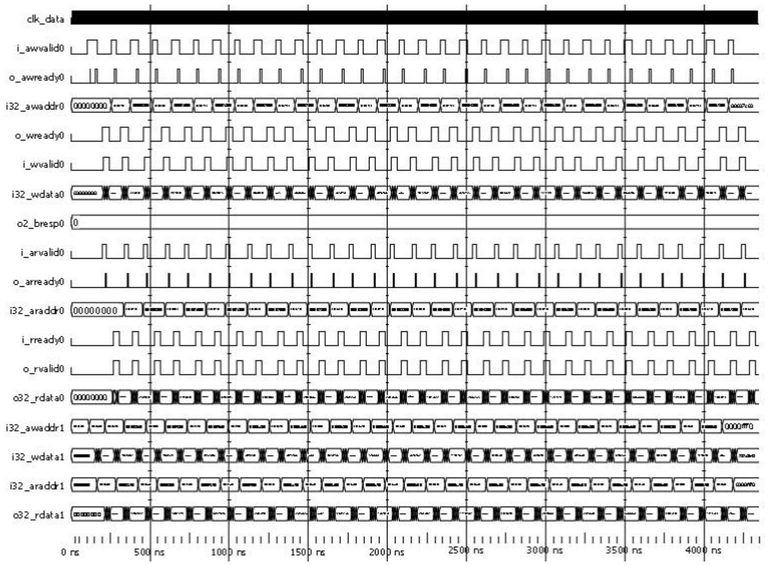
and AXI4 processing are in the row-major linear order, thus additional boundary-crossing operations are required for the block tests, which involve much more time consumption and power usage.

The ASBUS block test is shown in Figure 4.12(c). First, the ASBUS signal "i11_dma_len" is driven to be hexadecimal 11'h520. Thus, it is indicated as a AS transfer because the most significant bit of "len[10]" is logic one, and the block size is 32-state or 4×32-word as the signal bits from nine to six is hexadecimal 4'h4 and the bits from five to zero is hexadecimal 6'h20. Second, due to the specific interleaving state transfer feature, the time consumption of the AS test is around 2.6 ms, which is only 29% and 58% of the latency of AHB and AXI4 tests, respectively. Third, it is obvious that the signal toggle rate of ASBUS is less than that of AHB and AXI4, as shown in Figure 4.12(c), Figure 4.12(a) and Figure 4.12(b), respectively. At the end of simulation, all the signal toggle information is saved to the VCD file.

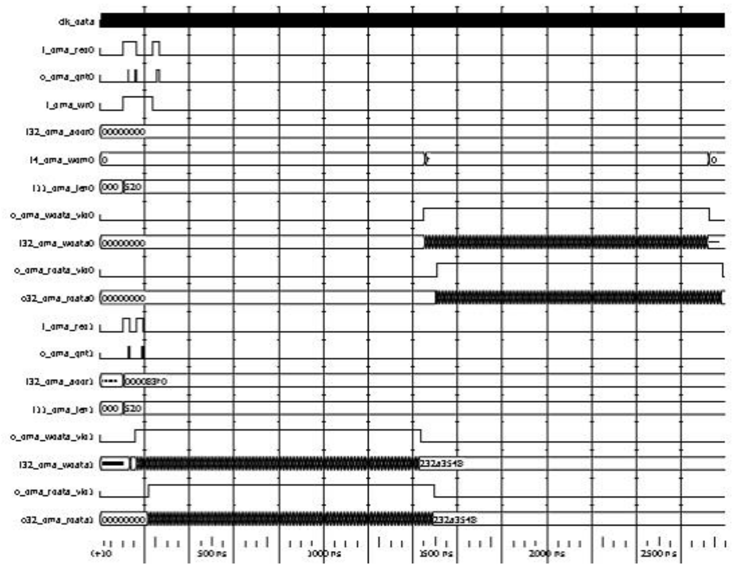
In addition, the wireless data intended to produce a regular analog waveform is shown in 4.13. Figure 4.13(a) illustrates the transmission frame including initial legacy short and long training field (STF and LTF), and HT signal field (SIG) by real data. Figure 4.13(b) shows the reception waveform, which is adjusted after LNA and VGA gain, thus the magnitude decreases after a few seconds.



(a) AHB DMA

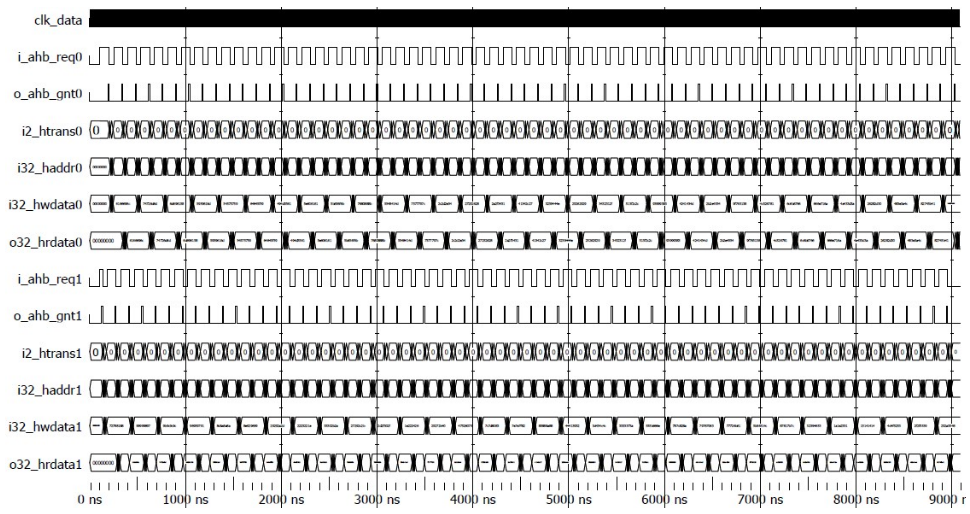


(b) AXI4 DMA

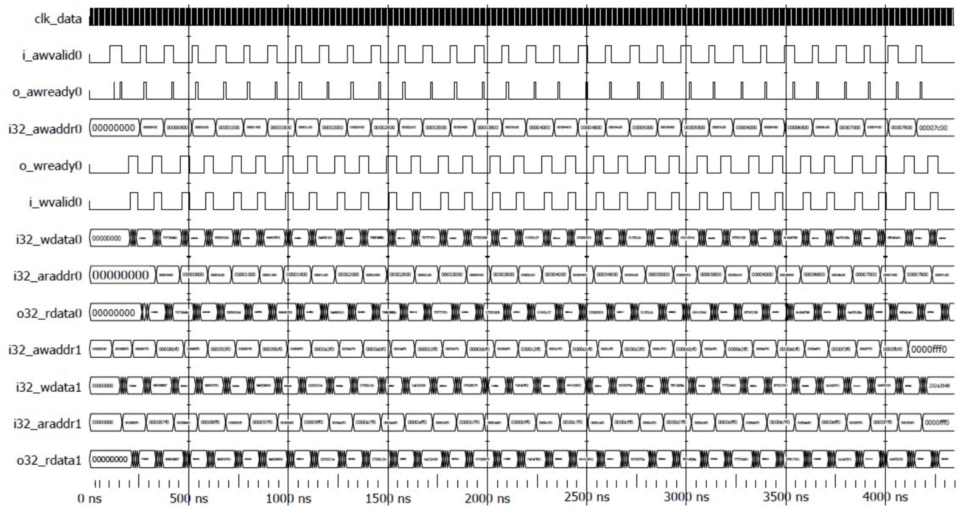


(c) ID DMA

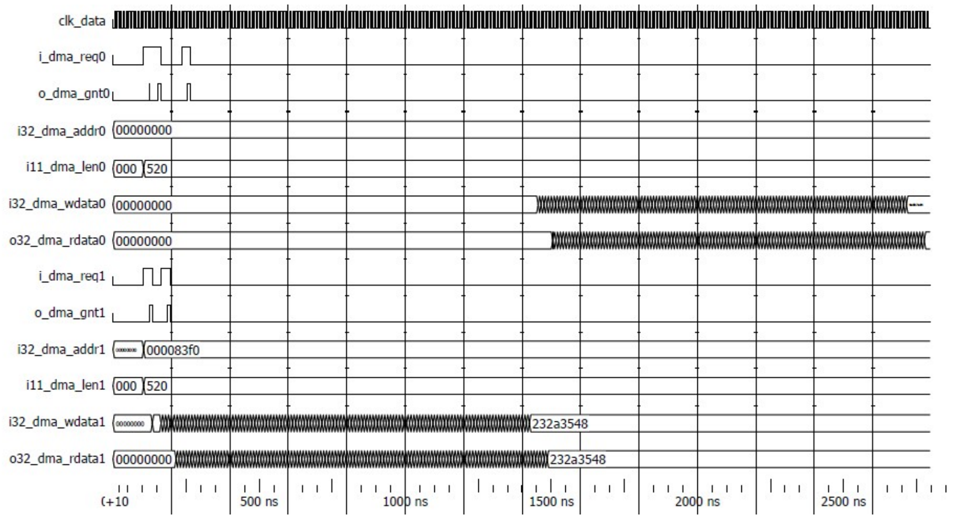
Figure 4.11: Timing Diagrams of the Linear Tests



(a) AHB DMA

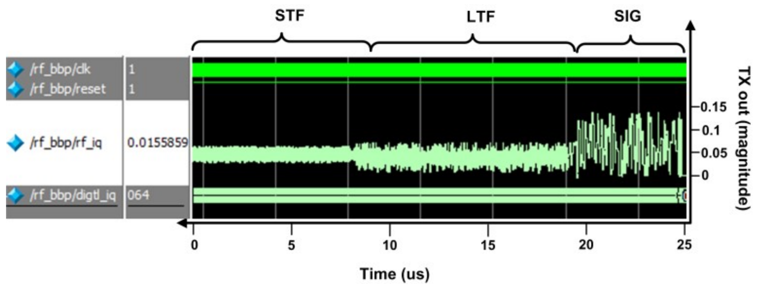


(b) AXI4 DMA

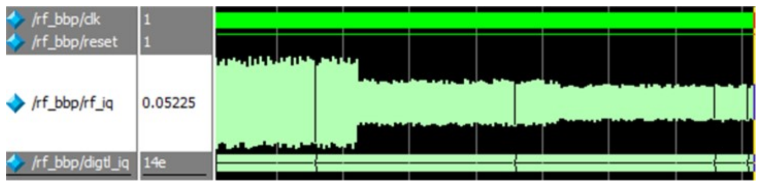


(c) ID DMA

Figure 4.12: Timing Diagrams of the Block Tests

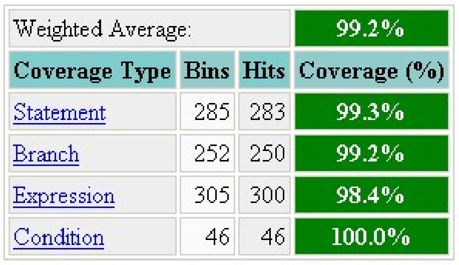


(a) Transmission



(b) Reception

Figure 4.13: 802.11a RF Waveform



Weighted Average:			99.2%
Coverage Type	**Bins**	**Hits**	**Coverage (%)**
Statement	285	283	99.3%
Branch	252	250	99.2%
Expression	305	300	98.4%
Condition	46	46	100.0%

Figure 4.14: I2C-BUS Controller Code Coverage

Coverage analysis provides metrics to evaluate verification rate of progress. It is also possible, by means of code coverage analysis, to find missing test cases and possible bugs. Specifically, 100% functional coverage must be attained because the test bench writer usually sets a one-to-one relationship between the cover point and functional point of DUT. However, it is normally impossible to reach 100% for code coverage because of missing test cases or coding style that makes coverage cases logically uncoverable. As an example, Figure 4.14 shows a coverage report of one of the modules of SoC, the I2C bus controller.

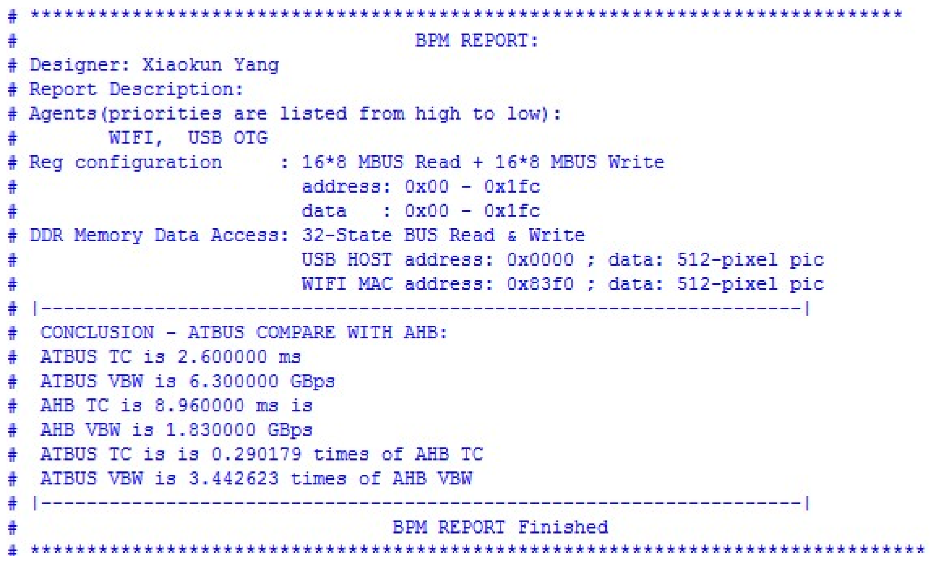
Table 4.3: Experimental Results

Test Cases	TC (us)	VDB (GBps)	DE (uJ)
HL	6.08	2.69	1.76
XL	3.04	5.39	1.40
ASL	2.60	6.30	0.99
HB	8.96	1.83	2.77
XB	4.48	3.66	2.24
ASB	2.60	6.30	1.05

4.5.4 Experimental Results

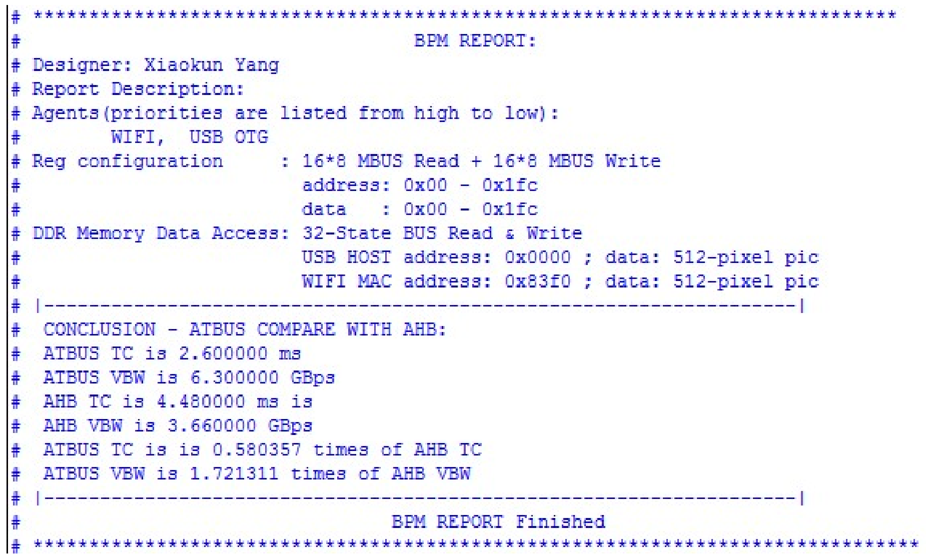
Following the evaluation methodology, the time consumption and valid data bandwidth metrics can be obtained using bus performance model based test bench during the front-end design flow. As an example, Figure 4.15(a) and Figure 4.15(b), respectively, show the performance reports of ASBUS and AHB, and ASBUS and AXI. In conclusion, we summarize all the final performance metrics including time consumption, valid data bandwidth, and dynamic energy in Table 4.3.

The difference among these tests is made clear through the analysis of time consumption, valid data bandwidth, dynamic power and dynamic energy features shown in Figure 4.16. To be a high speed bus, ASBUS time consumption is less than AHB and AXI4 time consumption as illustrated in Figure 4.16(a): ASBUS consumes 43% and 86% time that of AHB and AXI4 in the linear tests, and 29% and 58% time that of AHB and AXI4 in the block tests. Although the traditional bandwidth of ASBUS is the same as AXI4, the valid data transferring per cycle of ASBUS is far more than AXI4, which is an evidence of the high efficiency of ASBUS. As shown in Figure 4.16(b), in the block tests, the valid data bandwidth of ASBUS is 3.4 and 1.7 times that of AHB and AXI4, respectively. Figure 4.16(c) shows that the dynamic power of ASBUS is less than that of AXI4 but more than that of AHB. However, the dynamic energy of ASBUS is less than that of both AHB and AXI4



```
# ******************************************************************************
#                                   BPM REPORT:
# Designer: Xiaokun Yang
# Report Description:
# Agents(priorities are listed from high to low):
#       WIFI,  USB OTG
# Reg configuration     : 16*8 MBUS Read + 16*8 MBUS Write
#                         address: 0x00 - 0x1fc
#                         data   : 0x00 - 0x1fc
# DDR Memory Data Access: 32-State BUS Read & Write
#                         USB HOST address: 0x0000 ; data: 512-pixel pic
#                         WIFI MAC address: 0x83f0 ; data: 512-pixel pic
# |--------------------------------------------------------------------|
#  CONCLUSION - ATBUS COMPARE WITH AHB:
#  ATBUS TC is 2.600000 ms
#  ATBUS VBW is 6.300000 GBps
#  AHB TC is 8.960000 ms is
#  AHB VBW is 1.830000 GBps
#  ATBUS TC is is 0.290179 times of AHB TC
#  ATBUS VBW is 3.442623 times of AHB VBW
# |--------------------------------------------------------------------|
#                                 BPM REPORT Finished
# ********************************************************************************
```

(a) AS Bus and AHB Comparison



```
# ******************************************************************************
#                                   BPM REPORT:
# Designer: Xiaokun Yang
# Report Description:
# Agents(priorities are listed from high to low):
#       WIFI,  USB OTG
# Reg configuration     : 16*8 MBUS Read + 16*8 MBUS Write
#                         address: 0x00 - 0x1fc
#                         data   : 0x00 - 0x1fc
# DDR Memory Data Access: 32-State BUS Read & Write
#                         USB HOST address: 0x0000 ; data: 512-pixel pic
#                         WIFI MAC address: 0x83f0 ; data: 512-pixel pic
# |--------------------------------------------------------------------|
#  CONCLUSION - ATBUS COMPARE WITH AHB:
#  ATBUS TC is 2.600000 ms
#  ATBUS VBW is 6.300000 GBps
#  AHB TC is 4.480000 ms is
#  AHB VBW is 3.660000 GBps
#  ATBUS TC is is 0.580357 times of AHB TC
#  ATBUS VBW is 1.721311 times of AHB VBW
# |--------------------------------------------------------------------|
#                                 BPM REPORT Finished
# ********************************************************************************
```

(b) AS Bus and AXI4 Comparison

Figure 4.15: Bus Performance Evaluation Reports

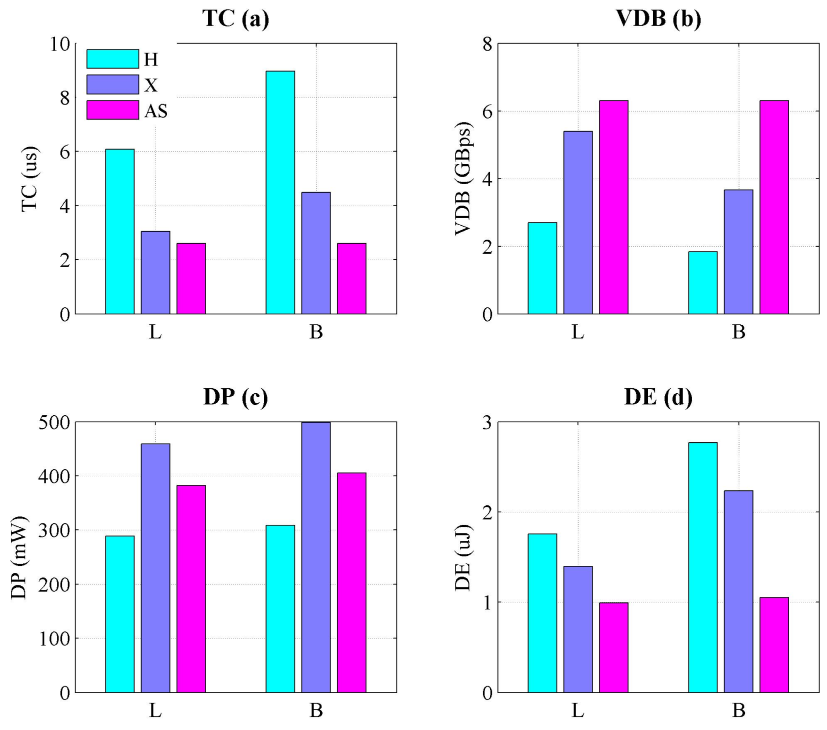


Figure 4.16: Performance Comparison

in the linear tests as shown in Figure 4.16(d). Furthermore, ASBUS reduces the dynamic energy to less than a half of that of AHB and AXI4 in the block tests.

Another contribution of this chapter is the evaluation methodology derived from the standard circuit design flow. Inserting performance estimation models in the UVM environment and the back-end FPGA design flow, the time consumption, valid data bandwidth, and dynamic energy information can be computed automatically. In addition, before the hardware implementation, the static models can be used to predict time consumption and valid data bandwidth. dynamic energy can be calculated using the static time consumption and the back-end result dynamic power. In order to verify the validation of the static models, we compare all the

Table 4.4: Modeling Results

Test Cases	TC (us)	VDB (GBps)	DE (uJ)
HL	6.11	2.68	1.77
XL	2.85	5.76	1.31
ASL	2.70	6.08	1.03
HB	8.99	1.82	2.78
XB	4.25	3.86	2.12
ASB	2.70	6.08	1.09

metrics resulted from the implementation flow and analytical models. The static modeling results are summarized in Table 4.4.

Furthermore, the difference as an error percentage of the practical value is shown in Figure 4.17. The worst case of the comparison is 6.9% at the AXI4 valid data bandwidth modeling shown in Figure 4.17(b). The reason is that as a full-duplex bus model, the AXI4 valid data bandwidth is based on the assumption of fully overlapping write and read operations, but in practice, each read operation is behind write to monitor the correctness of the memory accesses. Therefore, the experimental valid data bandwidth is less than modeling results. Furthermore, the average estimation error is around 3.3%. Through the experiments, we verify that all the proposed models can be effectively applied for evaluating the bus performance.

4.6 Summary

First of all, this chapter proposes a specific interleaving AES state transfer mode, AS transfer. In order to help architects to estimate the transfer performance accurately and automatically, a performance methodology is also proposed. Then, AHB-, AXI4-, and ASBUS-based designs are implemented as a case study. Comparing with the traditional AHB and AXI4, ASBUS achieves the highest speed and bandwidth, and the energy consumption of ASBUS-based design is less than a half

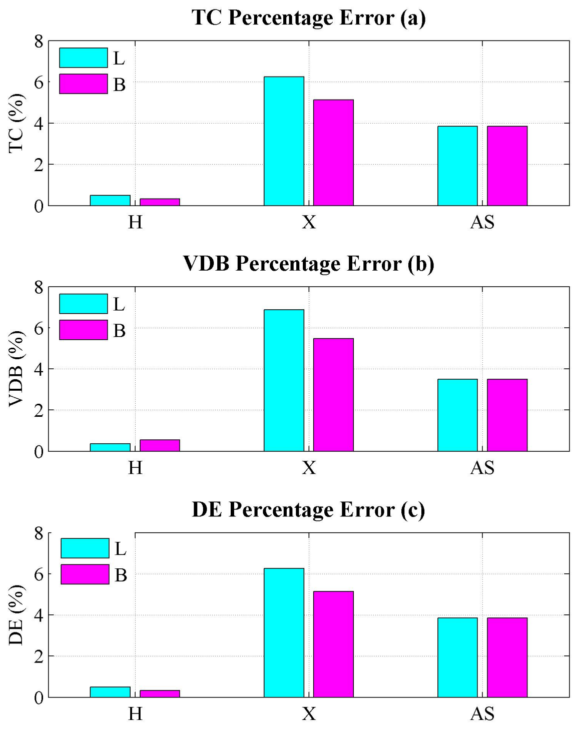


Figure 4.17: Percentage Error between Experimental Results and Modeling Results

of AHB and AXI4. Moreover, qualitative and quantitative statistical comparisons between the hardware implementation and modeling results show that very high accuracy is achieved by the proposed models and the evaluation methodology.

In addition, this chapter has presented a verification process of a HDTV mixed-signal SoC at the system level. Much has been done in the digital design space, digital simulator and verification methods, to address this verification complexity and minimize time consumption during simulation. Key advantages for this implementation include: (i) real data in a digital-metric simulation tool and test bench, (ii) verification methodologies applicable to mixed-signal SoC, (iii) reusable real signal models or VIPs, (iv) a faster approach that can speed up the regression testing period, (v) constrained random input data, (vi) assertion-based verification, (vii) code coverage and functional coverage verification. The results show that it is one of the best choices for complex mixed-signal SoC level verification, because not only does it have faster simulation speed but also is easier to migrate to the advanced digital verification technologies. The drawbacks of such models tend to be less accurate and it is very difficult to write equivalent models for many classes of circuits. Applications of this mixed-signal verification solution are better used for functional simulation.

In brief, this chapter links analog and digital circuits in a common verification environment to help the verification engineer to apply more verification methods then in traditional approaches. The future research will focus on improving the accuracy of analog models, thereby increasing the flexibility and reusability of the verification.

CHAPTER 5

AN ADVANCED AES-ENCRYPTED IBUS ARCHITECTURE

Security is becoming a de-facto requirement of system-on-chips (SoC), leading up to a significant share of circuit design cost [74, 63, 140]. To improve chip performance and the capabilities to provide efficient architectural support for the complex AES algorithm, an advanced IBUS architecture for AES-encrypted circuits is proposed in this chapter. Then, different FPGA implementations, 32-, 64-, and 128-bit IBUS and AXI DMAs, combined with full pipeline and maximum overlapping AES core and memory controller (IDAM and XDAM), are optimized and evaluated to identify the high-speed and low-power architectures for the low-cost and low-power chips. The results show that the presented IBUS structure outperforms the AXI design. As an example, the 32-bit IDAM costs less in terms of hardware resources and achieves higher throughput (1.30$\times$) than the 32-bit XDAM, and the dynamic power consumed by the IBUS cipher test is reduced to 71.27% compared with the AXI cipher test.

5.1 Related Work

The rapid rise in Internet-connected devices imposes increasingly higher requirements on high-performance and high-security SoCs, in terms of low-cost, high-speed, energy-efficiency, and data security. It creates new problems and challenges between the complexity of security algorithms and the limited resource and power of embedded chips. In order to protect data communication in wireless networks, several cryptology algorithms have been widely used in hardware today. The AES, issued by the US NIST in 2011, is the dominant symmetric-key cryptosystem [19]. For decades, numerous hardware implementations were proposed and their performance were evaluated using ASIC [72] and FPGA [175, 132]. However, all the previous research attempts to optimize AES-encrypted chips frequently fall back on refining the AES cores rather than on AES system as a whole; indeed, refining part of the system is useful, yet the focus, such as transfer efficiency and energy consumption, is still on bus architectures.

Mathematically, AES operates on a 4×4 column-major order matrix of bytes, termed the state. Each state is performed by 10, 12, or 14 rounds of transformations with key lengths equal to 128, 192, or 256 bits, respectively. In each round, except for the final round, four transformations, including SubBytes (SB), ShiftRows (SR), MixColumns (MC), and AddRoundKey (AR) are performed for encryption, while InvSubBytes (ISB), InvShiftRows (ISR), InvMixColumns (IMC), and AR are performed for decryption.

Among the transformations in AES encryption/decryption, the SB/ISB transformation is a non-linear operation requiring the highest area and consuming much power of the circuit. Some of the earlier SB/ISB implementations are based on look-up table, such as those described in [61, 124, 99]. The unbreakable LUT accessing limits the high-efficiency applications, such as parallel computation and pipeline operations. Thus, an alternative composite field method for the S-Box computation [144] is suggested by V. Rijmen, who is one of the AES inventors. Based on this finite field arithmetic, many high-performance implementations are proposed to replace the LUT-based S-Box transformations by combinational logics [193, 47, 127, 91, 27].

Moreover, [192] and [117] analyze and compare the complexity of the SB implementation using different irreducible polynomials. The AES performance is also considered on the core structural level in [41], [146, 30, 159, 77, 163]. For instance, the four primitive transformations are decomposed, rearranged, and regrouped as new linear and non-linear operations in [41] to provide 1.28 Gbps (0.16 GBps) throughput for 128-bit keys. In [146], the transformations A/IA, SR/ISR, and MC/IMC are combined into a single function unit A/SR/MC or IMC/ISR/IA, and the sub-structure sharing algorithm is applied to reduce the area cost.

However, all the previous AES research was based on the assumption that all the AES states can be input to the encrypter (ENC)/decrypter (DEC) column-by-column immediately. Actually, it is unfeasible to transfer data without any bus protocol overhead, particularly to process data by shifted/inverse-shifted block in the column-major order. Traditional bus architectures, such as the AMBA AHB [1] and AXI [5] from ARM Holdings, Wishbone from Silicore Corporation [6], OCP from OCP-IP [4], CoreConnect from IBM [2], and STBus from STMicroelectronics [7], process data in the row-major order and are very low-efficiency to supply the rectangular array of bytes. The IBUS proposed in [186] can transfer data by block. However, the operating order of IBUS is in the row-major order as well. From the system point of view, the bus architecture plays a pivotal role in advancing the AES-encrypted circuit performance: the resource costs are influenced by the architectural degree of parallelism and the number of pipeline stages, the speed is determined by the maximum operating frequency (MOF) of the whole system and the bus efficiency, and the energy consumption is dependent on the gate count and logic, signal & IO toggle rates (LSIO).

To tackle these issues, we further propose an advanced IBUS architecture for the AES-encrypted embedded systems to enhance the SoC performance with the overhead cost of security applications. To the best of our knowledge, this is the first performance analysis for AES circuits on the bus architectural point of view. Furthermore, different solutions of the state transfers are presented, which are used to consider the tradeoffs among area, throughput, and power consumption together in the architectural design.

The organization of this chapter is as follows: section 5.2 briefly introduces the AES system structure, section 5.3 presents the IBUS architecture and analyzes its performance statically. In section 5.4, the RTL design, simulation, synthesis, and power analysis are illustrated. The experimental results are shown in section 5.5. Finally, section 5.6 concludes this chapter.

5.2 AES-Encrypted Circuit

In this section, we briefly illustrate the AES system structure, and then explain the detailed logic implementations of the ENC and DEC engines based on the composite field arithmetic.

5.2.1 Mathematical Preliminaries

The AES standard specifies the Rijndael algorithm, a symmetric block cipher that can process 128-bit states, using cipher keys with lengths of 128, 192, and 256 bits. The key length is represented by N_b=4, 6, or 8, which denotes the number of 32-bit data in the cipher key. For the AES algorithm, the number of rounds to be performed depends on the key size. It is represented by N_r, where N_r=10 when N_b=4, N_r=12 when N_b=6, and N_r=14 when N_b=8. In our study, only the 10-round AES algorithm with 4-word key is implemented as a case study.

For both cipher and inverse-cipher processes, each AES round, except for the final round, consists of four different byte-oriented 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. Among the four transformations, SB/ISB is the bottleneck of the speed and power consumption of the AES core. The most common strategy to implement the S-box is employing the LUT-based design. It results in very high area overhead and a non-parallel structure due to the fixed operational delay of LUTs. Therefore, the composite field arithmetic over GF(2^8), which employs combinational logic only, is used to exploit the advantage of internal pipeline in our work. In theory, the composite field of GF(2^8) can be built iteratively from GF(2) using the irreducible polynomials [138]:

$$GF(2) \rightarrow GF(2^2) : x^2 + x + 1 \tag{5.1}$$

$$GF(2^2) \rightarrow GF((2^2)^2) : x^2 + x + \phi \tag{5.2}$$

$$GF((2^2)^2) \rightarrow GF(((2^2)^2)^2) : x^2 + x + \lambda \tag{5.3}$$

First, x^2+x+1 is the only irreducible polynomial of degree 2 over GF(2). Second, there are two values of ϕ that make x^2+x+ϕ irreducible over GF(2^2), and 8 possible values of λ that make x^2+x+λ irreducible over GF($(2^2)^2$) constructed by using each of ϕ. All together, there are sixteen ways to construct the composite field GF($((2^2)^2)^2$)) using irreducible polynomials in the equations. As a case study, we use ϕ=$\{10\}_2$, λ=$\{1100\}_2$ in our work.

5.2.2 AES Circuit Structure

Figure 5.1 shows a 32-bit AES system structure including ENC and DEC engines. For the non-cipher mode, the AES ENC engine is bypassed on the read data path, and the AES DEC engine is bypassed on the write data path. Otherwise, the write data are decrypted before being stored into the memory, and the read data are encrypted before being transferred on the data bus. Both ENC and DEC engines include 2 sub-stages (SS), SS1 and SS2, of the 10-round function. The SB/ISB

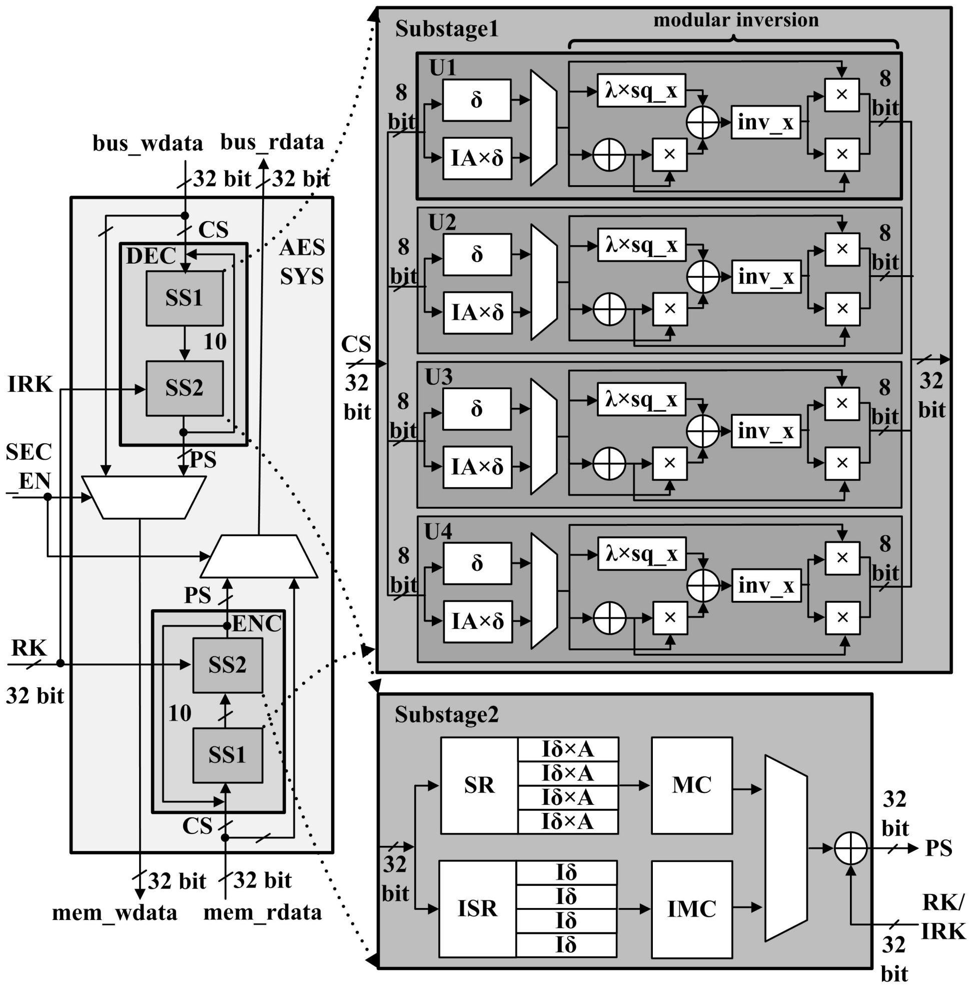


Figure 5.1: AES Circuit Structure.

transformation is decomposed as a modular inversion over GF(2^4) located in SS1 and four linear functions (A, IA, δ, and $I\delta$). In order to shorten the SB/ISB critical path, IA is combined with δ ($IA \times I\delta$) in SS1, and $I\delta$ is merged with A ($I\delta \times A$) in SS2. In addition, the SR/ISR, MC/IMC, and AK transformations are integrated in SS2 to obtain approximately equal delay to SS1. The key expansion unit can be instantiated as either a hardware or a software generator. Concentrating on the transfer efficiency of the system, the round keys are configured by software through control bus in our work.

In what follows, we briefly explain the gate-level implementations of all the AES operators. Assume that all the functions are black boxes with logic input and output. Let "a" denote the input and "b" denote the output in a one-in, one-out assignment hereafter. The bit-width of "a" and "b" are 8-, 4-, and 2-bit, respectively, when the operator is in GF(2^8), GF(2^4), and GF(2^2) fields. Hence, the logic designs of δ and $I\delta$ are written below:

$$\begin{aligned} b =\{&a_7 \oplus a_5, a_7 \oplus a_6 \oplus a_4 \oplus a_3 \oplus a_2 \oplus a_1, a_7 \oplus a_5 \oplus a_3 \oplus a_2,\\ &a_7 \oplus a_5 \oplus a_3 \oplus a_2 \oplus a_1, a_7 \oplus a_6 \oplus a_2 \oplus a_1, a_7 \oplus a_4 \oplus a_3 \oplus a_2 \oplus a_1,\\ &a_6 \oplus a_4 \oplus a_1, a_6 \oplus a_1 \oplus a_0\} \end{aligned} \tag{5.4}$$

$$\begin{aligned} b =\{&a_7 \oplus a_6 \oplus a_5 \oplus a_1, a_6 \oplus a_2, a_6 \oplus a_5 \oplus a_1, a_6 \oplus a_5 \oplus a_4 \oplus a_2 \oplus a_1,\\ &a_5 \oplus a_4 \oplus a_3 \oplus a_2 \oplus a_1, a_7 \oplus a_4 \oplus a_3 \oplus a_2 \oplus a_1, a_5 \oplus a_4,\\ &a_6 \oplus a_5 \oplus a_4 \oplus a_2 \oplus a_0\} \end{aligned} \tag{5.5}$$

Derived from Verilog Hardware Description Language (HDL) [20], the concatenation operator "{,}"combines the bits of 2 or more data objects. In equation 5.4 and equation 5.5, δ and $I\delta$ are implemented by "XOR" gates denoted as "$\oplus$" hereafter. Likewise, the logic designs of A and IA can be represented as

$$\begin{aligned} b =\{&a_7 \oplus a_6 \oplus a_5 \oplus a_4 \oplus a_3, \sim a_6 \oplus a_5 \oplus a_4 \oplus a_3 \oplus a_2, \sim a_5 \oplus a_4 \oplus a_3 \oplus a_2 \oplus a_1,\\ &a_4 \oplus a_3 \oplus a_2 \oplus a_1 \oplus a_0, a_7 \oplus a_3 \oplus a_2 \oplus a_1 \oplus a_0, a_7 \oplus a_6 \oplus a_2 \oplus a_1 \oplus a_0,\\ &\sim a_7 \oplus a_6 \oplus a_5 \oplus a_1 \oplus a_0, \sim a_7 \oplus a_6 \oplus a_5 \oplus a_4 \oplus a_0\} \end{aligned} \tag{5.6}$$

$$\begin{aligned} b =\{&a_6 \oplus a_4 \oplus a_1, a_5 \oplus a_3 \oplus a_0, a_7 \oplus a_4 \oplus a_2, a_6 \oplus a_3 \oplus a_1, a_5 \oplus a_2 \oplus a_0,\\ &\sim a_7 \oplus a_4 \oplus a_1, a_6 \oplus a_3 \oplus a_0, \sim a_7 \oplus a_5 \oplus a_2\} \end{aligned} \tag{5.7}$$

, respectively. In these 2 equations, the "$\sim$" operator indicates a bit-wise logic inversion of each input bit [20]. The multiplicative inversion module can be shared in a combined structure. Theoretically, any arbitrary polynomial can be represented as px+q where p is the upper half term and q is the lower half term. Denoting the irreducible polynomial as x^2+Ax+B, the multiplicative inversion for an arbitrary polynomial px+q is given by

$$(px+q)^{-1} = p(p^2B+pqA+q^2)^{-1}x + (q+pA)(p^2B+pqA+q^2)^{-1} \tag{5.8}$$

Therefore, the inversion calculation in GF(2^8) is now translated to calculate the inversion in GF(2^4) with performing some multiplications, squaring, and additions in GF(2^4). As shown in Figure 5.1, the multiplication with constant λ and squaring in GF(2^4) can be combined together to reduce the combinational logic cost and shorten the critical path, which is modified as below:

$$\begin{aligned} b_3 &= a_2 \oplus a_1 \oplus a_0 \\ b_2 &= a_3 \oplus a_0 \\ b_1 &= a_3 \\ b_0 &= a_3 \oplus a_2 \end{aligned} \tag{5.9}$$

Using the combining logic in equation 5.9, the implementation of multiplication with constant λ and squaring in GF(2^4) can be optimized as 4 “XOR” gates with 2 “XOR” gates in the critical path. It reduces one “XOR” gate delay in the critical path compared to [193].

Moreover, the multiplication in GF(2^4) field can be further decomposed into multiplication in GF(2^2), and then to GF(2). For a two-in, one-out assignment, let “a” and “b” denote 2 inputs, and “c” denote the output hereafter. The bit-width of “a”, “b”, and “c” are 4-bit and 2-bit if the operator is in GF(2^4) and GF(2^2), respectively. Assume c=$a \times b$, where a=a_Hx+a_L and b=b_Hx+b_L. Here, a_H and b_H are the upper half term, and a_L and b_L are the lower half term. Then, the product of a and b is

$$c = (b_H a_H + b_H a_L + b_L a_H)x + b_H a_H \theta + b_L a_L \tag{5.10}$$

This equation is in the form of GF(2^2). In order to decompose the GF(2^2) multiplication to GF(2), the logic for computing GF(2) multiplication is rewritten as

$$\begin{aligned} c_1 =& b_1 a_1 \oplus b_0 a_1 \oplus b_1 a_0 \\ c_0 =& b_1 a_1 \oplus b_0 a_0 \end{aligned} \tag{5.11}$$

and the logic for computing GF(2) multiplication with constant φ is

$$\begin{aligned} b_1 =& a_1 \oplus a_0 \\ b_0 =& a_1 \end{aligned} \tag{5.12}$$

Using equation 5.11 and equation 5.12, the multiplication in GF(2^4) can be implemented in hardware as multiplication involving only “XOR” and “AND” gates. In theory, the inversion in GF(2^4) can be implemented by repeating squaring and multiplication, decomposing inversion by applying formulas similar to equation 5.8 iteratively, and computing each inverse bit individually [138]. Using the direct implementation of the inverse bit, the GF(2^4) inversion is shown as below:

$$\begin{aligned} b_3^{-1} =& a_3 \oplus a_3 a_2 a_1 \oplus a_3 a_0 \oplus a_2 \\ b_2^{-1} =& a_3 a_2 a_1 \oplus a_3 a_2 a_0 \oplus a_3 a_0 \oplus a_2 \oplus a_2 a_1 \\ b_1^{-1} =& a_3 \oplus a_3 a_2 a_1 \oplus a_3 a_1 a_0 \oplus a_2 \oplus a_2 a_0 \oplus a_1 \\ b_0^{-1} =& a_3 a_2 a_1 \oplus a_3 a_2 a_0 \oplus a_3 a_1 \oplus a_3 a_1 a_0 \oplus a_3 a_0 \oplus a_2 \oplus a_2 a_1, a_2 a_1 a_0 \oplus a_1 \oplus a_0 \end{aligned} \tag{5.13}$$

So far, the SB/ISB logic implementation split into several operators in the composite field is completed. For the SR/ISR transformation, the bytes in the last three rows of the state are cyclically-shifted/cyclically-inverse-shifted over different numbers of bytes. The first row is not shifted. The second, third, and fourth rows are left shifted one, two, and three bytes for the SR transformation, and right shifted one, two, and three bytes for the ISR transformation, respectively. Since the cyclic rotation does not affect the regrouping result, the order of $I\delta \times A/I\delta$ and SR/ISR is further exchanged, as shown in Figure 5.1. In this way, the four byte-size outputs of SS1 can be reordered as the shifted/inverse-shifted rules and merged with $I\delta \times A/I\delta$ operators, then combined with the word-size input of the MC/IMC transformation in SS2. In our study, the XTime method composed of a fundamental multiplication block called XTime that multiplies a byte with constant values {02} and {04} is used. Let s denote the initial bytes of a state, the logic designs of {02}s and {04}s are

$$\begin{aligned} b =&\{a_6, a_5, a_4, a_3 \oplus a_7, a_2 \oplus a_7, a_1, a_0 \oplus a_7, a_7\} \\ b =&\{a_5, a_4, a_3 \oplus a_7, a_2 \oplus a_6 \oplus a_7, a_1 \oplus a_6, a_0 \oplus a_7, a_6 \oplus a_7, a_6\} \end{aligned} \tag{5.14}$$

, respectively. Let the prefix "s_" denote the MC output signal and "is_" denote the IMC output signal. The logic implementations of MC and IMC are rewritten as:

$$\begin{aligned} s\_s_0 =&\{02\}(s_0 \oplus s_1) \oplus s_2 \oplus s_3 \oplus s_1 \\ s\_s_1 =&\{02\}(s_1 \oplus s_2) \oplus s_3 \oplus s_0 \oplus s_2 \\ s\_s_2 =&\{02\}(s_2 \oplus s_3) \oplus s_0 \oplus s_1 \oplus s_3 \\ s\_s_3 =&\{02\}(s_3 \oplus s_0) \oplus s_1 \oplus s_2 \oplus s_0 \end{aligned} \tag{5.15}$$

$$\begin{aligned} is\_s_0 =&s\_s_0 \oplus (\{02\}(\{04\}(s_0 \oplus s_2) + \{04\}(s_1 \oplus s_3)) + \{04\}(s_0 \oplus s_2)) \\ is\_s_1 =&s\_s_1 \oplus (\{02\}(\{04\}(s_0 \oplus s_2) + \{04\}(s_1 \oplus s_3)) + \{04\}(s_1 \oplus s_3)) \\ is\_s_2 =&s\_s_2 \oplus (\{02\}(\{04\}(s_0 \oplus s_2) + \{04\}(s_1 \oplus s_3)) + \{04\}(s_0 \oplus s_2)) \\ is\_s_3 =&s\_s_3 \oplus (\{02\}(\{04\}(s_0 \oplus s_2) + \{04\}(s_1 \oplus s_3)) + \{04\}(s_1 \oplus s_3)) \end{aligned} \tag{5.16}$$

In equation 5.15 and 5.16, s_0, s_1, s_2, and s_3 represent the first, second, third, and fourth bytes in a column of a state, respectively. In the final AK transformation, a round key is added to the state by a simple bitwise XOR operation. For the ENV

Table 5.1: AES ENC/DEC Gate Count and Critical Path

Modules	Total Gates	Critical Path
δ	12XOR	4XOR
$x^2 \times \lambda$	4XOR	2XOR
Multiplication in GF(2^4)	21XOR+9AND	4XOR+1AND
x^{-1}	14XOR+9AND	3XOR+2AND
$\delta^{-1} \times A$	19XOR	4XOR
$A^{-1} \times \delta$	19XOR	3XOR
δ^{-1}	17XOR	3XOR
MC	108XOR	3XOR
IMC	193XOR	7XOR

engine, the 10-round keys from RK(0) to RK(a) are forward input, otherwise, the direction is reversed from RK(a) to RK(0) for the DEC engine.

5.2.3 AES Circuit Performance Analysis

As can be observed from the equations introduced in subsection 5.2.2, the gate costs and critical path for each operator are summarized in Table 5.1. The internal pipeline structure can achieve the maximum speed if each round unit can be divided into SSs with equal delay. Considering the whole system performance, we divide the cipher/inverse-cipher core into two SSs with approximately equal critical path latencies. For the ENC engine, the critical path of SS1 has 15 XOR gates and 1 MUX, and the critical path of SS2 has 8 XOR gates and 1 MUX. For the DEC engine, the critical path of SS1 has 16 XOR gates and 1 MUX, and the critical path of SS2 has 11 XOR gates and 1 MUX. Moreover, notice that four 8-bit interface units (U1, U2, U3, and U4) are instanced in SS1 to interconnect with 32-bit SS2. In SS2, the 8-bit operator, $I\delta \times A$, is duplicated four times to match 32-bit SR/ISR and MC/IMC transformations.

5.3 AES-Encrypted IBUS Protocol

Derived from IBUS protocol, this section presents an advanced on-chip data communication standard for designing low-power and high-speed AES-encrypted microcontrollers. In what follows, the architectural performance is statically analyzed.

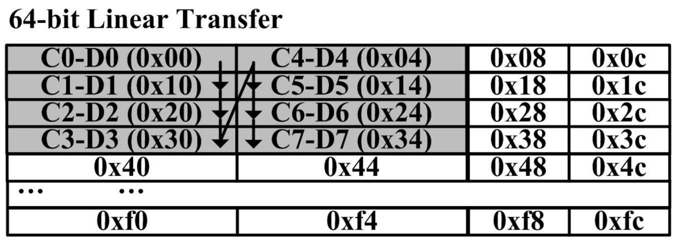
5.3.1 State Transfer Mode

To illustrate the transfer types provided by IBUS, a two-state memory access example is shown in Figure 5.2. In Figure 5.2(a), eight consecutive linear transfers are required to access two 4×4-byte matrices. Each transfer includes one command stage (C) and one data stage (D). In addition, the block transfer is supported by IBUS to improve the performance of matrix-based applications in some specific fields, such as image processing, computer vision, and wireless communication [71, 3, 15]. The block transfer defines the rectangle size and makes every memory boundary-crossing command computable by hardware, so that the time consumption of software configuration and bus commands is reduced. Since the consecutive data of the rows of the array are contiguous in memory, the block transfer is essentially a row-major order transfer as well. Figure 5.2(b) shows a memory access example of two 4×4-byte matrices using the block mode. Two block transfers are required to load or store two matrices, and each of the transfers involves one command stage and four data stages.

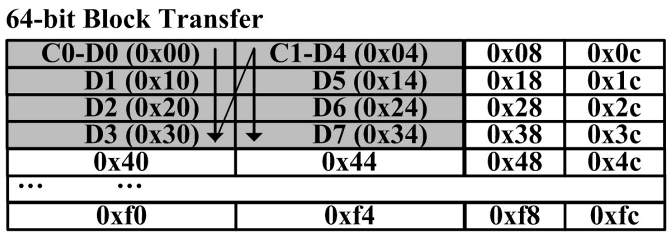
A novel transfer mode, the AES state transfer, is the main contribution to the AES-encrypted IBUS architecture in this chapter. It advantageously optimizes data supply efficiency involving encryption/decryption processing. This transfer mode may reduce the processing load of data scheduling and buffering and power consumption in system environments making use of AES cryptographic processing. In implementations with the AES state transfer mode, the "AES state" is adopted as the basic unit of data transfer on the DBUS. The AES state transfer is processed on the DBUS in the column-major order, rather than the row-major order of linear and block modes. In a "read" operation, the plaintext state is cyclically-shifted into the ENC engine, and on a "write" operation the ciphertext state is cyclically-inverse-shifted into the DEC engine. Figure 5.2(c) shows the memory layout, where only one command (C0) is required to transfer two AES states (S0 and S1). Each state is processed in column major order (i.e., column-by-column) and cyclically-shifted/cyclically-inverse-shifted.

For example, assume the byte sequence in an AES state is from hexadecimal "0" to "3", "4" to "7", "8" to "b", "c" to "f" for the first, second, third, and fourth columns, respectively, as shown in memory sequence. The first write data sequence shown on the 64-bit DBUS is "0", "5", "a", "f", "4", "9", "e", "3", and the second write data sequence is "8", "d", "2", "7", "c", "1", "6", "b", which are cyclically inverse shifted before entering the DEC engine. Likewise, the first read data sequence is "8", "5", "2", "f", "c", "9", "6", "3", and the second read data sequence is 8, 5, 2, f, c, 9, 6, 3, which are cyclically shifted before enter the ENC engine.

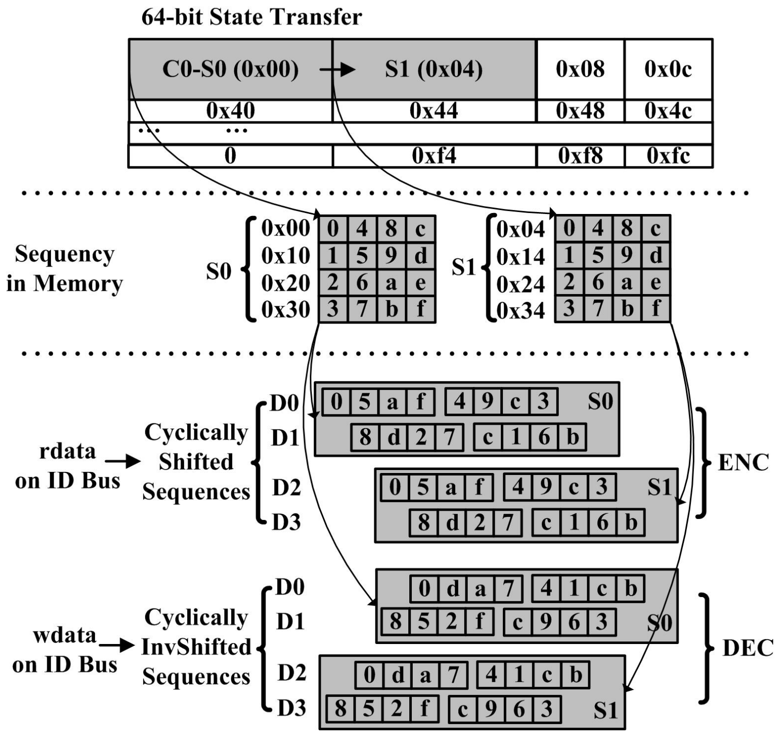
Figure 5.3 shows a timing diagram example of a 64-bit linear transfer. As a traditional data transfer mode, additional commands are required for each non-linear boundary-crossing operation of memory. Thus, 8 transfers, including command (C0 to C7) and data stages (D0 to D7), are necessary to access two 4×4-byte matrices. Moreover, IBUS provides the command preprocessing scheme and full-duplex bus operations. As shown in the figure, the command stages are consecutive and parallel with the data phases. The detailed information of the linear write and read processing are shown in Figure 5.3(b) and Figure 5.3(c), respectively. To



(a) Linear



(b) Block



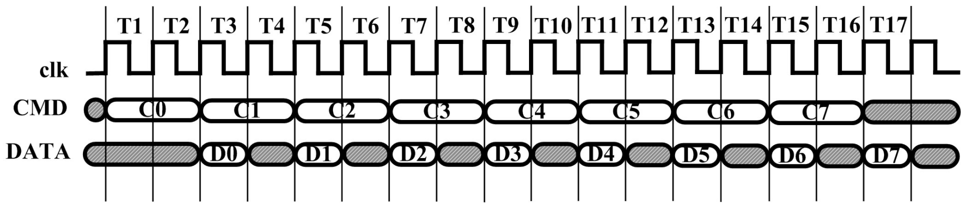
(c) State

Figure 5.2: Mem Access

illustrate the data sequence driven on IBUS, let "$\rightarrow$" operator indicate the associated memory address of the data in byte hereafter. Since the bus width is 64-bit in this case, only the write data bits from 63 to 32 are valid for the first to the fourth transfers (C0-D0 to C3-D3), and only the write data bits from 31 to 0 are valid for the fifth to the seventh transfers (C4-D4 to C7-D7), which are indicated by the "wdata_vld" signal as "8'hf0" and "8'h0f", respectively.

Apart from the conventional linear transfer, IBUS supports data transfer by block as well. The block mode defines all the block boundary-crossing addresses and the transfer size with the initial command. Thus, only two command stages (C0 and C1) are required to access two 4×4-byte matrices. As a timing diagram example of the 64-bit block transfer shown in Figure 5.4(a), it can be observed that the command stage of the second transfer (C1) is overlapped with the first and the second data stages (D0 and D1). Additionally, the detailed commands and data of the block write and read processing are shown in Figure 5.4(b) and Figure 5.4(c). The 4×4-byte block size is represented by the signals "len[9:6]" and "len[5:0]" as the column number (hexadecimal 4'h1) and the row number (hexadecimal "6'h4"). Similar to the linear write transfer example, 5.4(b) shows that only the write data bits from 63 to 32 are valid for the first ma-trix transfer (C0-D0 to D3), and only the write data bit from 31 to 0 are valid for the second matrix transfer (C1-D4 to D7), which are indicated by the "wdata_vld" signal as "8'hf0" and "8'h0f", respectively.

Finally, we consider the high-efficiency state transfer mode for the AES-encrypted systems. As a timing diagram example shown in Figure 5.5(a), only one command stage (C0) is required for the two-state (S0 and S1) transfer. Notice that the encryption/decryption processing starts at the T4 cycle immediately, since the first double word driven at the T3 cycle are cyclically-shifted/cyclically-inverse-shifted already using the specific state transfer type. More specifically, each AES state pro-



(a) Timing Diagram

C0: {addr, len, wdata_vld, wr}={32'h00, 12'h1, 8'hf0, 1'b1}
D0: wdata->{0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07}

C1: {addr, len, wdata_vld, wr}={32'h10, 12'h1, 8'hf0,1'b1}
D1: wdata->{0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x17}

……

C3: {addr, len, wdata_vld, wr}={32'h30, 12'h1, 8'hf0,1'b1}
D3: wdata->{0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37}

C4: {addr, len, wdata_vld, wr}={32'h00, 12'h1, 8'h0f, 1'b1}
D4: wdata->{0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07}

……

C7: {addr, len, wdata_vld, wr}={32'h30, 12'h1, 8'h0f, 1'b1}
D7: wdata->{0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37}

(b) Write Operation

C0: {addr, len, wr}={32'h00, 12'h1, 1'b0}
D0: rdata->{0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07}

C1: {addr, len, wr}={32'h10, 12'h1, 1'b0}
D1: rdata->{0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x17}

……

C3: {addr, len, wr}={32'h30, 12'h1, 1'b0}
D3: rdata->{0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37}

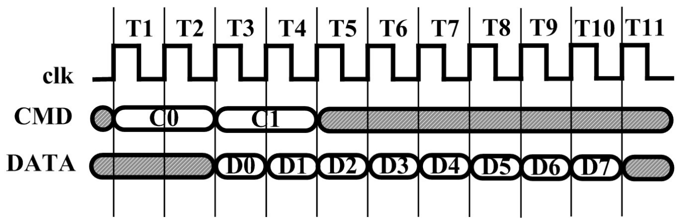
C4: {addr, len, wr}={32'h00, 12'h1, 1'b0}
D4: rdata->{0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07}

……

C7: {addr, len, wr}={32'h30, 12'h1, 1'b0}
D7: rdata->{0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37}

(c) Read Operation

Figure 5.3: IBUS Linear Transfer Processing.



(a) Timing Diagram

C0: {addr, len, wdata_vld, wr}={32'h00, 12'h444, 8'hf0, 1'b1}
D0: wdata->{0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07}
D1: wdata->{0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x17}
D2: wdata->{0x20, 0x21, 0x22, 0x23, 0x24, 0x25, 0x26, 0x27}
D3: wdata->{0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37}

C1: {addr, len, wdata_vld, wr}={32'h00, 32'h444, 8'h0f, 1'b1}
D4: wdata->{0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07}
D5: wdata->{0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x17}
D6: wdata->{0x20, 0x21, 0x22, 0x23, 0x24, 0x25, 0x26, 0x27}
D7: wdata->{0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37}

(b) Write Operation

C0: {addr, len, wr}={32'h00, 12'h444, 1'b0}
D0: rdata->{0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07}
D1: rdata->{0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x17}
D2: rdata->{0x20, 0x21, 0x22, 0x23, 0x24, 0x25, 0x26, 0x27}
D3: rdata->{0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37}

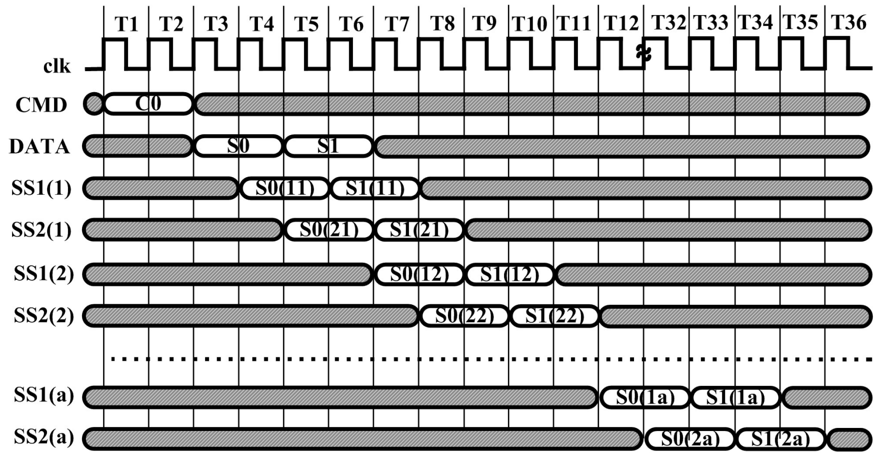
C1: {addr, len, wr}={32'h00, 32'h444, 1'b0}
D4: rdata->{0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07}
D5: rdata->{0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x17}
D6: rdata->{0x20, 0x21, 0x22, 0x23, 0x24, 0x25, 0x26, 0x27}
D7: rdata->{0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37}

(c) Read Operation

Figure 5.4: IBUS Block Transfer Processing.

cessing includes ten rounds, and each round of the encryption/decryption involves two sub-stages: SS1(n) and SS2(n), where "n" denotes the round number ranging from hexadecimal "1" to "a". For the write data process, the ciphertext states require ten-round decryption (SS1(1) & SS2(1) to SS1(a) & SS2(a)) before they can be written into memory. Likewise, for the read data process, the plaintext states require ten-round encryption (SS1(1) & SS2(1) to SS1(a) & SS2(a)) before they can be transferred on bus. Let S0(mn) and S1(mn) denote the first and the second states in the mth SS of the nth round, respectively. So "m" ranges from hexadecimal "1" to "2", which represents the first and the second SS, and "n" ranges from hexadecimal"1" to "a", which represents the first to the tenth round. Notice that the state processing within ten rounds of the same AES state are internal pipeline (from S0(m1) to S0(ma), or from S1(m1) to S1(ma)) and parallel (S0(1n) and S0(2n), or S1(1n) and S1(2n)), and the state processing among different AES states are external pipeline (from S0(mn) to S1(mn)). Therefore, for the 64-bit bus, the shifted plaintext states read from memory are continuous and the ciphertexts shown on the bus can be consecutive after 30-cycle encryption, and the inverse-shifted ciphertext states shown on bus are consecutive and the plaintexts written into memory can be continuous after 30-cycle decryption.

Figure 5.5(b) and Figure 5.5(c) show the detailed commands and data of the state transfer write and read operations. First, all the write data driven on IBUS are valid due to the specific state-unit operation of the state mode, which is indicated by the "wdata_vld" signal as hexadecimal "8'hff". Second, the read/write data is cyclically-shifted/cyclically-inverse-shifted before enter the ENC/DEC engine. As an example shown in Figure 5.5(c), the byte-unit memory addresses of the first word data, which are driven on the upper half term of the first double-word, are



(a) Timing Diagram

C0: {addr, len, wdata_vld, wr}={32'h00, 12'h802, 8'hff, 1'b1}
D0: wdata->{0x00, 0x13,0x22,0x31, 0x01, 0x10, 0x23, 0x32}
D1: wdata->{0x02, 0x11,0x20,0x33, 0x03, 0x12, 0x21, 0x30}
D2: wdata->{0x04, 0x17,0x26,0x35, 0x05, 0x14, 0x27, 0x36}
D3: wdata->{0x06, 0x15,0x24,0x37, 0x07, 0x16, 0x25, 0x34}

(b) Write Operation

C0: {addr, len, wr}={32'h00, 12'h802, 1'b0}
D0: rdata->{0x00, 0x11,0x22,0x33, 0x01, 0x12, 0x23, 0x30}
D1: rdata->{0x02, 0x13,0x20,0x31, 0x03, 0x10, 0x21, 0x32}
D2: rdata->{0x04, 0x15,0x26,0x37, 0x05, 0x16, 0x27, 0x34}
D3: rdata->{0x06, 0x17,0x24,0x35, 0x07, 0x14, 0x25, 0x36}

(c) Read Operation

Figure 5.5: IBUS State Transfer Processing.

hexadecimal 0x00, 0x11, 0x22, and 0x33. They are cyclically-shifted as the first column of the state input to the encrypter.

5.3.2 Transfer Latency Models

To estimate the IBUS transfer efficiency, we formulate and compare performance metrics of both AXI and IBUS in this subsection.

Let P_{XL} and P_{IL}, respectively, denote the probability of the back-to-back transfers of AXI and the probability of the back-to-back transfers of IBUS in a linear transfer. Since the command and data phases can be overlapped between two consecutive transfers, the AXI linear (XL) transfer latency, denoted by CY_{XL}, can be formulated as

$$CY_{XL} = 4 \times ceil(\frac{N_L}{XS}) + N_L - 2 \times ceil(\frac{N_L}{XS}) \times P_{XL}. \tag{5.17}$$

where P_{XS} ranges from 0 to [ceil(N_L/XS)-1]/ceil(N_L/XS). In this equation, the ceil() function represents that rounds fraction up. XS indicates the maximum AXI burst size, specified by ARLEN for read and AWLEN for write. It is 16 for AXI3 and 256 for AXI4 compatibility. In this equation, each AXI transfer requires four command cycles, two request, one address, and one response, when the response to any bus transfer is always available immediately, and all the command transactions are back-to-back.

In contrast, IBUS integrates the arbitration and address phases together, and also combines the data and slave-driven response phases. Therefore, it uses only two cycles with an immediate grant. The total latency of IBUS transfers, denoted by CY_{IL}, thus is

$$CY_{IL} = 2 \times ceil(\frac{N_L}{IS}) + N_L - 2 \times ceil(\frac{N_L}{IS}) \times P_{IL}. \tag{5.18}$$

where IS represents the maximum IBUS transfer size, which is 1024 beats for the 10-bit IBUS length signal. In this equation, $P_I L$ ranges from 0 to [ceil(N_L/IS)-1]/ceil(N_L/IS).

AXI protocol does not define how to access data by block. Hence, designers must consider the specific operations for the matrix-based applications and algorithms. Using the AXI linear transfer type, the total cycles of a block processing can be calculated as

$$CY_{XB} = 4 \times N_H \times ceil(\frac{N_W}{XS}) + N_H \times N_W - 2 \times N_H \times ceil(\frac{N_W}{XS}) \times P_{XB}. \quad (5.19)$$

Here, P_{XB} represents the probability of the back-to-back AXI block transfers, which ranges from 0 to [N_H*ceil(N_W/XS)-1]/[N_H*ceil(N_W/XS)].

Due to the built-in boundary crossing scheme of the block transfer, each matrix operation consumes only one command stage for IBUS. The total cycle cost of an IBUS block transfer can be formulated as

$$CY_{IB} = 2 \times ceil(\frac{N_H}{DH}) \times ceil(\frac{N_W}{DW}) + N_H \times N_W - 2 \times ceil(\frac{N_H}{DH}) \times ceil(\frac{N_W}{DW}) \times P_{IB}. \quad (5.20)$$

where DH and DW are the maximum block height and the maximum block width that can be processed by the IBUS block transfer. As an example, DH is 32 for a 5-bit block height signal, and DW is 16 for a 4-bit block width signal. $P_I B$ denotes the probability of the back-to-back IBUS block transfers, which ranges from 0 to [ceil(N_H/DH)*ceil(N_W/DW)-1]/[ceil(N_H/DH)*ceil(N_W/DW)].

Finally, the AES cipher latency on AXI and IBUS is considered. To evaluate the AES state transfer performance, we implement the AES engines using the composite filed arithmetic, based on the 10-round algorithm with 4-word key, and the GF$(((2^2)^2)^2)$ construction with $\phi=\{10\}_2$ and $\lambda=\{1100\}_2$.

Considering the whole system performance, we split each AES round into two subsections: subsection1 (SS1) merging the non-linear functions δ/IA $\times\delta$ with the modular inversion operator, and subsection2 (SS2) integrating SR/ISR, I$\delta \times A$/Iδ, MC/IMC, and AK transformations. Assume that the encryption/decryption processing is full pipeline, each cipher round thus uses 5 clock cycles for the 32-bit bus, in which 4 cycles are consumed by SS1 and 4 cycles are consumed by SS2 with 3 cycles overlapped. Moreover, assume that all the transfers are back-to-back, and the command stages, data stages, and AES cipher operations are completely overlapped. Therefore, the number of cycles spent by AXI bus to transfer N_E AES states (AS) can be calculated as

$$CY_{AS} = 4 \times 4N_E + 4N_E - 2 \times 4N_E \times P_{AS}. \tag{5.21}$$

where the back-to-back probability of AXI state transfers, denoted as P_AS, ranges from 0 to ($4N_E$-1)/$4N_E$. Furthermore, for the AES cipher/inverse-cipher processing, the state transfer consumes not only the command and data cycles on bus, but also the AES encryption/decryption latency. Assume that the encryption/decryption processing is full pipeline, each cipher/inverse-cipher round uses 5 clock cycles for the 32-bit bus, in which 4 cycles are consumed by SS1 and 4 cycles are consumed by SS2 with 3 cycles overlapped. Likewise, 3 cycles are needed for the 64-bit bus and 2 cycles are needed for the 128-bit bus to complete each AES state round. Furthermore, assume that all the transfers are back-to-back, and the command stages, data stages, and AES cipher/inverse-cipher operations are completely overlapped. The total number of cycles spent by the 32-, 64-, and 128-bit AXI encryption/decryption (XE) procedures are

$$CY_{XE32} = 2 + 6 \times N_E + 50 \times N_E - (12 \times N_E + 38 \times N_E) \times P_{XE}. \tag{5.22}$$

$$CY_{XE64} = 2 + 4 \times N_E + 30 \times N_E - (6 \times N_E + 24 \times N_E) \times P_{XE}. \tag{5.23}$$

Table 5.2: Modeling Performance Comparison

Tests	CY
XL	$(4-2P)ceil(\frac{N_L}{XS}) + N_L$
IL	$(2-2P)ceil(\frac{N_L}{IS}) + N_L$
XB	$(4-2P) \times N_H \times ceil(\frac{N_W}{XS}) + N_H \times N_W$
IB	$(2-2P) \times ceil(\frac{N_H}{DH}) \times ceil(\frac{N_W}{DW}) + N_H \times N_W$
XE32	$2 + 2N_E(28-25P)$
XE64	$2 + 2N_E(17-15P)$
XE128	$2 + N_E(23-20P)$
IE32	$2 + 2N_E(27-25P)$
IE64	$2 + 2N_E(16-15P)$
IE128	$2 + N_E(21-20P)$

$$CY_{XE128} = 2 + 3 \times N_E + 20 \times N_E - (3 \times N_E + 17 \times N_E) \times P_{XE}. \tag{5.24}$$

Notice that the back-to-back probability of AXI cipher test ranges from 0 to (N_E-1)/N_E .

For the specific state transfer mode of IBUS, only one command is required for a write or read operation with less than or equal to 1024 states, due to the 10-bit width definition of the "id_len[9:0]" signal. The number of processing cycles depends on the IBUS size. For instance, 4N, 2N, and N cycles are needed to transfer N states for the 32-, 64-, and 128-bit IBUS, respectively. Therefore, the total cycles consumed by IBUS encryption/decryption (IE) tests are

$$CY_{IE32} = 2 + 4 \times N_E + 50 \times N_E - (4 \times N_E + 46 \times N_E) \times P_{IE}. \tag{5.25}$$

$$CY_{IE64} = 2 + 2 \times N_E + 30 \times N_E - (2 \times N_E + 28 \times N_E) \times P_{IE}. \tag{5.26}$$

$$CY_{IE128} = 2 + N_E + 20 \times N_E - (N_E + 19 \times N_E) \times P_{IE}. \tag{5.27}$$

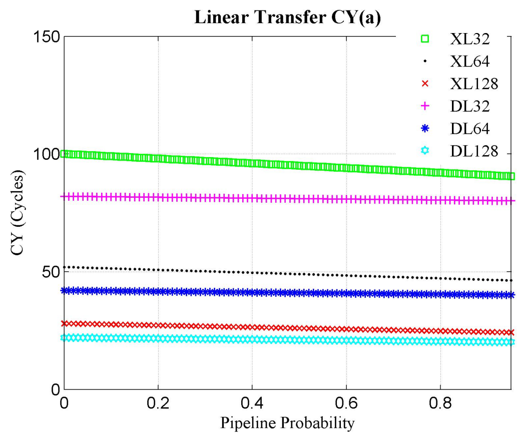
for the 32-, 64-, and 128-bit IBUS, respectively. In equation 5.25, equation 5.26, and equation 5.27, 50, 30, and 20 cycles are used to encrypt/decrypt one AES state. Table 5.2 simplifies and summarizes all the above analysis.

5.3.3 Static Performance Analysis

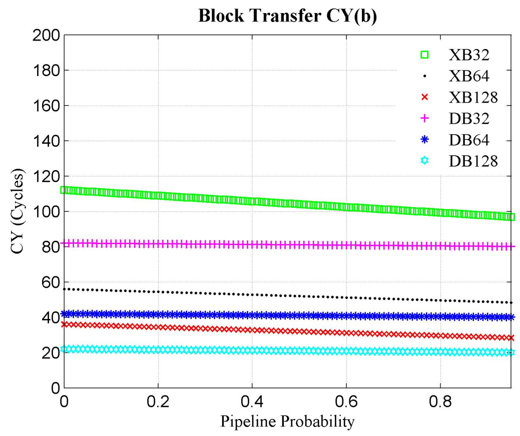
The difference between AXI and IBUS with different bus sizes is made clear through the analysis of cycle cost in Figure 5.6. Assume that the total state number (N) is 10, which is the smallest state number for a ten-round pipeline and parallel processing of encryption & decryption. The horizontal axis represents the back-to-back probability (P) ranging from 0 to 0.95 in this case.

As the latency of linear test cases, involving XL and IL shown in Figure 5.6(a), the clock cycles consumed by the IL transfers are 88.51%, 86.61%, and 83.06% for all the 32-, 64-, and 128-bit bus sizes, compared with the XL tests when P reaches the maximum (0.95). Likewise, the clock cycles consumed by the IB transfers are 82.75%, 82.85%, and 70.77%, respectively, compared with the XB tests, for all the three bus sizes' tests, as shown in Figure 5.6(b). The comparison between AXI and IBUS cipher tests are further shown in Figure 5.6(c). For the same bus size, the IE test consumes less cycles than the XE transfer. As an example, when P is the maximum 0.95, the clock cycles consumed by IBUS transfers are 76.74%, 64.29%, and 51.22% compared with AXI transfers for 32-, 64-, and 128- buses, respectively.

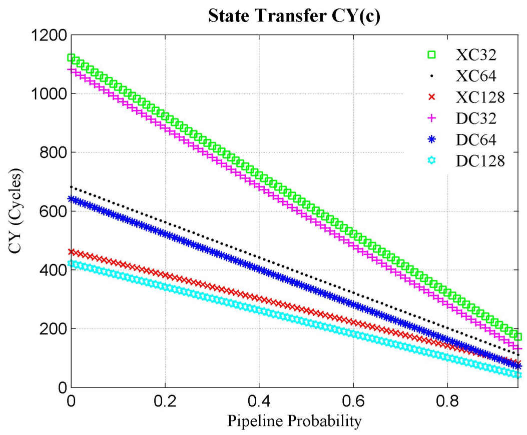
In what follows, the resource costs are considered statically. To realize the ENC/DEC engine, we need to pay for large overhead logics and optimize the number of parallel resource costs. Figure 5.7 shows the pipeline structures and the resource costs depending on different bus-based implementations. Let S and M denote the logic utilization of SS1 and SS2, respectively. When the bus size is 32-bit shown in Figure 5.7(a), four parallel S (4S) connected with one M (1M) instances are necessary to internally pipeline and parallel all the ten-round cipher/inverse-cipher processing per state. In order to externally pipeline all the ten rounds among different states, the hardware resource must be duplicated ten times as the overhead for high-speed transfers.



(a) Linear



(b) Block



(c) State

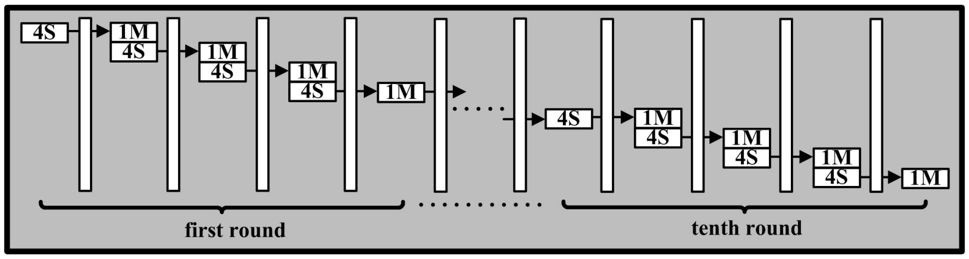
Figure 5.6: Static Performance Analysis.

Furthermore, the resources are doubled to externally parallel the write and read channels of the full-duplex bus. As the 64-bit bus-based implementation shown in Figure 5.7(b), the cipher/inverse-cipher processing can be sped up, but the S and M instances are doubled. It requires 8 S (8S) and 2 M (2M) instances for the encryption/decryption process of each round to make all the data transfer internal pipeline and parallel. Similarly, sixteen S (16S) and four M (4M) of each round are necessary to the 128-bit bus-based implementation shown in Figure 5.7(c). To externally pipeline different states and parallel the write & read channels, both 64- and 128-bit bus-based designs require ten-time duplication, and then double the S and M instances.

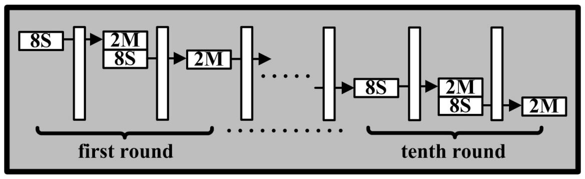
As an alternative technology to the ASIE design, FPGA implements the basic combinational logic by the 2^k-bit static random-access memory (SRAM), which represents a K-input and one-output LUT. Different from the logic gate computation, it is capable of realizing any Boolean function of up to K variables by loading the SRAM cell with the truth table of that function. Therefore, although the 128-bit bus design costs more S and M instances, it reduces the FPGA slice usage due to the short path of each cipher/inverse-cipher round.

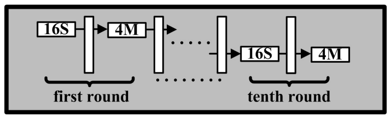
5.4 Hardware Implementation

This section presents all the 32-, 64-, and 128-bit implementations using AXI and IBUS architectures, targeted to accurately evaluate the architectural performance. We use Verilog HDL [20] to complete the RTL design, and set up a UVM [16, 18] environment to verify all the DUT. Finally, the FPGA back-end flow is performed to estimate the area cost and power consumption.



(a) 32-bit bus





(b) 64-bit bus (c) 128-bit bus

Figure 5.7: Pipeline Structures of AES Cores.

5.4.1 RTL Design and Verification

In our study, we implemented all the 32-, 64-, and 128-bit AXI and IBUS DMA, combined with AES core and memory controller (XDAM and IDAM), in order to compare the power-area-throughput performance among different tests. As an example, the IDAM structure with ENC and DEC engines is shown in Figure 5.8. The memory controller is is used to provide the control interface for external memory, and address mapping from hexadecimal 0x00 to 0xff.

Generally, IBUS is a dual-bus structure, including a compact control bus (IC) and a high performance data bus (ID). As one of the IC bus's slaves, the IDAM is configured by the IC bus's single master, the micro-processor. Its functional registers include control, status, and round key registers. In addition, as the single slave of ID bus, the IDAM can be accessed by all the masters located on ID bus. All the requests are granted sequentially, according to each master's priority configured through IC bus. The IDAM arbiter performs this function by observing a number of different, and deciding which is currently the highest priority master. In the scheduler, all the bus requests can be preprocessed using the command queues. Since the queue level is four for both write and read, the maximum command number can be pushed into the buffer is eight, four read and four write. After the memory interface is released, the commands can be popped out, and then translated to be memory side commands by the memory command controller and the address mapping modules. The data path modules, write data path and read data path, are used to multiplex cipher and non-cipher data processing between ID bus masters and memory. More specifically, the AES ENC/DEC engine is bypassed for the conventional linear and block transfers. For the cipher tests, such as XE and IE, the write data path decrypts the ciphertexts then writes the plaintexts into memory, or the read data path encrypts the plaintexts from memory then transfer the ciphertexts to the bus.

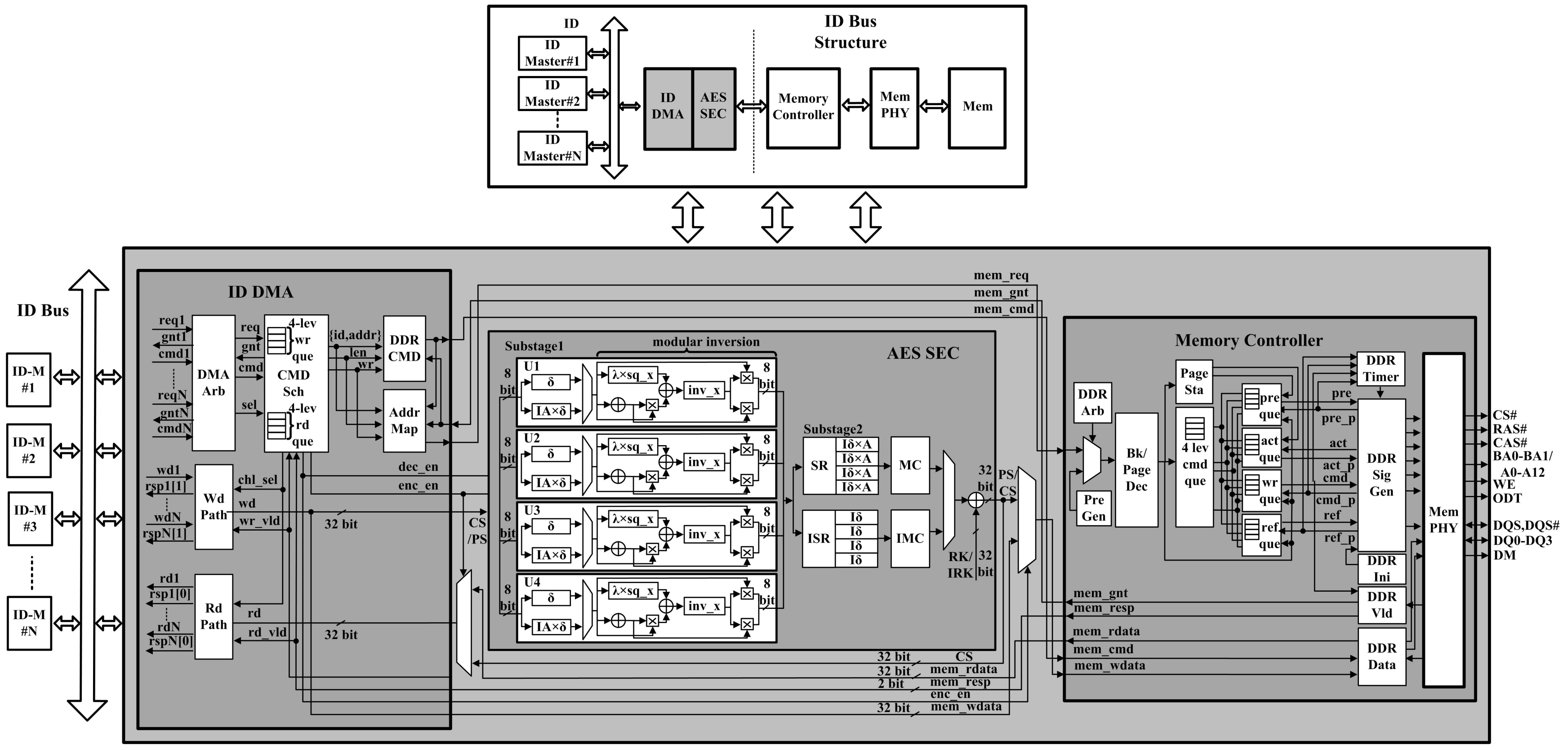


Figure 5.8: IDAM Structure.

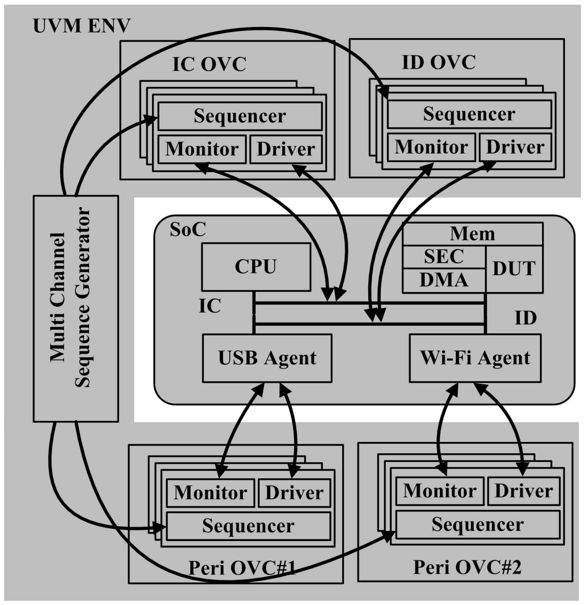


Figure 5.9: UVM Environment.

To verify DUT and evaluate transfer performance, we build up a UVM-based verification environment shown in Figure 5.9. It integrates four encapsulated ready-to-use and configurable verification agents: the only master of IC bus denoted as IC bus OVC (micro-processor), the only slave of ID bus denoted as the ID bus OVC (Memory Controller), and two ID bus masters indicated as Peripheral OVC #1 (USB2.0 Host Controller) and Peripheral OVC #2 (Wi-Fi Mac) in the figure. Each of them contains three components: the sequencer, driver, and monitor. The driver is an active entity that emulates logic that drives the DUT. It repeatedly receives a data item and drives it to the DUT by sampling and driving DUT signals. The sequencer is an advanced stimulus generator that controls the items that are provided to the driver for execution. The monitor is a passive entity that samples DUT signals but does not drive them. It collects coverage information and performs checking. The sequence generator is a control center to sync up all the OVC sequencers.

The typical test cases used in our study are that 40 words, 10×4 words, and 10 AES states are written into memory then read out, respectively, using linear, block, and state transfer modes. For the non-cipher tests, including linear and block cases, the ENC/DEC engine is bypassed by IDAM and XDAM. Otherwise, the AES core is used to encrypt or decrypt data for the cipher tests. As an example, the USB2.0 agent initiates a 10-state write command to the data bus. The initial address is hexadecimal 0x00 and the data are encrypted initial states. Then, IDAM/XDAM responds to the request, decrypts the ciphertexts, and then writes the plaintexts into memory. After the first state is written into the memory, the Wi-Fi Mac agent requests a 10-state read operation to the same memory address immediately. Paralleling with the write operations, IDAM/XDAM responds the request, reads data out and encrypts the plaintexts to be ciphertexts, and then sends them on the data bus. During the data transfers, the control bus is responsible for initiating the AES round keys, controlling the DAM execution, handling the interrupts, and monitoring the bus status.

5.4.2 Area and Power Analysis

All the 32-, 64-, and 128-bit XDAM and IDAM are synthesized and placed & routed using Xilinx ISE 14.7 with the target device Virtex6xc6vlx550t-2ff1760 [14]. Then, several fully placed & routed NCD files and physical constraint PCF files are generated. Table 5.3 shows the synthesis results including IO number, resource utilization, and MOF. As shown in the second column, it can be observed that IDAM uses less IO ports than XDAM for the same bus size. For the different bus sizes, it is obvious that the 128-bit bus costs much more IOs than 32- and 64-bit buses. In the third column, the total number of occupied slices of IDAM designs are less than the AXI

Table 5.3: Resource Comparison

Resource Costs	IOs	Slices	MOF (MHz)
32-bit XDAM	533	26106	133.010
32-bit IDAM	324	24822	183.636
64-bit XDAM	661	22603	131.528
64-bit IDAM	460	21319	176.154
128-bit XDAM	917	18344	130.152
128-bit IDAM	732	17060	184.176

based implementations for all the 32-, 64-, and 128-bit bus structures. Moreover, the compact IBUS structure achieves higher operational clock frequency than AXI based designs for all the three bus sizes, which is shown in the fourth column.

Furthermore, inputting all the NCD and PCF files, as well as the simulation VCD files into the XPower Analyzer tool, the power statistics of AXI- and IBUS-based designs are obtained in Table 5.4. Since static power consumption is mostly determined at the circuit level, the static power of the same design is almost a constant for different test cases, as shown in the second column. Our work, thus, concentrates on analyzing dynamic power shown in the third column.

First of all, it can be observed that the DBUS tests consume less dynamic power compared with AXI tests, because of the less toggle rate of logic, signal, and IO (LSIO). In addition, the wider bus consumes more DP in all the block and cipher tests. In the linear mode, however, the 32-bit bus consumes more dynamic power than the 64-bit bus, because the LSIO switching rate is very low in this test and the clock power becomes the dominant factor of the dynamic power consumption.

5.5 Experimental Results

In this section, we summarize the experimental results involving cycle cost, valid bandwidth, dynamic energy, slice efficiency, and dynamic energy efficiency in Table

Table 5.4: Power Consumption

Test Cases	SP (mW)	DP (mW)	TP (mW)
32-bit XL	3799	612	4411
64-bit XL	3796	577	4373
128-bit XL	3797	623	4420
32-bit IL	3796	574	4370
64-bit IL	3794	540	4335
128-bit IL	3796	584	4381
32-bit XB	3801	752	4553
64-bit XB	3812	971	4783
128-bit XB	3826	1263	5089
32-bit IB	3798	695	4493
64-bit IB	3802	927	4729
128-bit IB	3828	1226	5054
32-bit XE	3805	771	4576
64-bit XE	3818	1063	4881
128-bit XE	3852	1747	5599
32-bit IE	3802	716	4518
64-bit IE	3816	995	4810
128-bit IE	3847	1650	5497

Table 5.5: Experimental Results Comparison

Tests	CY	VDB (GBps)	DE (uJ)	SE (KBps/Slice)	DEE (GBps/J)
XL32	92.00	0.70	0.56	26.65	1.14
XL64	48.00	1.33	0.28	58.99	2.31
XL128	26.00	2.46	0.16	134.19	3.95
IL32	82.00	0.78	0.47	31.44	1.36
IL64	42.00	1.52	0.23	71.48	2.82
IL128	22.00	2.91	0.13	170.52	4.98
XB32	98.00	0.65	0.74	25.02	0.87
XB64	50.00	1.28	0.49	56.63	1.32
XB128	30.00	2.13	0.38	116.30	1.69
IB32	82.00	0.78	0.57	31.44	1.12
IB64	42.00	1.52	0.39	71.48	1.64
IB128	22.00	2.91	0.27	170.52	2.37
XE32	172.00	0.37	1.33	14.25	0.48
XE64	112.00	0.57	1.19	25.28	0.54
XE128	82.00	0.78	1.43	42.55	0.45
IE32	132.00	0.48	0.95	19.53	0.68
IE64	72.00	0.89	0.72	41.69	0.89
IE128	42.00	1.52	0.69	89.32	0.92

5.5. In the practical tests, read commands follow write commands to verify the memory accessing correctness. Thus, the read and write transfers are not completely overlapped.

It is clear to illustrate the performance comparison between IDAM and XDAM in Figure 5.10. In Figure 5.10(a) and Figure 5.10(b), the performance ratios for non-cipher tests, including linear and block cases, are shown. Since all the time consumption ratios are less than 1, IBUS consumes less time than the AXI for all the three bus sizes' implementations. Particularly for the block tests, the latency of IBUS are 83.67%, 84.00%, and 73.33%, respectively, compared with AXI for all the 32-, 64-, and 128-bit buses. Additionally, the dynamic energy, which is the integral of dynamic power, or the product of average dynamic power and transfer time, is considered in our study. Although the dynamic power consumed by IDAM and XDAM are close to each other, the dynamic energy consumption of IL tests are

83.60%, 81.89%, and 79.32%, respectively, compared with the XL tests, and the dynamic energy consumption of IB tests are 77.33%, 80.19%, and 71.19%, respectively, compared with the XB tests, for all the 32-, 64-, and 128-bit bus implementations. Furthermore, based on the fair assumption of the same operational clock frequencies for IBUS and AXI, the conventional bandwidth between full-duplex IBUS and AXI are the same. However, the valid data bandwidth of IBUS surpasses AXI due to the high efficient structure. For example, the valid data bandwidth of IL test is 1.18 times of XL test, and the valid data bandwidth of IB test can reach 1.36 times of XB test, when the bus size is 128 bit.

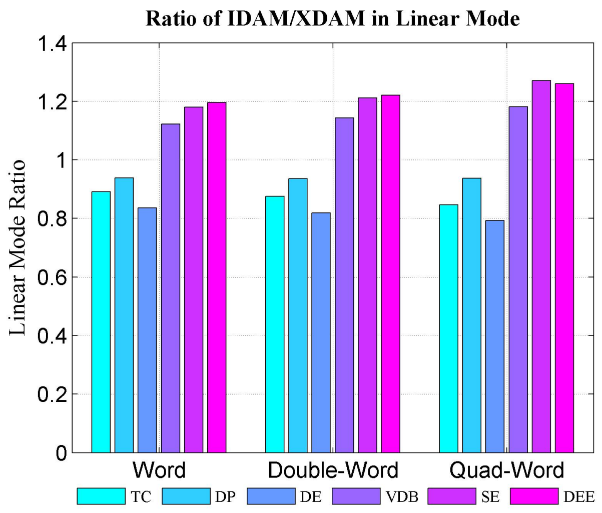
In order to evaluate the area-efficiency, slice efficiency is also computed in terms of valid data number that can be transferred per second per slice. It can be observed that the slice efficiency of IL tests are around 1.27 times of XL tests, and the slice efficiency of IB test is 1.47 times compared with XB test when the bus size is 128 bits. Then, dynamic energy efficiency is further defined in terms of valid data number that can be transferred per second per watt, or valid data number that can be transferred per joule. The dynamic energy efficiency of IL tests are around 1.26 times compared with XL tests for all the three bus-size designs, and the dynamic energy efficiency of IB test can reach 1.40 times of XB test when the bus size is 128 bits. In other words, IBUS can transfer 1.40 times as much data as AXI with the same time and power consumption in this case.

In this chapter, we focus on comparing the cipher test performance shown in Figure 5.10(c). Using the high-efficiency state transfer mode for the AES-encrypted circuits, the IE tests achieve higher performance than the AXI tests. First, the time spent by IE tests are 76.74%, 64.29%, and 51.22%, respectively, compared with XE tests for 32-, 64-, and 128-bit buses. Second, the dynamic energy consumed by the IE tests are 71.27%, 60.17%, and 48.38% compared with the XE tests for the 32-, 64-, and 128-bit buses, respectively, though the dynamic power of IE tests and XE tests are very close to each other. Third, the conventional bandwidth and valid data bandwidth of the IE transfers can reach 2.95 GBps and 1.52 GBps, respectively, on the 128-bit IBUS. The IDAM/XDAM valid data bandwidth ratios are 1.30, 1.56, and 1.95, respectively, when the bus size is 32, 64, and 128 bits. Finally, we consider the slice efficiency and dynamic energy efficiency of all the AXI and IBUS tests. The 128-bit IE test can transfer 89.32 Kbytes per second per slice cost. As the highest slice efficiency of all the cipher tests, it is 2.10 times compared with the 128-bit XE test. Additionally, the dynamic energy efficiency of the IE tests are 1.40, 1.66, and 2.07 times compared with the XE tests for the 32-, 64-, and 128-bit buses, respectively. That means, IBUS can transfer 2.07 times as much data as AXI with the same time and power consumption when bus sizes are 128 bits.

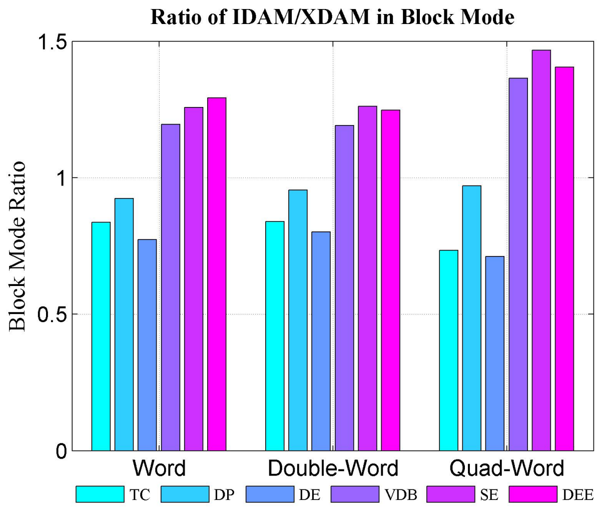
In conclusion, the system performance of IBUS surpasses that of AXI due to the specific block and state transfer modes. Moreover, the 128-bit implementations cost more IOs and dynamic power, but achieves a higher slice and dynamic energy efficiency than 32- and 64-bit buses, for all the linear, block, and cipher transfer tests. Considering the design requirements and resource limitation, designers can choose different bus sizes based implementations.

5.6 Summary

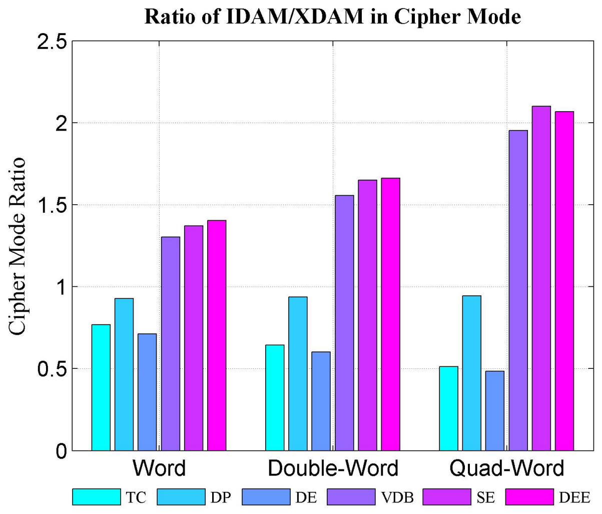
In this chapter, we further propose an advanced IBUS architecture for the AES-encrypted embedded systems. To date, it is the first performance analysis on the bus architectural level for the AES-encrypted chips. As the results, the IBUS based designs cost less in terms of hardware resource than the AXI based implementations, and the IBUS cipher tests achieve higher valid bandwidth and consume less dynamic



(a) Linear



(b) Block



(c) Cipher

Figure 5.10: Performance Comparison.

power than AXI. Based on the IBUS architecture, we also evaluate the performance on different bus size implementations. To sum up, the 128-bit design achieves higher valid bandwidth, but consumes more dynamic power than the 32- and 64-bit designs. In contrast, the 32-bit DAM consumes the least power, but sacrifices bandwidth and area. Based on the resource and performance requirements, a user can choose the proposed IBUS implementations to fulfill the tradeoffs of different applications.

With the emerging area of IoT, leveraging limited resource cost for embedded chips and overhead cost for security mechanisms at the SoC level are challenging issues. We believe that our work provides a solution that uses a high-efficiency bus architecture for the AES-encrypted circuits to meet high-security and high-performance chip requirements.

CHAPTER 6

A CONFIGURABLE AND SYNTHESIZABLE IBUS ARCHITECTURE FOR INTEGRATING INDUSTRIAL STANDARD IPS

This chapter proposes a configurable and synthesizable IBUS architecture for integrating IP blocks from multiple vendors. More precisely, we combine three novel bus transfer modes: linear, block, and AES state, into IBUS protocol to improve bus performance and structural support for the AES algorithm. We also present an auto-generated method, capable of creating IBUS structure to seamlessly integrate third-party IPs. The results show large reduction in IBUS architecture area and energy consumption (66.19% in cipher tests), compared with AXI3-based integration. Moreover, IBUS-based designs achieve higher valid throughput (up to 1.48×) than AXI3 implementations.

6.1 Related Work

With industry expectations of billions of new smart connected things today, commonly referred to as the IoT, we see a growing demand for highly cost-effective, power-efficient, and secure circuits. However, the existing on-chip bus architectures, such as AHB [1] and AXI [5] from ARM Holdings, Wishbone from Silicore Corporation [6], and OCP from OCP-IP [4], are much more costly in terms of both area and energy consumption, due to a large number of IO and signal definitions. In addition, all the aforementioned buses transfer data in the linear-major order, which is very low efficient to process the matrix-based arithmetic, such as the industrial widely used AES algorithm [19]. Under this context, the brief [186] has proposed a low-cost and low-power bus architecture, IBUS, for the small-scale and energy-limited chips.

Furthermore, a specific AES state-based (AS) transfer mode, based on the ID bus protocol, has been presented in [178]. The motivation for using the AS transfer is that it balances the embedded chip's performance with the overhead costs of the AES algorithm, and provides a very cost-effective on-chip bus structure.

However, new protocols, such as [186] and [178], also create new issues to integrate the industrial standard IPs from multiple vendors. The growing number and complexity of IP blocks and subsystems in today's SoC designs challenge even the most experienced design teams, especially when the on-chip bus architecture is based on the protocol that is new. This is one reason why standard buses are still the predominant structure of choice in many embedded chips, even though they are not suitable for the specific IoT embedded chips. To overcome the IP integration problem, earlier work mainly focused on static bridge designs [44, 75]. Since the static approaches are inherently non-scalable and limited in the ability to provide high performance in cases where the traffic characteristics vary dynamically, a number of automatic designs were proposed to dynamically optimize the bus-based architectural topology [98, 52, 151]. Moreover, several design methodologies and design flow for customizing these architectures to adapt to traffic characteristics have further been studied [79, 48, 191]. However, many of these research aim at exploiting theories on the system-modeling level, making the implementations of the design flow difficult, sometimes unfeasible tasks. Additional problems arise from the high abstract models, making it hard to achieve high accuracy of the system performance, especially in the early development stages.

In this chapter, we further propose a configurable and synthesizable IBUS architecture for integrating industrial standard IPs. To the best of our knowledge, this is the first analysis and implementation of architecture generation for high-performance and high-security IoT embedded chips, by using the real integrated circuit design flow. The specific contributions of our works are as follows.

- We propose a high efficient IoT on-chip bus protocol that combines the traditional linear transfer mode, and the novel block and AES state transfer types [186, 178], in order to improve IoT embedded chip performance and security.
- Based on IBUS protocol, we also propose a configurable and synthesizable on-chip bus structure for integrating industrial standard IPs. The trade-offs among resource cost, speed, and power consumption of the IP integration are considered and analyzed in the architectural generation algorithm.
- In order to meet the chip requirements, the system performance is estimated in this chapter as well. Four DUTs, together with user-configured and auto-generated bus wrappers, are implemented and evaluated as case studies: the AXI3 DMA with AES engine and memory controller (XDAM), IBUS DMA with AES engine and memory controller (IDAM), XDMA connected with bridges (XDAM-B), and IDAM connected with wrappers (IDAM-W).

The remainder of this chapter is organized as follows: Section 6.2 introduces the IBUS architecture and bus protocol, and section 6.3 presents the solution of IP integration using the IBUS architecture. In section 6.4, the case studies are illustrated, and then the experimental results are analyzed and compared. Finally, section 6.6 concludes this paper.

6.2 IBUS Architecture

In this section, we introduce the IBUS structure, and then briefly explain the bus protocol.

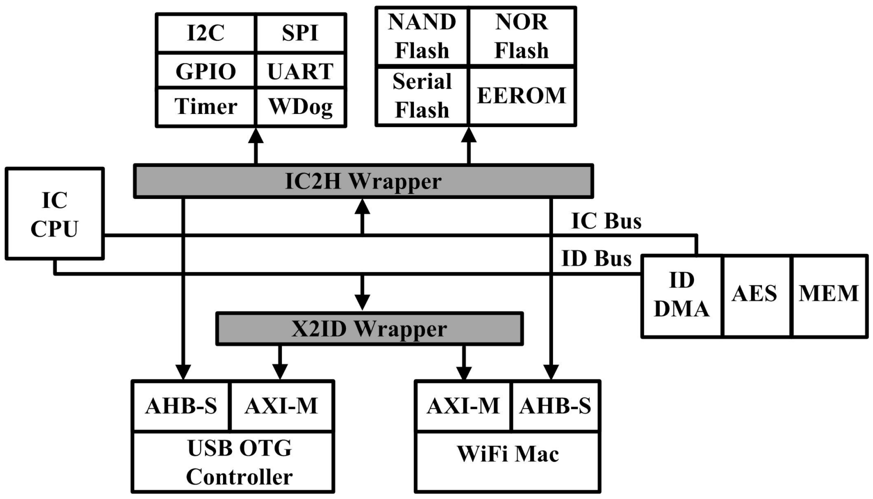


Figure 6.1: IBUS Integration Structure.

6.2.1 IBUS Structure

Figure 6.1 shows the IBUS structure composed of the control bus (IC) and data bus (ID). The IC bus mainly takes charge of low-speed and low-bandwidth functional register configuration with a reduced interface and minimal power consumption, and ID bus is mainly responsible for high-bandwidth data transfers with full-duplex interface and high efficient transfer types.

The wide adoption of the AMBA specification throughout the semiconductor industry has driven a comprehensive market in AMBA-based IP products. To integrate these IPs, overhead costs should be added to convert bus protocols and deal with clock domain crossing issues. As an example shown in Figure 6.1, the IC2H wrapper is insert between the IC bus masters and AHB-controlled slave interfaces. In addition, the X2ID wrapper is resided between AXI-based master interfaces and ID bus slaves.

6.2.2 IBUS Protocol

To optimize the on-chip communication for AES-encrypted circuits, we combine the linear, block, and AES state transfer modes into the IBUS protocol. The bus attributes, including request arbitration, slave response, and transfer types, are briefly introduced as follows.

Arbitration: Bus arbitration is necessary for the multi-master buses, such as AXI and ID bus. It ensures that only one master has access to the bus at any one time. If the ID bus grant acknowledges the request immediately, a total transfer of an ID command takes at least two cycles for synchronous clock domains. As a single-master bus, IC bus does not require the arbitration scheme, thus only one master cycle is consumed by each IC command.

Slave Response: The handshaking between IC command and data valid synchronizes all the signals crossing between master and slave domains, and avoids command queue overflows. Each IC bus slave must send a response within a timeout window. Otherwise, the handshaking is a timeout, and the command is indicated as ERROR, and should be RETRY or DESCARD. For ID bus, a two-bit response signal indicates that the slave is ready to accept the command and associated data, the higher bit for write and the lower bit for read.

Transfer Type: The IC bus, on which a large number of peripherals are located, is defined as a low-speed and low-power bus. Hence, it only supports the SINGLE transfer mode and unpipelined protocol.

In contrast, three transfer types: linear, block, and AES state, are combined into the ID bus. In the linear type, the signal transfer length, denote as N_L, gives the exact number of beats in the row-major order. Notice that the number of beats in a transfer is not the number of data bytes. The total amount of data transmitted or received in a linear transfer, denoted as TB_L, can be calculated by multiplying the number of beats by the amount of data in each beat.

$$TB\_L = N_L \ll buswidth. \tag{6.1}$$

where the bus width values of 0, 1, 2, 3, and 4, respectively, represent bus size as byte, half word, word, double-word, and quad word. The shift operator "$\ll$" performs left shift of $N_H \times N_W$ by bus width.

Moreover, N_H represents the matrix height and N_W represents the matrix width (unit in burst beat) in the block mode. Hence, the total block transfer bytes,denoted as TB_B, can be calculated as:

$$TB\_B = (N_W \ll buswidth) \times N_H. \tag{6.2}$$

In the AES state mode, N_C represents the number of AES states in the non-linear major order. More specifically, the write data is in the column-major and byte-deinterleaving order, while the read data is in the column-major and byte-interleaving order. The transfer bytes of state mode, denoted as TB_S, can be calculated as

$$TB\_S = (N_C \ll 4). \tag{6.3}$$

6.3 IBUS SoC Integration

In our study, the IBUS structure can be automatically generated following the user-configured parameters, including bus width and operational frequency, the interface size of integrated IPs, the typical data processing length of the applications, and so forth. The generated logic is optimized considering tradeoffs among area, speed, and power consumption. As a case study, we present an IBUS structure generation with two specific wrappers, IC2H and X2ID, in this section.

6.3.1 AXI and IBUS Static Analysis

Before discussing the logic generation algorithm, we first discuss the transfer speed of the AXI and ID buses. The valid data frequency (VDF) is defined as the valid data without protocol overhead that can be transferred in one second (unit in MHz). It can be formulated as

$$VDF = OF \times (\frac{N}{CY}). \tag{6.4}$$

where OF represents the operating frequency, and N is the valid data transferred in CY cycles. The data crossing between AXI and ID bus can achieve the highest efficiency if the VDFs of two bus sides are balanced. In other words, the AXI OF divided by the ID OF, denoted as XOF/IDOF, should be equal to the bus latency ratio of (CY_{IL}/CY_{XL}), (CY_{IB}/CY_{XB}), and (CY_{IE}/CY_{AS}), respectively, for the linear, block, and cipher modes, to balance the data operations between AXI and ID buses. The bus latency ratio in the linear mode can be represented as

$$\frac{CY_{IL}}{CY_{XL}} = \frac{(2-2P_{IL}) \times ceil(\frac{N_L}{IL}) + N_L}{(4-2P_{XL}) \times ceil(\frac{N_L}{XL}) + N_L} \tag{6.5}$$

This ratio indicates the difference of valid data rates between AXI and ID bus. For example, ID bus and AXI consume 82 and 92 cycles, respectively, to process 80-word data when both P_{IL} and P_{XL} reach the maximum probabilities. Similarly, the ratio of CY_{IB}/CY_{XB} in the block mode can be written as

$$\frac{CY_{IL}}{CY_{XL}} = \frac{(2-2P_{IB}) \times ceil(\frac{N_H}{DH}) \times ceil(\frac{N_W}{DW}) + N_H \times N_W}{(4-2P_{XB}) \times N_H \times ceil(\frac{N_W}{XL}) + N_H \times N_W} \tag{6.6}$$

When processing a 20×16 pixel grayscale picture, ID bus and AXI, respectively, use 82 and 122 cycles with the maximum back-to-back probabilities. In the cipher mode, the ratio of CY_{IE}/CY_{AS} can be formulated as

$$\frac{CY_{IE}}{CY_{AS}} = \frac{2 + 2N_C \times (27 - 25P_{IE})}{4N_C \times (5 - 2P_{AS})} \tag{6.7}$$

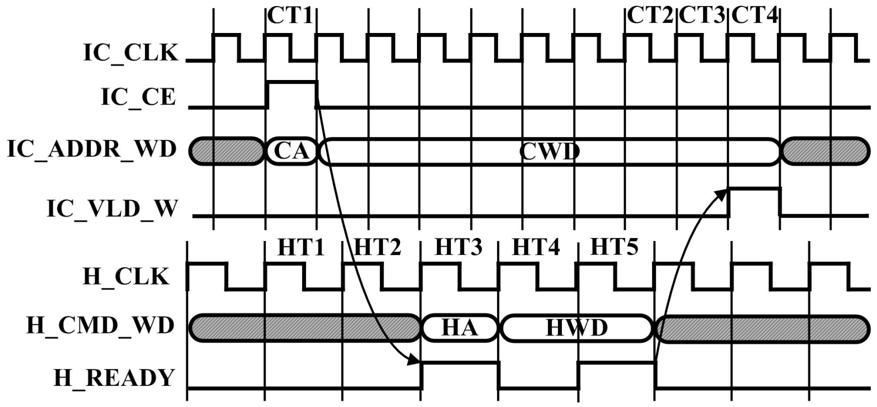
To encrypt/decrypt 20 AES states, 132 cycles are required for ID bus using the specific state transfer mode, and 172 cycles are needed on the AXI bus side.

6.3.2 Integration Configuration

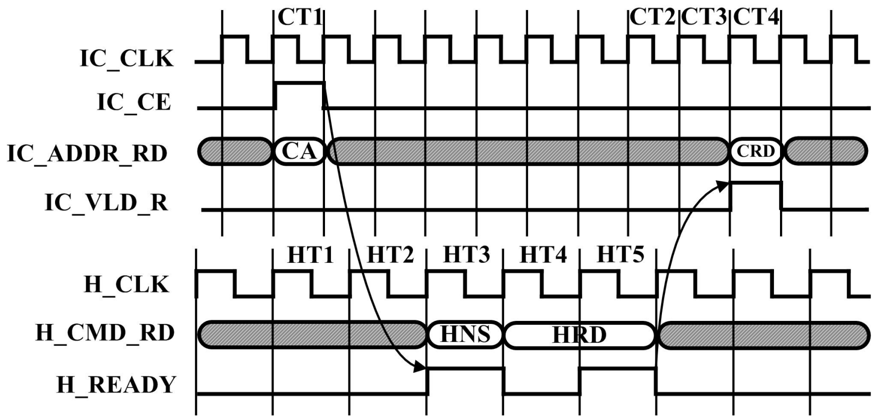
The main operations of SoC integration are: 1) bus protocol conversion, and 2) signals crossing between different time domains. In this work, we name the conversion module requiring both the first and the second operations as bus wrapper. In contrast, the conversion module requiring only the second operation is named as bus bridge.

In order to avoid the input jitter sampling and ensure that the receiving interface does not enter a metastable stage, two methods are used in the structure generation algorithm: clock-domain handshaking and asynchronous data FIFOs addition. In general, asynchronous FIFO is essentially a memory queue requiring the highest area and consuming much power, but it is high efficient to preprocess data transfers and release bus accesses immediately. The other method, asynchronous signal handshaking, is cost-effective but high-latency on data bus.

We consider the IC2H wrapper first. Since IC bus is created as a low-cost and low-speed control bus, the handshaking approach is well suited for the IC2H wrapper to synchronize signals between IC and AHB clock domains. As an example shown in Figure 6.2, a pair of handshaking signals, the IC command enable signal (C_CE) and IC data valid signal (C_VLD_W/C_VLD_R), is used to synchronize the signals crossing between IC and AHB bus domains. Additionally, the C_CE signal should be delayed by two AHB cycles before it can be used by the AHB IP. Likewise, the C_VLD_W/C_VLD_R signal is active after two IC cycles delay of the AHB data ready (H_READY).



(a) Write Operation



(b) Read Operation

Figure 6.2: IC2H Wrapper Structure.

In what follows, we focus on analyzing the X2ID wrapper configuration shown in Algorithm 1. Two asynchronous FIFOs, one for read and one for write, are used to improve the transfer efficiency of the full-duplex ID bus. The FIFO size is mainly considered as the most critical factor responsible for bottleneck in the wrapper design of the high speed data transfer.

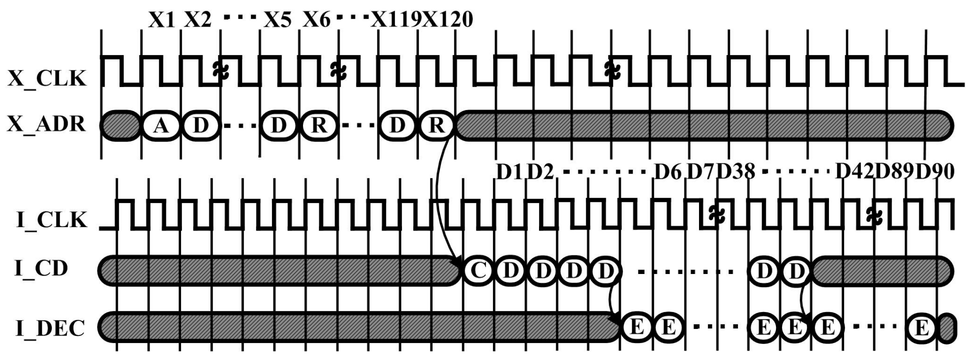
Here, we explain the conversion between AXI3 and ID bus protocols as a case study. The FIFO sizes are determined by several factors: the bus sizes, the maximum data length of specific IP applications, and the coefficients of write and read FIFOs denoted as WR_C and RD_C, respectively. Hence, the minimum write data FIFO size, denoted as X2ID_WF_SIZE, can be calculated as

$$X2ID\_WF\_SIZE = WR\_C \times N \times (\frac{XS}{IDS}). \tag{6.8}$$

where WR_C is greater than or equal to 2. Although one $N \times XS/IDS$ of memory is sufficient, allotting twice this amount allows simultaneous operations. That is, when the AXI is writing one packet, the ID bus can read the previous packet. If the XOF/IDOF ratio is less than or equal to R, or the AXI latency is high, allot more space of memory to hold multiple write data packets to improve the transfer efficiency. Considering the tradeoff between speed and area cost, the maximum X2ID_WF_SIZE is $6 \times N \times (XS/IDS)$. Similarly, the minimum read FIFO size, denoted as X2ID_RF_SIZE, is calculated as

$$X2ID\_RF\_SIZE = RD\_C \times N \times (\frac{XS}{IDS}). \tag{6.9}$$

where RD_C is greater than or equal to 2. Allotting twice of the maximum amount improves the read efficiency. When the AXI is reading one packet, the ID bus can write the next packet. If the AXI clock is fast, more memory size is allotted to hold more read packets.



X1 X2 · · · X5 X6 · · · X119 X120
X_CLK
X_ADR
A
D
D
R
D
R
D1 D2 · · · · · · · D6 D7 D38 · · · · · · D42 D89 D90
I_CLK
I_CD
C
D
I_DEC
E

Figure 6.3: X2ID Write/Decryption Timing Diagrams.

As an example, the timing diagram of an IP write operation or AES decryption is shown in Figure 6.3. Except for the arbitration process, AXI3 IP costs 120 AXI3 cycles to push 10 ciphertext states into the write data FIFO. Each AXI3 transfer involves three phases: address (A), data (D), and response (R). On the ID bus side, the ciphertext states should be popped out and decrypted, and then the plaintext states are written into memory. The data transferred on ID bus are consecutive, and also paralleled with the AES decryption process. Apart from the arbitration and command stage (C), it consumes 90 ID bus cycles for the cipher test, including 40 cycles for data transfer on bus (D) and 86 cycles for AES decryption (E) with 36 cycles overlapped.

6.4 Case Studies and Experimental Results

In our work, the IBUS structure, together with bus wrappers, can be auto-generated according to users' configuration. This solution enables industrial standard IPs to be seamlessly integrated into IBUS architecture. In order to evaluate the IBUS system performance, we also build up a UVM environment, interconnecting with several VIPs and design modules. Finally, the FPGA back-end flow is performed to estimate area costs and power consumption.

6.4.1 SoC Environment

Figure 6.4 shows the UVM test bench, involving several bus performance models and mixed-signal open verification components (OVCs), and also merging the traditional verification methods, such as ABV, CDV, and the CRV.

Three AHB- and AXI-based VIPs, the USB2.0 Host controller, graphic module, and Wi-Fi Mac, are integrated into the IBUS structure to estimate three transfer modes. Assume that the operating frequencies of the VIPs and IBUS SoC are asynchronous. IC2H and X2ID wrappers are thus required for IP integration.

In this figure, the generated modules are highlighted using gray background, and the verification components are represented by dark background. To evaluate the IP integration performance, AXI based DUTs, XDAM and XDAM-B, and IBUS based DUTs, IDAM and IDAM-W, are implemented as case studies, where DAM is the top module of DMA interconnected with AES engines and memory controller, and the suffixes "-B" and "-W", respectively, indicate the bridges used in AXI DUTs and wrappers used in IBUS DUTs. All the DUTs are represented by white background in this figure.

6.4.2 IBUS Structure Configuration

Following the users configuration, the IBUS structure with several wrappers can be generated. As an example shown in the second column of Table 6.1, a high-speed USB2.0 Host Controller, supporting 90MHz AXI interface and a typical control packet size (64 Words), is analyzed. Using the algorithm illustrated in Algorithm 1, the write and read asynchronous FIFOs are configured as 256 and 128 words, respectively, to achieve high performance for the IBUS SoC.

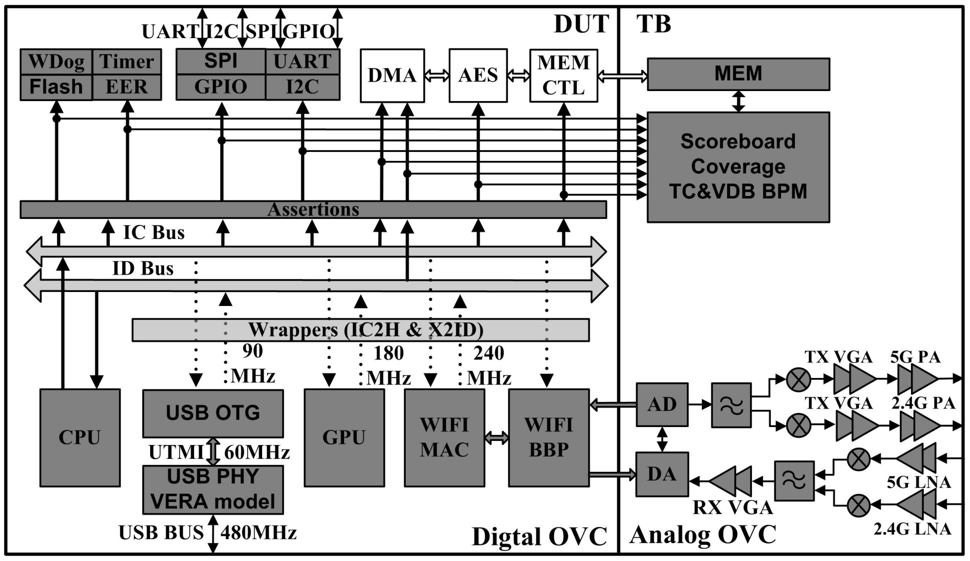


Figure 6.4: SoC Environment.

Table 6.1: Wrapper Configuration.

Configuration		USB 2.0 IP	Graphic IP	WiFi IP
	XOF (MHz)	90	180	240
Features	IDOF(MHz)	180	180	180
	Packet Size	64 Words	16×16 pixels	10 States
WFIFO (Word)		4×64=256	2×64=128	2×40=80
RFIFO (Word)		2×64=128	4×64=256	6×40=240

Likewise, the graphic IP typically processing a 16×16 pixels is considered in the third column. Its operational frequency is 180MHz, the write and read FIFO sizes thus are calculated as 128 and 256 words, respectively.

In the fourth column, a WiFi MAC providing AES encryption/decryption and supporting 240MHz bus frequency is integrated into the IBUS SoC. Assume that the maximum pipeline number is 10 for our AES core, the configured FIFO sizes are thus 80 and 240 words, respectively, for write and read operations.

As an example, a typical write test in our study is shown in Figure 6.5, that is, all the three AXI3 VIPs write 80 words to the memory. The system command is split into two bus requests by software, one 64-word and one 16-word control

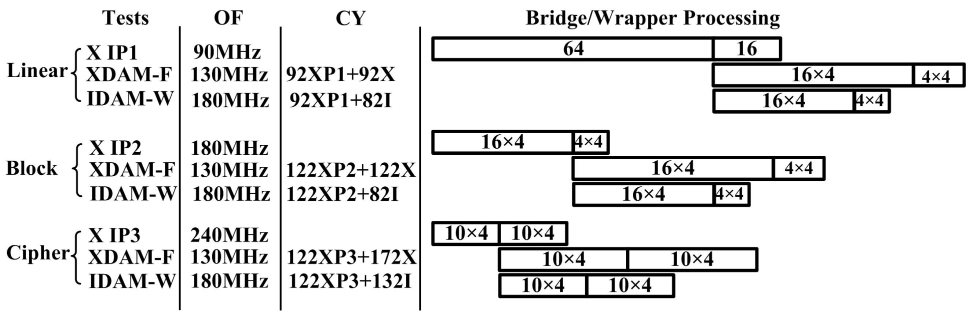


Figure 6.5: Test Vectors.

packets for USB 2.0 Host, one 16 × 16 pixel matrix and one 4 × 16 pixel matrix for the graphic VIP, and two 10-state transfers for WiFi Mac. Assume that all the transfers are fully back-to-back, and all the grant and data valid signals are asserted immediately. Since the maximum load per AXI3 transfer can reach 16 beats, the total cycles needed by AXI3 IP are 92, 122, and 172, respectively, for the linear, block, and cipher tests. On the other side, the IBUS costs 82, 82, and 132 cycles, respectively, for the three test modes, and the XDAM consumes 92, 122, and 122 cycles.

6.5 IBUS Performance Analysis

6.5.1 Implementation Flow

In Table **??**, several performance parameters are summarized. First, the latency cycles, and the time consumption formulized as the product of latency cycles and the maximum operational frequency, and the valid data bandwidth defined as the valid data can be transferred per second, are measured in the front-end simulation and shown in the fifth to seventh columns. In addition, the simulation VCD files with the exact switching activities of IOs, signals, and logic are also collected.

Then, we obtain the fully placed and routed NCD files and physical constraint PCF files by synthesizing all the RTL designs, using Xilinx ISE 14.7 with Virtex6xc6vlx550t-2ff1760 as the target device. The second and third columns show the synthesis results, including occupied slices and MOF.

In what follows, inputting all the NCD and PCF files, as well as the VCD files into XPower Analyzer, the dynamic power statistics are obtained in the eighth column. Since static power is mostly determined at the circuit level and almost a constant even using different tests, our work thus concentrates on analyzing dynamic power. In the ninth column, dynamic energy is further calculated as the integral of dynamic power, or the product of dynamic power and time consumption when dynamic power is obtained as the average dynamic power.

Finally, slice efficiency is computed in terms of valid data number that can be transferred per second per slice, which is shown in the tenth column. In the eleventh column, dynamic energy efficiency is defined as the valid data number that can be transferred per second per watt, or valid data number that can be transferred per joule.

Algorithm 1 X2ID Wrapper Configuration.

```
1: Let R = bus latency ratio of ID bus divided by AXI
2: Let N = the maximum data transfer length of specific IP applications
3: Let LN_EN/BL_EN/CP_EN = linear/block/AES cipher mode enable
4: Let WR_C/RD_C = coefficient of write/read data FIFO size
5: Let XS/IDS = AXI bus size/ID bus size
6: if CP_EN then
7:     $R = \frac{2+2N_C\times(27-25P_{IE})}{4N_C\times(5-2P_{AS})}$; $N = 4 \times N_C$;
8: end if
9: if BL_EN then
10:    $R = \frac{(2-2P_{IB})\times ceil(\frac{N_H}{DH})\times ceil(\frac{N_W}{DW})+N_H\times N_W}{(4-2P_{XB})\times N_H\times ceil(\frac{N_W}{XL})+N_H\times N_W}$; $N = N_H \times N_W$;
11: end if
12: if LN_EN then
13:    $R = \frac{(2-2P_{IL})\times ceil(\frac{N_L}{IL})+N_L}{(4-2P_{XL})\times ceil(\frac{N_L}{XL})+N_L}$; $N = N_L$
14: end if
15: if $XOF/IDOF <= 1/2R$ then
16:    $WD\_C = 6$; $RD\_C = 2$;
17: end if
18: if $1/2R < XOF/IDOF <= R$ then
19:    $WD\_C = 4$; $RD\_C = 2$;
20: end if
21: if $R < XOF/IDOF <= 2R$ then
22:    $WD\_C = 2$; $RD\_C = 4$;
23: end if
24: if $XOF/IDOF > 2R$ then
25:    $WD\_C = 2$; $RD\_C = 6$;
26: end if
27: $X2ID\_WF\_SIZE = WR\_C \times N \times \frac{XS}{IDS}$;
28: $X2ID\_RF\_SIZE = RD\_C \times N \times \frac{XS}{IDS}$;
```

Table 6.2: Performance Evaluation

DUTs	SC	MOF (MHz)	Tests	CY (ns)	TC (ns)	VDB (GBps)	DP (mW)	DE (uJ)	SE (KBps/S)	DEE (GBps/J)
			XL	92.00	707.69	0.90	612.00	0.43	34.64	1.48
XDAM	26106	133.010	XB	122.00	938.46	0.68	752.00	0.71	26.12	0.91
			XE	172.00	1323.08	0.48	771.00	1.02	18.53	0.63
			IL	82.00	455.56	1.40	574.00	0.26	56.60	2.45
IDAM	24822	183.636	IB	82.00	455.56	1.40	695.00	0.32	56.60	2.02
			IE	132.00	733.33	0.87	716.00	0.53	35.16	1.22
			XL	$92 \times XP + 92 \times ID$	1729.91	0.37	667.00	1.15	13.61	0.55
XDAM-B	27192	133.010	XB	$122 \times XP + 122 \times ID$	1616.24	0.40	792.00	1.28	14.56	0.50
			XE	$122 \times XP + 172 \times ID$	1831.41	0.35	802.00	1.47	12.85	0.44
			IL	$92 \times XP + 82 \times ID$	1477.78	0.43	632.00	0.93	15.82	0.69
IDAM-W	27375	183.636	IB	$122 \times XP + 82 \times ID$	1133.33	0.56	755.00	0.86	20.63	0.75
			IE	$122 \times XP + 132 \times ID$	1241.67	0.52	783.00	0.97	18.83	0.66

6.5.2 Performance Analysis

In the second column of Table **??**, it can be observed that the IBUS architecture reduces the slice cost due to the compact bus architecture, compared with AXI3 based designs. However, since much more slices are costed for wrapper generation when integrating the third-party IPs, the slice count of IDAM-W is as many as that of XDAM-B. In addition, IBUS based implementations shorten the critical path, turning out to achieve higher MOF than the AXI based designs.

Furthermore, it is clear to analyze the other performance metrics between IBUS and AXI architectures shown in Figure 6.6(a) and Figure 6.6(b). In Figure 6.6(a), it can be observed that IBUS outperforms AXI3, especially for the cipher tests. The time consumption and dynamic power, respectively, consumed by the IBUS cipher test are reduced to 55.43% and 92.87%, compared with the AXI3 vectors, due to the specific AES state transfer mode. In sum, the slice efficiency and dynamic energy efficiency of IBUS are 1.90 times and 1.94 times of those of AXI3.

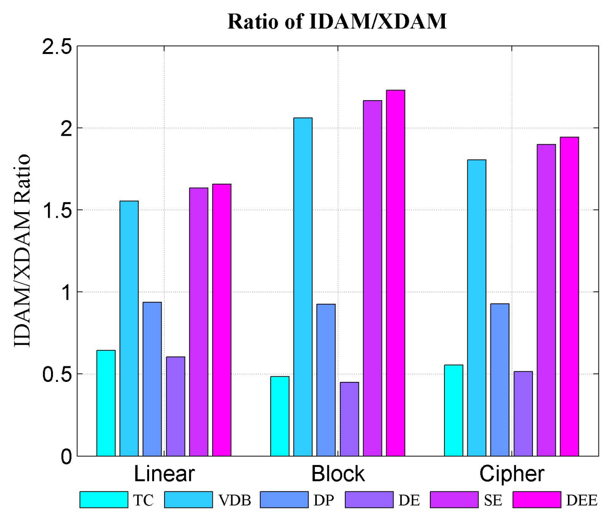
In this chapter, we focus on the performance comparison of IP integration shown in Figure 6.6(b). First, IDAM-W consumes less time than XDAM-B for all the linear, block, and AES state tests, due to the high-efficient bus structure. The valid data bandwidth achieved by the IBUS implementation are around 1.17, 1.43, and 1.48 times, respectively, compared with the AXI3 design. Moreover, although the dynamic power consumption are close to each other, the dynamic energy consumption of IDAM-W is reduced to 80.94%, 66.85%, and 66.19%, respectively, compared with XDAM-B for the three tests. In conclusion, the valid data number can be transferred per second per slice cost of IBUS based integration is 1.47 times compare with AXI3 based implementation in the cipher mode. Furthermore, the valid data number can be processed per second per watt consumption of IDAM-W is close to 1.51 times compared with XDAM-B.

It is obvious that much more overhead costs are needed for integrating industrial standard IPs, not only for the third-party IPs integration, but also for the same bus protocol IPs with different operating frequency. In Figure 6.6(c), the additional area cost, latency, and dynamic power consumption of IBUS and AXI3 SoC integration are analyzed. First, more area is added to the SoC structure as wrapper or bridge design logic. Second, the latency and dynamic power consumed by additional bridge designs are less than those consumed by wrappers, due to the compatibility of standard AXI3 IPs. For instance, the time consumption and dynamic power consumed by XDAM-B is 1.38 and 1.04 times, respectively, of those of the XDAM in the cipher mode. On the contrary, much more time consumption and dynamic power are required for AXI3 IPs integration with IBUS structure. In the cipher test, the time consumption and dynamic power costs of IDAM-W is 1.69 and 1.09 times, respectively, compared with IDAM.

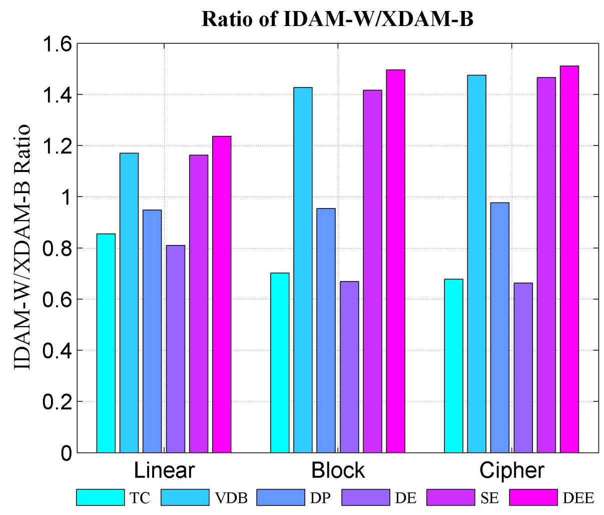
6.6 Summary

In this chapter, we propose a configurable and synthesizable IBUS architecture for industrial standard IP integration, tailored for tiny scale, low power, and high performance AES-encrypted IoT embedded chips. Unlike the previous research, the proposed approach aims at automatically creating the system structures, based on the real circuit design flow and performance evaluation methodology.

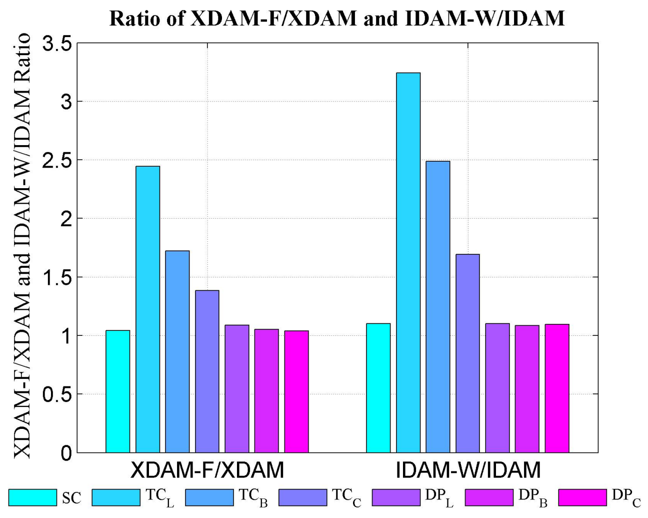
We also apply the auto-generation algorithm to create an IBUS structure as a case study. Comparing with AXI3 based designs, it achieves higher valid bandwidth and consumes less dynamic energy with less slice usage. In the future, we plan to optimize the wrapper factory to satisfy more industrial buses for guaranteeing quality and quantity of services to IBUS-based integration.



(a) IDAM/XDAM



(b) IDAM-W/XDAM-B



(c) XDAM-F/XDAM and IDAM-W/IDAM

Figure 6.6: Performance Comparison.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

In this chapter, we summarize our contributions presented in this dissertation. We then discuss the possible directions for our future research work.

7.1 Summary

IoT is expected to grow to include more than 50 billion devices by 2020. In the growing chip market, the things or devices have unique requirements with profound effects on the way SoCs for IoT ecosystems are developed. The unique requirements are:

- System cost: For mass adoption, devices must be cost-effective to deploy and require low-cost integrated circuits.
- Ultra low power consumption: Both dynamic and leakage power consumption should be reduced. Devices tend to be either battery powered, utilizing energy harvesting, or the power consumption budget is small, because connecting to the Internet is not the main function of the device.
- Security: The potential for someone gaining control of anything connected to the Internet, particularly when it comes to the areas like healthcare or critical national infrastructure, is a major concern.
- User configurability/personalization: Integrated circuits often require personalization to perform their function, including a broad set of IPs, microprocessor, wireless communication, graphic processing unit, memory controllers, external interfaces, and several SoC peripherals.

Each requirement helps guide design decisions related to cryptosystems, power and energy efficiency, verification methodology, mixed-signal design, and design automation. In this dissertation, we present our research work that has been done on each requirement and describe their impact on the designers' choices.

First of all, we proposed a low-cost and low-power bus architecture, termed IBUS, for high-performance and high-efficiency IoT embedded chips. The compact bus protocol provides an excellent balance of cost and energy-efficiency. It is optimized with two novel and high-efficient transfer modes, block and AES state modes, and also backwards supports the conventional linear transfer mode. Experimental results show that IBUS costs the least IOs and gate count than AMBA AHB and AXI. Remarkably in the block transfer mode, IBUS latency is close to 30% of AHB and 63% of AXI, and the dynamic energy consumption of IBUS based designs is close to a half compared with AHB and AXI based implementations.

In order to evaluate the architecture performance automatically and accurately, an evaluation methodology was further presented. Not only the traditional performance metrics, such as slice count, the maximum operational frequency, transfer latency, power consumption, and bandwidth, but also several new performance parameters, including valid bandwidth, dynamic energy consumption, wire efficiency, slice efficiency, and dynamic energy efficiency, can be measured efficiently. Multiple case studies demonstrate that the evaluation methodology can be effectively used to estimate chip performance.

Since security and privacy become one of the top tier concerns of IoT embedded chips, we further presented an advanced IBUS structure to improve the system performance and provide an architectural support for the block-based AES algorithm. Furthermore, the 32-, 64-, and 128-bit bus-based designs are implemented to compare and analyze the system performance using linear, block, and state tests. Based on the available resource, structural efficiency demands, and circuit performance requirements, one can choose the proposed implementations based on different bus sizes to fulfill the constraints of different applications. FPGA comparison results show that IBUS based design costs less in terms of hardware resource and achieves up to $\times 1.3$ times throughput of the AXI based implementation, and the dynamic power consumption of IBUS test is 71.3% compared with the AXI test.

The increasing scale and complexity of SoCs demand flexible bus structures and verification methodologies. We thus proposed a highly configurable and synthesizable on-chip architecture for IoT SoCs that can seamlessly interconnect with industrial standard IPs, delivering a broad-range of applications, including microprocessors, on-chip memory, security, wireless communication, and so on. To meet the tight time-to-market constraints and to effectively handle the design complexity, not only the cost-effective, low-energy, and AES-embedded bus structure can be designed automatically, but also a mixed-signal verification environment, with several VIPs and performance models, can be auto generated.

7.2 Future Work

Aiming at IoT circuits, the IBUS architecture is optimized as a reduced interface complexity, minimal power consumption, and high security bus structure. Three novel transfer modes, linear, block, and AES state modes, are presented in the IBUS protocol, and a configurable IP integration structure is further proposed. Essentially, our proposed IBUS structure is a single processor/master and multiple clients on-chip bus.

In the future, we can extend our research on a variety of chip design topics, such as multi-core bus architecture, IP/VIP integration, design automation, and verification methodology, that are top concerns particularly for new bus protocols.

Multi-Core Architectures has been the new generation for processors of today. With Moore's law constantly growing and the frequency reaching a maximum for single CPU chip designs, manufacturers have transformed their designs to multi-cores, where each core is much smaller and has less functionality then a CPU. These cores work together in order to process a job in an efecient manner. The use of multiple cores allows for the frequency of the processor to be reduced, thus reducing the temperature of the system. With multi-cores, instructions are allowed to run on individual cores simultaneously, which increases the amount of parallelism.

In addition, with the increasing SoC hardware and software complexity, developers need more from their IP/VIP providers to help meet their project schedules. Traditional IP/VIP blocks alone are no longer adequate to address the growing SoC design and integration challenges. Today, IP/VIP vendors should do more preparation of the designs to make it ready to use than in the past, including a much more complete view of what the SoC needs for the IP/VIP, such as a configurable reference design, auto generated behaviour models, and flexible verification environment.

Multi-Core Bus Architecture: As a single-processor and multi-client bus structure, IBUS reduces resource utilization and energy consumption, and limits the complexity of circuits. The IBUS protocol is thus very desirable for small scale embedded chips with requirements of a low-cost interface and high energy efficiency.

If single processor can supply sufcient computational power, then the ease with which they can be programmed pulls system designers toward uniprocessor IoT SoCs. However, thanks to Moore's Law advances, we believe that new applications that will require the development of new multi-core IoT SoCs will emerge. FPGA based complex algorithms, such as high-definition video processing and pattern recognition for drones and unmanned aerial systems, are examples of an emerging field that can use essentially unlimited amounts of computational power but must also meet real-time, low-power, and low-cost requirements.

In addition, applications like multimedia and high-speed data communication not only require high levels of performance, but also require implementations to meet strict quantitative goals. The term "high-performance computing" is traditionally used to describe applications such as scientic computing that requires large volumes of computation but do not set out strict goals about how long those computations should take. Embedded computing, in contrast, implies realtime performance. In real-time systems, if the computation is not done by a certain deadline, the system fails. If the computation is done early, the system may not benet.

Actually, multi-core system has been widely used in networking, communications, signal processing, and multimedia among other applications today. The design methods and tools that have been developed for multi-core SoCs will continue to be useful for these next-generation systems.

Therefore, our proposed IBUS will be extended to a multi-core architecture in the future. As an example shown in 7.1, three cores, one main processor and two co-processors, are integrated in an extended IBUS architecture. All of them can access the IC bus through IC master interfaces. The IC arbiter ensures that only one processor has access to the IC bus at any one time. Additionally, all the processors can also send requests to ID bus through the ID master interfaces.

Computer Aided Design Automation: In our work, a new IBUS structure was proposed, in order to reduce the interconnect wires and logic gates, so as to resolve routing congestion and timing closure issues at the back-end place-and-route stage for low-cost IoT chips, resulting in shorter development cycle time, faster operational frequencies, smaller chip area, and less power consumption.

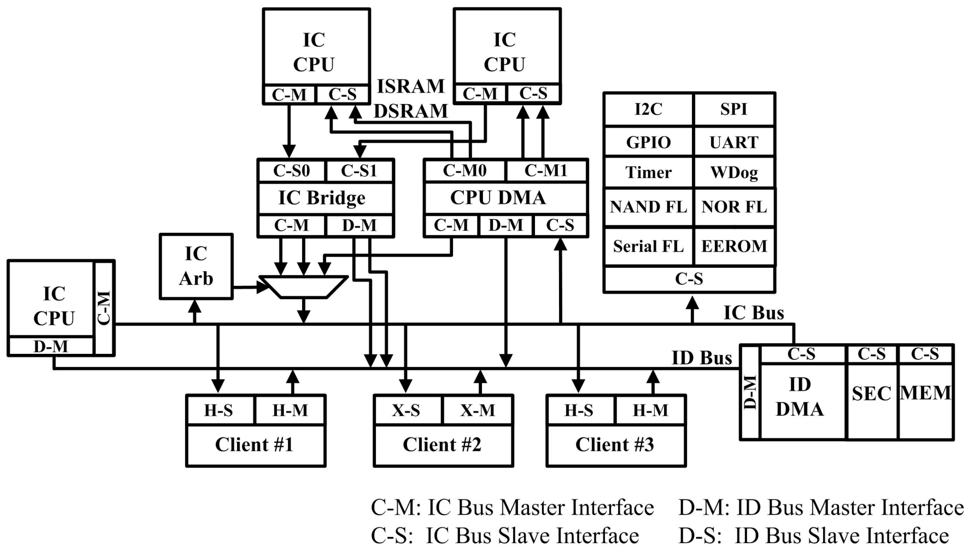
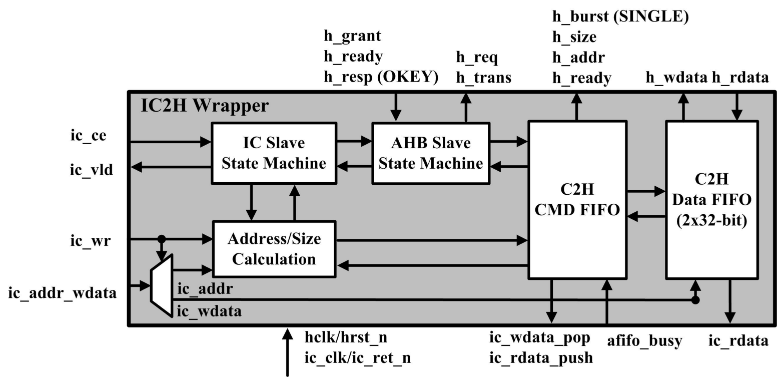


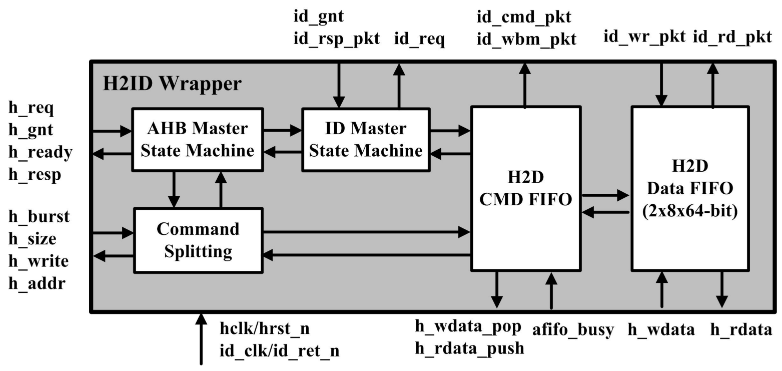
Figure 7.1: IBUS Multi-Core Structure.

In the future, a variety of the third party IP blocks, such as processors, DMA, memory controller, and other peripherals, will be selected from libraries, and can be interconnected using right kinds of protocol matching wrappers. As an example shown in Figure 7.3, an IBUS based SoC utilizes an automated wrapper-based bus structure generation process and tool set, in order to expand the potential of the multi-core IBUS structure introduced in Figure 7.1. More specifically, Figure 7.2(a), Figure 7.2(b), and Figure 7.2(c) show the structures of IC2H, H2ID, and X2ID wrappers, respectively.

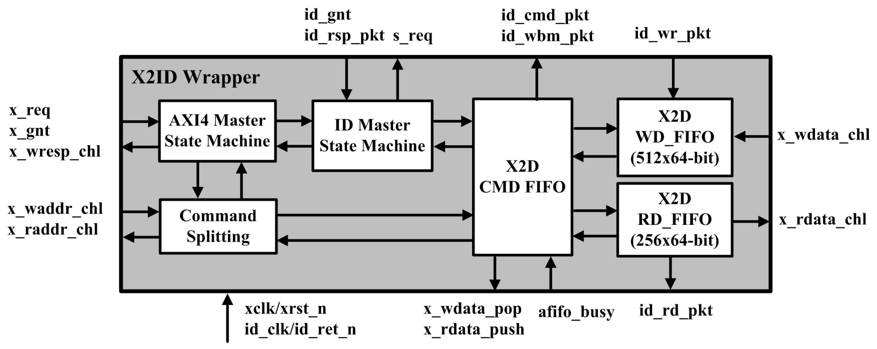
In addition, designers require solutions that ease IP configuration and integration into the overall SoC as well as accelerate their software development effort. Therefore, the aim of design automation is that deliver a solid package with comprehensive solutions, including RTL code, verification environment, environment control scripts, place and route guidelines, and supportive documentations, to lower the development cost, reduce integration risk, and meet their market schedules. As an example shown in Figure 7.4, not only the bus structure with bus wrappers and



(a) IC2H Wrapper

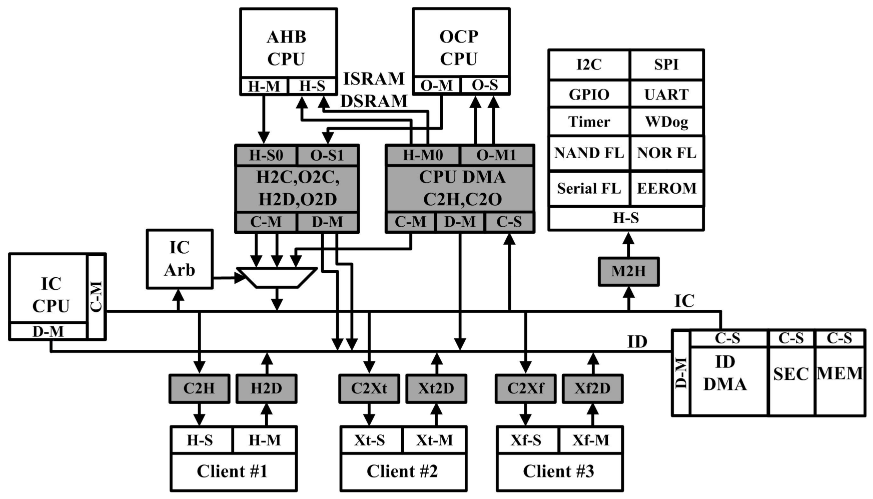


(b) H2ID Wrapper



(c) X2ID Wrapper

Figure 7.2: Wrapper Structures.



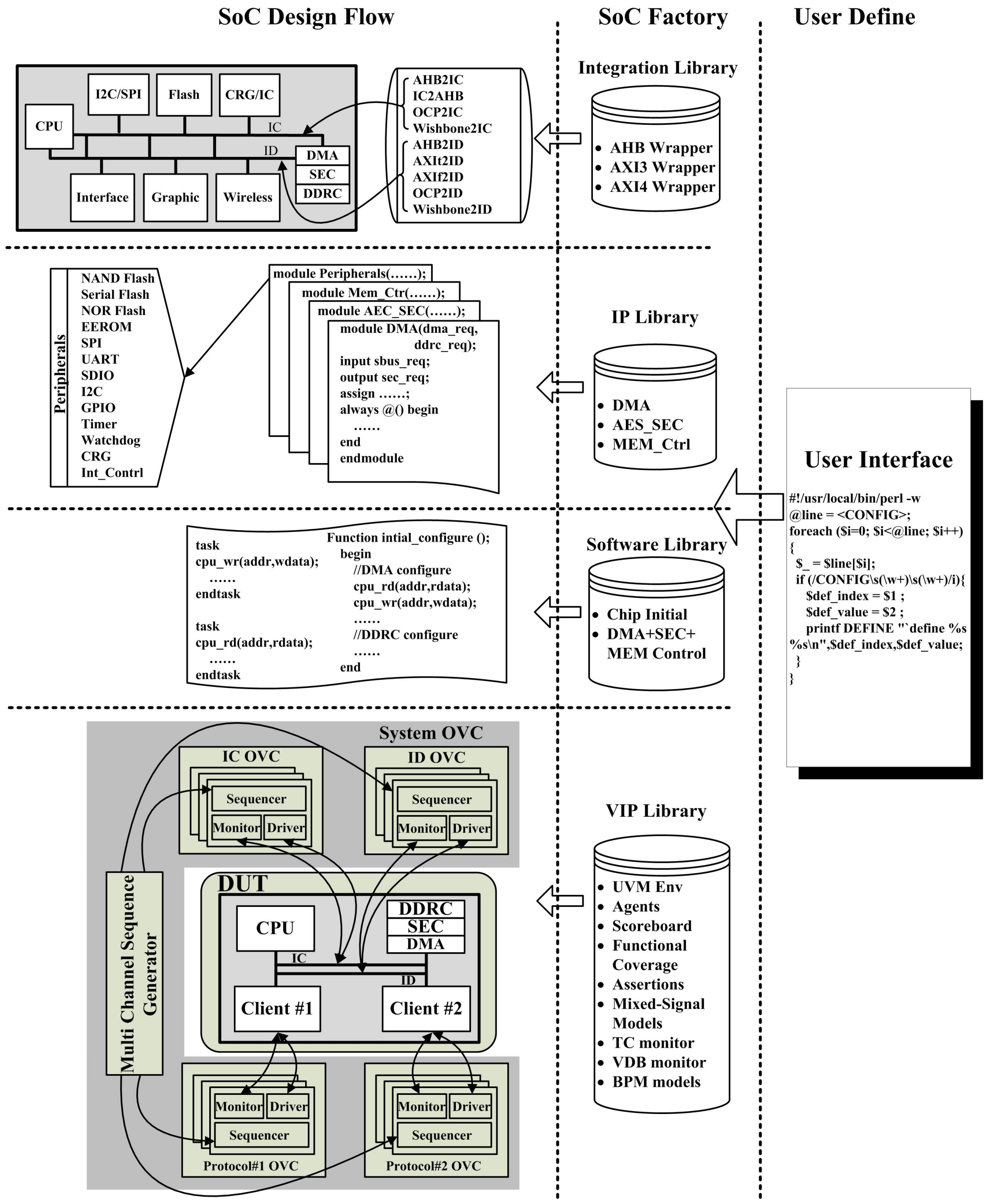
C-M: IC Master Interface	Xt-M: AXI3 Master Interface	C2H: IC2AHB Wrapper	H2C: AHB2IC Wrapper
C-S: IC Slave Interface	Xt-S: AXI3 Slave Interface	H2D: AHB2ID Wrapper	O2C: OCP2IC Wrapper
D-M: ID Master Interface	Xf-M: AXI4 Master Interface	C2Xt: IC2AXI3 Wrapper	O2D: OCP2ID Wrapper
D-S: ID Slave Interface	Xf-S: AXI4 Slave Interface	Xt2D: AXI32ID Wrapper	C2O: IC2OCP Wrapper
H-M: AHB Master Interface	O-M: OCP Master Interface	C2Xf: IC2AXI4 Wrapper	
H-S: AHB Slave Interface	O-S: OCP Slave Interface	Xf2D AXI42ID Wrapper	

Figure 7.3: Wrapper-Based IBUS Structure.

synthesizable IPs, but also control tasks, VIP and BPM based test bench can be configured and auto generated from libraries.

So far, our discussions are based on the physical embedded chips. In the future work, we can also consider IoT as an ecosystem, where it is not only a physical circuit to transfer data, but also a platform connected with several areas, such as biomedical engineering, cyber security, and even big data, cloud computing, and machine learning.

Biomedical Engineering: Things, in the IoT sense, can refer to a wide variety of biomedical devices, such as heart rate monitors, collectors for vital signs and health information, smart contact lens, and ingestible smart pills. These tiny biochips will have particular implications in the area of biomedical engineering, due to the increasing availability and use of health and fitness sensors, ultra-low power microcontrollers, and application-specific systems. Our proposed IBUS architecture is



```
module Peripherals(……);
  module Mem_Ctr(……);
    module AEC_SEC(……);
      module DMA(dma_req,
                 ddrc_req);
      input sbus_req;
      output sec_req;
      assign ……;
      always @() begin
        ……
      end
      endmodule
```

```
task
cpu_wr(addr,wdata);
   ……
endtask

task
cpu_rd(addr,rdata);
   ……
endtask
```

```
Function intial_configure ();
  begin
    //DMA configure
    cpu_rd(addr,rdata);
    cpu_wr(addr,wdata);
    ……
    //DDRC configure
    ……
  end
```

```
#!/usr/local/bin/perl -w
@line = <CONFIG>;
foreach ($i=0; $i<@line; $i++)
{
  $_ = $line[$i];
  if (/CONFIG\s(\w+)\s(\w+)/i){
     $def_index = $1 ;
     $def_value = $2 ;
     printf DEFINE "`define %s
%s\n",$def_index,$def_value;
  }
}
```

Figure 7.4: Design Automation System.

well-suited for the tiny biochips, with requirements of low-cost structure and high energy efficiency.

As an example, Figure 7.5 shows an application-specific IBUS based design structure, which is used to produce the AES-encrypted exercise data, such as maximum aerobic function (MAF) exercise levels and calorie consumption, to extend the workout based applications [120, 121]. Figure 7.6 shows the system environment, involving not only the DUT with white-background, but also the bus performance models, mixed-signal open verification components, and the traditional verification groups, such as assertion-based verification, coverage-driven verification, and the constrained random tests.

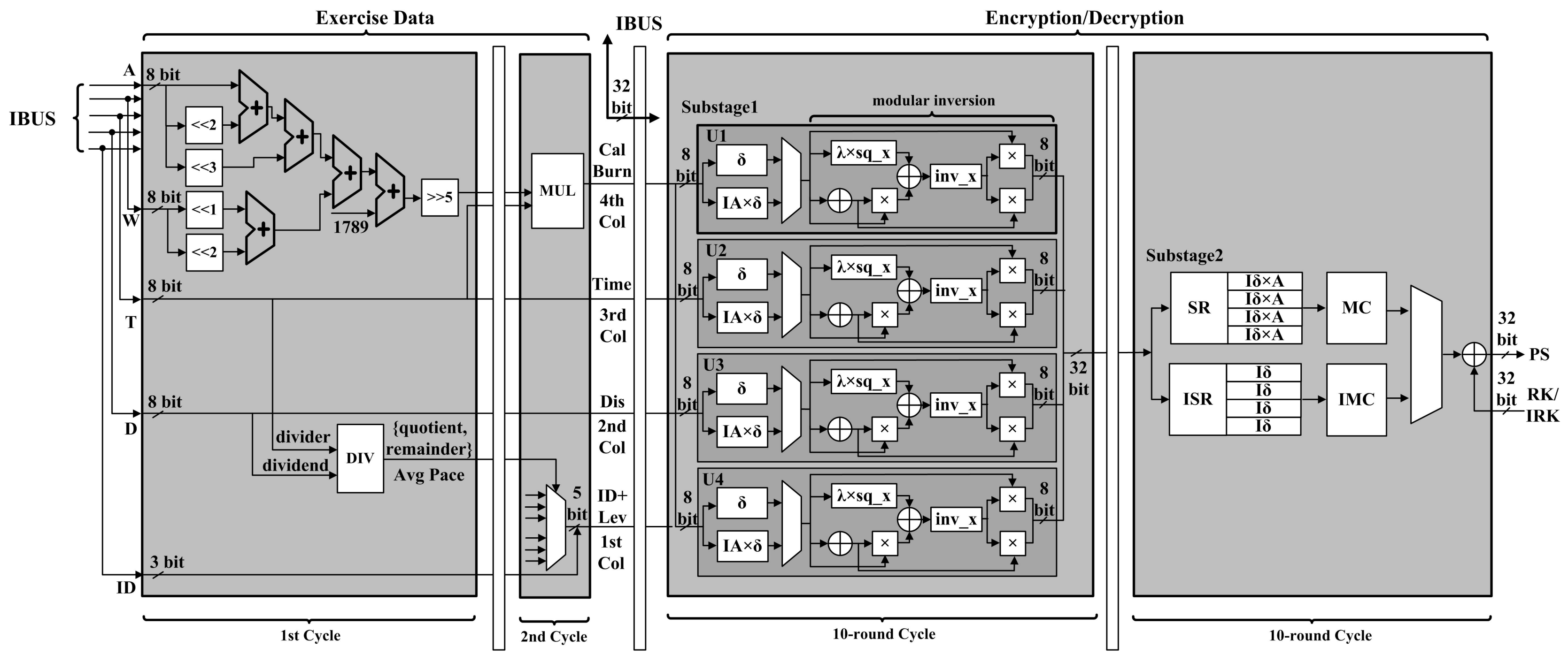


Figure 7.5: A Bio-Feedback Circuit Design Using IBUS Protocol.

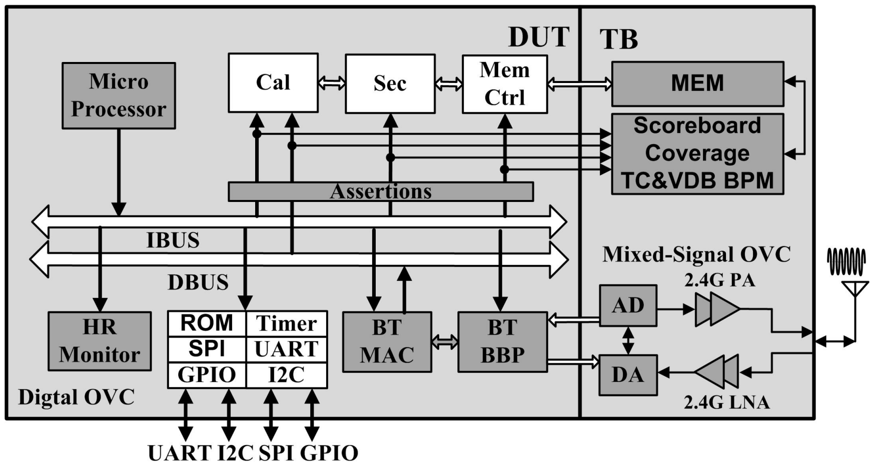


Figure 7.6: A UVM Based Verification Environment.

Cyber Security And Privacy: There is no doubt that security and privacy is one of the most critical issues for the IoT ecosystem, particularly when it comes to the areas like healthcare or critical national infrastructure. However, the limited resource on an tiny IoT chip highlights that current bus architectures are not capable of keeping up with the computational demands of security processing, and the power constraint emphasizes that the energy consumption overhead of supporting security on power-limited embedded systems is very high. Under this context, our proposed IBUS protocol can be expanded to optimize the security algorithms to leverage limited resources for IoT embedded chips and overhead costs for security mechanisms.

Big Data and Cloud Computing: IoT and big data are clearly intimately connected: billions of Internet-connected things will generate massive amount of data. Since the costs of on-chip implementations are much lower than that of the high-level operations, hardware-based data optimizations will dramatically improve the quality and reduce the quantity of raw data from IoT devices, reduce the system overhead, and improve the ecosystem performance.

Machine Learning of IoT Devices: There is a great vision that all “things” can be easily controlled and monitored, can be identified automatically by other things, can communicate with each other, and can even make decisions by themselves. The general premise is the machine learning algorithms: review and analyze the data have been collected to find patterns that can be learned from, so that better decisions can be made by smart devices.

In sum, our research can be connected to a broad area that is of practical significance and has immediate relevance and impact in industry. We are looking forward to continuing and making an impact in the emerging IoT area.

BIBLIOGRAPHY

[1] Amba specification. Axis, Sunnyvale, CA, USA, 1999.

[2] Coreconnect bus architecture. IBM, Yorktown Heights, New York, NY, USA, 1999.

[3] Wireless lan medium access control (mac) and physical layer (phy) specification. IEEE Standard 802.11-1999, 1999.

[4] Open core protocol specification. OCP Int. Partnership, Beaverton, OR, USA, 2001.

[5] Amba axi protocol specification. Axis, Sunnyvale, CA, USA, 2003.

[6] Wishbone bus. Silicore Corp., Corcoran, MN, USA, 2003.

[7] Stbus interconnect. STMicroelectronics, Geneva, Switzerland, 2004.

[8] In *SystemVerilog Unified Hardware Design, Specification, and Verification Language*, pages 1 – 1294. IEC 62530 Edition 2.0 2011-05 IEEE Std 1800, 2011.

[9] In *IEEE Standard for Standard SystemC Language Reference Manual*, pages 1 – 638. IEEE Std 1666-2011 (Revision of IEEE Std 1666-2005), 2012.

[10] In *IEEE Standard for Design and Verification of Low-Power Integrated Circuits*, pages 1 – 348. IEEE Std 1801-2013 (Revision of IEEE Std 1801-2009), 2013.

[11] Connections counter: The internet of everything in motion. *CCS, [Online]*, 2013.

[12] Si2 common power format specification. Austin, TX, USA, Feb. 2011.

[13] Ieee standard for system verilog-unified hardware design, specification, and verification language. in: proceedings of the IEEE Std 1800-2012, Design Automation Standards Committee, Feb. 21, 2013.

[14] Xilinx, virtex-6 family overview. Jan. 2012.

[15] Mpeg-2 standards. Part1 Systems, ISO/IEC, GE, Switzerland, Jun2 2010.

[16] Uvm 1.1 reference manual. Accellera, Tualatin, OR, USA, June 2011.

[17] 1801-2009 ieee standard for design and verification of low power integrated circuits. pages 1–218, New York, USA, Mar. 2009.

[18] Uvm 1.1 user guide. Accellera, Tualatin, OR, USA, May 2012.

[19] Fips pub 197, advanced encryption standard (aes). National Institute of Standards and Technology, U.S. Department of Commerce, Nov. 2001.

[20] Ieee standard verilog hardware description language. The Institute of Electrical and Electronics Engineers, Inc., 3 Park Ave., NY, USA, Sep, 2001.

[21] J. A. A. Flores, M. Acacio. Exploiting address compression and heterogeneous interconnects for efficient message management in tiled cmps. *Journal of Systems Architecture*, 56(9):429–441, Sept. 2010.

[22] M. E. A. Kumar, M. Bayoumi. A methodology for low power scheduling with resources operation at multiple voltages. *Elsevier, Integration, the VLSI Journal*, 37(1):29–62, Feb. 2004.

[23] S. A. A. Lakshminarayana and S. Shukla. High level power estimation models for fpga. *in Proc. IEEE Comput. Soc. Annu. Symp. VLSI*, pages 7–12, July, 2011.

[24] T. A. A. Miyamoto, N. Homma and A. Satoh. Systematic design of rsa processors based on high-radix montgomery multipliers. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 19(7):1136 – 1146, 2011.

[25] M. L. M. C.-N. N. F. F. A. Nery, L. Jozwiak. Hardware reuse in modern application-specific processors and accelerators. *2011 14th Euromicro Conference on Digital System Design*, pages 140–147, Aug. 2011.

[26] W. R. A. Parekhji, M. Butler. Guest editors' introduction: Speeding up analog integration and test for mixed-signal socs. *Design & Test, IEEE*, 32(1):6 – 8, 2015.

[27] K. T. A. Satoh, S. Morioka and S. Munetoh. A compact rijndael hardware architecture with s-box op-timization. *in Proc. ASIACRYPT*, pages 239–245, Dec. 2000.

[28] G. P. S. P. A.D. Grasso, D. Marano. Design methodology of subthreshold three-stage cmos otas suitable for ultra-low-power low-area and high driving capability. *IEEE Trans. Circuits and Syst. I, Reg. Papers*, 62(6):1453 – 1462, 2015.

[29] N. Ahmad and S. R. Hasan. Efficient integrated aes crypto-processor architecture for 8-bit stream cipher. *Electronics Letters*, 48(23):1456 – 1457, 2012.

[30] M. M.-K. AReyhani-Masoleh. Efficient and high-performance parallel hardware architec-tures for the aes-gcm. *IEEE Trans. Comput.*, 61(8):1165–1178, Aug. 2012.

[31] M. P. B. Amelifard. Design of an efficient power delivery network in a soc to enable dynamic power management. *2007 IEEE International Symposium on Low Power Electronics and Design (2007 ISLPED)*, pages 328–333, Aug. 2007.

[32] M. H. J.-E. J. J. K. B. Cc Lim, J. Mao. Digital analog design: Enabling mixed-signal system validation. *IEEE Design & Test*, 32(1):44 – 52, 2015.

[33] C.-Y. H. B.-H. Wu. A robust constraint solving framework for multiple constraint sets in constrained random verification. *2013 50th ACM/EDAC/IEEE Design Automation Conference (2013 DAC)*, pages 1 – 7, 2013.

[34] A. S. a. P. D. B. Pal, A. BaneIjee. Accelerating assertion coverage with adaptive test benches. volume 27, pages 967–9725, May 2008.

[35] G. W. D. B. B. Reagen, Y. Shao. Quantifying acceleration: Power performance trade-offs of application kernels in hardware. *2013 IEEE International Symposium on Low Power Electronics and Design (2013 ISLPED)*, pages 395–400, Sept. 2013.

[36] D. W. D. W. C. H. B. Schneier, J. Kelsey and N. Ferguson. The twofish encryption algorithm: A 128-bit block cipher. John Wiley & Sons, 1999.

[37] D. W. D. W. C. H. B. Schneier, J. Kelsey and N. Ferguson. Twofish: A 128-bit block cipher. NIST AES Proposal, June 1998.

[38] S. Baeg. Low-power ternary content-addressable memory design using a segmented match line. *IEEE Trans. Circuits and Syst. I, Reg. Papers*, 55(6):1485 – 1494, 2008.

[39] J. Bergeron. In *Writing Testbenches using SystemVerilog.* Springer, USA, 2006.

[40] E. D. A. R. G. S. H. C. J. S. M. L. C. M. P. D. S. C. Burwick, D. Coppersmith and N. Zunic. Mars-a candidata cipher for aes. NIST AES Proposal, Junn 1998.

[41] C. L. C. C. Hsing Wang and C. W. Wu. An efficient multimode multiplier supporting aes and fun-damental operations of public-key cryptosystems. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 18(4):553–563, Apr. 2010.

[42] S. H. B. X. C. Liang, G. Zhong. Uvm-ams based sub-system verification of wireless power receiver soc. *2014 12th IEEE International Conference on Solid-State and Integrated Circuit Technology (2014 ICSICT),*, pages 1 – 3, 2014.

[43] R. P. A. R. C. Sarkar, A. Nambi. Diat: A scalable distributed architecture for iot. *IEEE Internet of Things Journal*, 2(3):230–239, Dec., 2014.

[44] K. K. C. Wu, F. Huang and I. Huang. An ocp-ahb bus wrapper with build-in ice support for soc integration. *2012 International Symposium on VLSI Design, Automation, and Test (2012 VLSI-DAT)*, pages 1–4, Apr. 2012.

[45] R. D. L. S. R. K. C. Zhu, Z. Gu. Characterization of single-electron tunneling transistors for designing low-power embedded systems. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 17(5):646 – 659, 2009.

[46] D. B. R. B. G. D. R. G. W. L. A. S. B. T. C.A. Krygowski, E. Almog. Key advances in the presilicon functional verification of the ibm enterprise microprocessor and storage hierarchy. *IBM Journal of Research and Development*, 56(12):13:1 – 13:16, 2012.

[47] D. Canright. A very compact rijnael s-box. 2005.

[48] C. Chou and R. Marculescu. Designing heterogeneous embedded network-on-chip platforms with users in mind. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst. (TCAD)*, 29(9):1301–1314, Sept. 2010.

[49] C. Chung and J. Kim. Broadcast filtering: Snoop energy reduction in shared bus-based low-power mpsocs. *J. Syst. Archit.*, 55(3):196–208, March 2009.

[50] A. Cilardo. Exploring the potential of threshold logic for cryptography-related operations. *IEEE Trans. Compt.*, 60(4):452 – 462, 2011.

[51] A. Cohen and K. Parhi. Architecture optimizations for the rsa public key cryptosystem: A tutorial. *IEEE Circuits and Systems Magazine*, 11(4):24 – 34, 2011.

[52] K. Compton and S. Hauck. Automatic design of reconfigurable domain-specific flexible cores. *IEEE Trans. Very Large Scale Integr. Syst. (TVLSI)*, 16(5):493–503, May 2008.

[53] M. G. Corp.

[54] H. M. A. K. D. S. A. Elminaam and M. M. Hadhoud. Performance evaluation of symmetric encryption algorithm. *IJCSNS International Journal of Computer Science and Network Security*, 8(12):280–286, Dec.2008.

[55] J. Daemen and V. Rijmen. Aes proposal: Rijndael. NIST AES Proposal, June 1998.

[56] J. Z. J. H. D.M. Wang, Y.Y. Ding and H. Tan. Area-efficient and ultra-low-power architecture of rsa processor for rfid. *Electronics Letters*, 48(19):1185 – 1187, 2012.

[57] C. J. R. C. V. B. M. A.-R. J. W. S. Y. G. N. T. A. C. C. M. H. G. Y. M. S. E. Terzioglu, S. S. Yoon. Low power embedded memory design c process to system level considerations. *2011 IEEE International Conference on IC Design & Technology (2011 ICICDT)*, pages 1 – 4, 2011.

[58] G. R. E. Thambiraja and R. Umarani. A survey on various most common encryption techniques. *International Journal of Advanced Research in Computer Science and Software Engineering*, 2(7):226–233, July 2012.

[59] K. D. F. Shao, R. Sun and X. Wang. A new secure architecture of network computer based on single cpu and dual bus. *Fifth IEEE Int. Symposium on Embedded Computing*, pages 309–314, Oct. 2008.

[60] V. Fischer. Realization of the round 2 candidates using altera fpga. *Comments Third Advanced Encryption Standard Candidates Conf. (AES3)*, Apr. 2000.

[61] V. Fischer and M. Drutarovsky. Two methods of rijndael implementation in reconfigurable hardware. *in Proc. CHES 2001*, page 77C92, May 2001.

[62] B. A. Forouzan. Data communications and networking. McGraw-Hill Science, 2006.

[63] I. K. P. M. G. Bertoni, L. Breveglieri and V. Piuri. Error analysis and detection procedures for a hardware implementation of the advanced encryption standard. *IEEE Trans. Comput.*, 52(4):492C505, Apr. 2003.

[64] T. R. N. C. F. M. G. Guindani, C. Reinbrecht. Noc power estimation at the rtl abstraction level. *2008 IEEE Computer Computer Society Annual Symposium on VLSI*, pages 475–478, Apr. 2008.

[65] J.-J. Q. J. L. G. Rouvroy, F.-X. Standaert. Efficient uses of fpgas for implementations of des and its experimental linear cryptanalysis. *IEEE Trans. Compt.*, 52(4):473 – 482, 2003.

[66] M. M.-X. Z. G. Yang, L. Xie. A health-iot platform based on the integration of intelligent packaging unobtrusive bio-sensor, and intelligent medicine box. *IEEE Trans. on Industrial Informatics*, 10(4):2180–2191, Feb, 2014.

[67] K. Gaj and P. Chodowiec. Fast implementation and fair comparison of the final candidates for advanced encryption standard using field programmable gate arrays. *Proc. Cryptographers Track RSA Conf. (2001 CT-RSA)*, pages 84–99, 2001.

[68] K. Gaj and P. Chodowiec. Comparison of the hardware performance of the aes candidates using reconfigurable hardware. *Proc. Third Advanced Encryption Standard Candidate Conf. (AES3),*, pages 40–54, Apr. 2000.

[69] R. R. G.Bonfini, M.Chiavacci. A new verification approach for mixed-signal systems. *2005 IEEE International Behavioral Modeling and Simulation Conference(BMAS 2005)*, 2012.

[70] M. Glasser. mark2009ovm. In *Open Verification Methodology Cookbook*, Wilsonville, OR, USA, 2009. Springer.

[71] R. C. Gonzalez and R. E. Woods. Digital image processing. page 68C99, Englewood Cliffs, NJ, USA: Prentice-Hall.

[72] T. Good and M. Benaissa. 692-nw advanced encryption standard (aes) on a 0.13-um cmos. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 18(12):1753–1757, Dec. 2012.

[73] R. Grundler. Ip subsystems: The next frontier for ip integration. Synopsys, Mountain View, CA, USA, 2016.

[74] X. Guo and R. Karri. Low-cost concurrent error detection for gcm and ccm. *J. Electronic Testing*, 30(6):725–737, 2014.

[75] B. F. H. Michel, F. Bubenhagen and H. Michalik. Amba to socwire network on chip bridge as a backbone for a dynamic reconfigurable processing unit. *2011 NASA/ESA Conference on Adaptive Hardware and Systems (2011 AHS)*, pages 227–233, Jun. 2011.

[76] A. K. H.D. Foster and D. Lacey. Assertion-based design. Kluwer Academic Publishers, 2004.

[77] A. Hodjat and I. Verbauwhede. Area-throughput trade-offs for fully pipelined 30 to 70 gbits/s aes processors. *IEEE Trans. Comput.*, 55(4):366–372, Apr. 2006.

[78] X. Huang and W. Wang. A novel and efficient design for an rsa cryptosystem with a very large key size. *IEEE Trans. Circuits Syst. II. Exp. Briefs*, 62(10):972 – 976, 2015.

[79] D. A. I. Beretta, V. Rana and D. Sciuto. A mapping flow for dynamically reconfigurable multi-core system-on-chip design. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst. (TCAD)*, 30(8):1211–1224, Aug. 2011.

[80] C. T. I. Iliuta. Constraint random stimuli and functional coverage on mixed signal verification. *2014 International Semiconductor Conference (2014 CAS)*, pages 237 – 240, 2014.

[81] P. S. I. Verbauwhede and H. Kuo. Design and performance testing of a 2.29 gb/s rijndael processor. *IEEE J. Solid-State Circuits (JSSC)*, Mar. 2003.

[82] A. H.-A. N. J. Bergeron, E. Cerny. Verification methodology manual for system verilog. In *Springer Publisher*, page 260C282, 2005.

[83] A. H.-A. N. J. Bergeron, E. Cerny. In *Verification Methodology Manual for SystemVerilog*. Springer, USA,, 2006.

[84] A. N. J. Cao. A markov performance model for buffered protocol design. *in: proceedings of the 2011 IEEE Computer Society Annual Symposium on VLSI*, page 170C175.

[85] S.-I. C. J. Cho, S. Choi. Constrained-random bitstream generation for h.264/ avc decoder conformance test. *IEEE Trans. Consum. Electron*, 56(2):848C855, 2010.

[86] Z. W. X. H. K. Z. J. He, H. Liu. High-speed low-power viterbi decoder design for tcm decoders. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 20(4):755 – 759, 2012.

[87] X. S. J. Hu. The design methodology and practice of low power soc. *2008 International Conference on Embedded Software and Systems Symposia (2008 ICESS)*, pages 185 – 190, 2008.

[88] H.-J. L. J. Lee and C. Lee. A phase-based approach for on-chip bus architecture optimization. *Comput. J.*, 52(6):626–645, Aug. 2009.

[89] K. H. J. Leu, C. Chen. Improving heterogeneous soa-based iot message stability by shortest processing timing scheduling. *IEEE Trans. on Services Computing*, 7(4):575–585, May, 2013.

[90] A. C. J. Li, W. Song. Real-energy: a new framework and a case study to evaluate power-aware real-time scheduling algorithms. *2010 IEEE International Symposium on Low-Power Electronics and Design*, page 153C158, Aug. 2010.

[91] E. O. J. Wolkerstorfer and M. Lamberger. An asic implementation of the aes s-boxes. *in proc. ASICRYPT*, pages 239–245, Dec. 2000.

[92] D. S. P.-E. G. Y. L. G. D. M. J. Zhang, M. D. Marchi. Polarity-controllable silicon nanowire transistors with dual threshold voltages. *IEEE Trans. Electron Devices*, 61(11):3654 – 3660, 2014.

[93] V. R. JoanDaemen. The design of rijndael, aes c the advanced encryption standard, springer c verilag. page 238, 2002.

[94] K. Jones. Analog and mixed signal verification. *The FMCAD 2008 Formal Methods in Computer Aided Design*, pages 17–20, Nov. 2008.

[95] J. L. J. J. J. B. V. P. M. B. A. Z. J. S. C. M. J. K. J. H. B. B. K.-D. Schubert, W. Roesner. Functional verification of the ibm power7 microprocessor and power7 multiprocessor systems. *IBM Journal of Research and Development*, 55(3):10:1 – 10:17, 2011.

[96] K. O. K. Iokibe, T. Amano and Y. Toyota. Equivalent circuit modeling of cryptographic integrated circuit for information security design. *IEEE Trans. Electromagn. Compat.*, 55(3):581 – 588, 2013.

[97] G. L. K. Lahiri, A. Raghunathan. The lotterybus on-chip communication architecture. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 14(6):596–608, June 2006.

[98] A. R. K. Sekar, K. Lahiri and S. Dey. Dynamically configurable bus topologies for high-performance on-chip communication. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 16(10):1413–1426, Oct. 2008.

[99] O. A. M. K. Stevens. Single-chip fpga implementation of a pipelined, memory-based aes. *Canadian Conference on Electrical and Computer Engineering*, pages 1296–1299, 2005.

[100] S. Kim and S. Ha. Efficient exploration of bus-based system-on-chip architectures. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 14(7):681–692, July 2006.

[101] K. Kundert. Principles of top-down mixed-signal design. *[Online]*.

[102] C. L. L. Chiou, Y. Chen. System-level bus-based communication architecture exploration using a pseudoparallel algorithm. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, 28(8):1213–1223, Aug. 2009.

[103] W. T. S. V. L. Liu, D. Sheridan. A technique for test coverage closure using goldmine. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, 31(5):790 – 803, 2012.

[104] S. S. L. Lu, J.V. McCanny. Reconfigurable system-on-a-chip motion estimation architecture for multi-standard video coding. *IET Computers & Digital Techniques*, 4(5):349 – 364, 2010.

[105] R. K. V. V. L. Tumonis, M. Schneider. The structural mechanics of rail guns with discrete supports showing the influence of des. *IEEE Trans. Plasma Sci.*, 39(1):144 – 148, 2011.

[106] P. L. L. Yin, Y. Deng. Simulation-assisted formal verification of nonlinear mixed-signal circuits with bayesian inference guidance. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, 32(7):977 – 990, 2013.

[107] J. Lee. Design methodology for on-chip bus architectures using system-on-chip network protocol. *IET Circuits Devices Syst.*, 6(2):85–94, Mar. 2012.

[108] J. Lee and H.-J. Lee. Wire optimization for multimedia soc and sip designs. *IEEE Trans. Circuits and Syst. I, Reg. Papers*, 55(8):2002–2215, Sept. 2008.

[109] K. Li. Improving multicore server performance and reducing energy consumption by workload dependent dynamic power management. *IEEE Trans. Cloud Comput.*, PP(99):1 – 1, 2015.

[110] B. Liu and B. Baas. Parallel aes encryption engines for many-core processor arrays. *IEEE Trans. Compt.*, 62(3):536 – 547, 2013.

[111] R. G. F. K. M. Barachi, A. Kadiwal. The design and implementation of architectural components for the integration of the ip multimedia subsystem and wireless sensor networks. *IEEE Communications Magazine*, 48(4):42–50, Apr. 2010.

[112] K. E. T. V. J.-P. C. M.-M. L. F. P. Z. W. P. C. I. N. T. N. R. L. E. V. M. Barnasconi, M. Dietrich. Uvm-systemc-ams framework for system-level verification and validation of automotive use cases. *Design & Test, IEEE*, 32(6):76 – 86, 2015.

[113] H.-H. W.-W.-C. L. M.-D. Shieh, J.-H. Chen. A new modular exponentiation architecture for efficient design of rsa cryptosystem. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 16(9):1151 – 1161, 2008.

[114] M. K. J. T. V. D. M. Ghoneima, Y. Ismail. Serial-link bus: a low-power on-chip bus architecture. *2005 IEEE/ACM International Conference on Computer-Aided Design (2005 ICCAD)*, pages 541 – 546, 2005.

[115] J. T. Y. Y. N. K.-J. B.-S. N. Y. I. V. D. M. Khellah, M. Ghoneima. A skewed repeater bus architecture for on-chip energy reduction in microprocessors. *2005 IEEE International Conference on Computer Design: VLSI in Computers and Processors (2005 ICCD).*, pages 253 – 257, 2005.

[116] E. S. M. L. E. M. Lorenz, L. Mengibar. Low power data processing system with self-reconfigurable architecture. *Journal of Systems Architecture*, 53(9):568–576, Sept. 2007.

[117] A. K. N. M. M. Wong, M. L. D. Wong and I. Hijazin. Construction of optimum composite field architecture for compact high-throughput aes s-boxes. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 20(6):1151–1155, Jun. 2012.

[118] M. K. M. Rani. Accessing ahb bus using wishbone master in soc design. *2012 10th IEEE International Conference on Semiconductor Electronics (2012 ICSE)*, pages 631–635, Sept. 2012.

[119] P. V. M. Rungeler, J. Bunte. Design and evaluation of hybrid digital-analog transmission outperforming purely digital concepts. *IEEE Trans. Commun.*, 62(11):3983 – 3996, 2014.

[120] P. Maffetone. The maffetone method: The holistic, low-stress, no-pain way to exceptional fitness, ragged mountain press. Camden, ME, USA. 2000.

[121] P. Maffetone. The big book of endurance training and racing. Skyhorse publishing, New York, NY, USA. 2010.

[122] M. Masoumi and M. H. Rezayati. Novel approach to protect advanced encryption standard algorithm implementation against differential electromagnetic and power analysis. *IEEE Trans. Inf. Forensics Security*, 10(2):256 – 265, 2015.

[123] M. McLoone and J. McCanny. High-performance fpga implementation of des using a novel method for implementing the key schedule. *IEE Proceedings - Circuits, Devices and Systems*, 150(5):373–8, 2003.

[124] M. McLoone and J. V. McCanny. Rijndael fpga implementation utilizing look-up tables. *IEEE Workshop on Signal Processing Systems*, page 349C360, Sept. 2001.

[125] F. Mischkalla and W. Mueller. Architectural low-power design using transaction-based system modeling and simulation. *2014 International Con-*

ference on Embedded Computer Systems: Architectures, Modeling, and Simulation (2014 SAMOS XIV), pages 258 – 265, 2014.

[126] M. Mozaffari-Kermani and A. Reyhani-Masoleh. Efficient and high-performance parallel hardware architectures for the aes-gcm. *IEEE Trans. Compt.*, 61(8):1165 – 1178, 2012.

[127] E. N. Mui. Practical implementation of rijndael s-box using combina-tional logic. 2007.

[128] G. K. M.W.K. Nomani, M. Anis. Statistical approach for yield optimization for minimum energy operation in subthreshold circuits considering variability issues. *IEEE Trans. Semiconductor Manufacturing*, 23(1):77 – 86, 2010.

[129] K. S. N. Homma and T. Aoki. Toward formal design of practical cryptographic hardware based on galois field arithmetic. *IEEE Trans. Compt.*, 63(10):2604 – 2613, 2014.

[130] Y. K. N. Khan and H. Fang. Metric diven verification of mixedsignal designs. *Design and Verification Conference(DVCon),*, 2011.

[131] D. S. N. Kroupis. Filesppa: Fast instruction level embedded system power and performance analyzer. *Microprocess. Microsyst.*, 35(3):329–342, May 2011.

[132] B. P. N. Mentens, L. Batinan and I. Verbauwhede. A systematic evaluation of compact hardware implementa-tion for the rijndael s-box. *in Proc. Topics Cryptology (CT-RSA)*, 3376:323–333, 2005.

[133] J. Nunez-Yanez and G. Lore. Enabling accurate modeling of power and energy consumption in an arm-based system-on-chip. *Microprocess. Microsyst*, 37(3):7–12, 2011.

[134] J. L. H. F. O. Guzey, L.-C. Wang. Increasing the efficiency of simulation-based functional verification through unsupervised support vector analysis. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, 29(1):138 – 148, 2010.

[135] D.-N. L. P.-C. Hsiu, C.-K. Hsieh and T.-W. Kuo. Multilayer bus optimization for real-time embedded systems. *IEEE Trans. Comput.*, 61(11):1638–1650, Nov. 2012.

[136] Y. J. P. Chen, C. Wu. Bitmask-based code compression methods for balancing power consumption and code size for hard real-time embedded systems. *Microprocessors and Microsystems Journal*, 36(3):267–279, May 2012.

[137] S. D. P. G. P. Samanta, D. Chauhan. Uvm based stbus verification ip for verifying soc architectures. *18th International on VLSI Design and Test, Coimbatore*, pages 1–2, July 2014.

[138] C. Paar. Efficient vlsi architecture for bit-parallel computations in galois field. *Ph.D. dissertation, Institute for Experimental Mathematics*, 1994.

[139] E. B. R. Anderson and L. Knudsen. Serpent: A proposal for the advanced encryption standard. NIST AES Proposal, Jan. 1999.

[140] P. M. R. Karri, K. Wu and K. Yongkook. Fault-based side-channel cryptanalysis tolerant rijndael symmetric block cipher architecture. *in Proc. DFT*, page 418C426, Oct. 2001.

[141] C.-K. K. R. Lu, A. Cao. Samba-bus: A high performance bus architecture for system-on-chip. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 15(1):69–79, Jan. 2007.

[142] R. S. R. Rivest, M. Robshaw and Y. Yin. The rc6 block cipher. NIST AES Proposal, June 1998.

[143] P. B. R. Venkatasubramanian, S.K. Manohar. Nem relay-based sequential logic circuits for low-power design. *IEEE Trans. Nanotechnology*, 12(3):386 – 398, 2013.

[144] V. Rijmen. Efficient implementation of the rijndael s-box. 2000.

[145] D. L. B. H. S. Chen, H. Xu. A vision of iot: Applications, challenges, and opportunities with china perspective. *IEEE Internet of Things Journal*, 1(4):349–359, July, 2014.

[146] M. C. C. S. Fu Hsiao and C. S. Tu. Memory-free low-cost designs of advanced encryption standard using common subexpression elimination for subfunctions in transformations. *IEEE Trans. Circuits Syst. I, Reg. Papers*, 53(3):615 – 626, Mar. 2006.

[147] H. P. S. Hwang, D. Kang and K. Jhang. Implementation of a self-motivated arbitration scheme for the multilayer ahb busmatrix. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 18(5):818–830, May 2010.

[148] J.-Y. H. S.-J. Ruan, C.-Y. Wu. Low power design of precomputation-based content-addressable memory. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 16(3):331 – 335, 2008.

[149] J. S. P. V. S. Jairam, M. Rao. Clock gating for power optimization in asic design cycle theory & practice. *2008 IEEE International Symposium on Low Power Electronics and Design (2008 ISLPED)*, pages 307–308, Aug. 2008.

[150] C. M. R. T. S. B.-T. Y. S. Little, D. Walter. Verification of analog/mixed-signal circuits using labeled hybrid petri nets. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, 30(4):617 – 630, 2011.

[151] L. B. S. Murali and G. Micheli. An application-specific design methodology for on-chip crossbar generation. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst. (TCAD)*, 26(7):1283–1296, July 2007.

[152] K.-C. C. H.-W. H. S.-R. Kuang, J.-P. Wang. Energy-efficient high-throughput montgomery modular multipliers for rsa cryptosystems. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 21(11):1999 – 2009, 2013.

[153] T. L. W. W. N. V.-Y. X. S. Yang, W. Wang. Case study of reliability-aware and low-power design. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 16(7):861 – 873, 2008.

[154] S. G. Sabiro. Event-driven (rn) modeling for ams circuits. *2013 Forum on Specification & Design Languages (2013 FDL)*, pages 1 – 8, 2013.

[155] R. Sarpeshkar. Universal principles for ultra low power and energy efficient design. *IEEE Trans. Circuits and Syst. II, Exp. Briefs*, 59(4):193 – 198, 2012.

[156] M. Schubert. An analog-node model for vhdl-based simulation of rf integrated circuits. *IEEE Trans. Circuits and Syst. I, Reg. Papers*, 56(12):2717 – 2727, 2009.

[157] A. K. S.H. Chitra. A high performance on-chip segmented bus architecture using dynamic bridge-by-pass technique. *2010 International Conference on Industrial and Information Systems (2010 ICIIS)*, pages 249 – 254, 2010.

[158] G. Singh and Supriya. A study of encryption algorithms (rsa, des, 3des and aes) for information security. *International Journal of Computer Applications*, 67(19):33–38, Apr. 2013.

[159] N. Sklavos and O. Koufopavlou. Architectures and vlsi implementations of the aes-proposal rijndael. *IEEE Trans. Comput.*, 51(12):1454–1459, Dec. 2012.

[160] F. I. S.M. Ismail, A.B.M.S. Rahman. Low power design of johnson counter using clock gating. *2012 15th International Conference on Computer and Information Technology (2012 ICCIT),*, pages 510 – 517, 2012.

[161] H. Sohofi and Z. Navabi. System-level assertions: approach for electronic system-level verification. *IET Computers & Digital Techniques*, 9(3):142 – 152, 2015.

[162] W. Stallings. Cryptography and etwork security: Principles and practic. Pearson Education/Prentice Hall, 2013.

[163] W. Suntiamorntut and W. Wittayapanpracha. The study of aes encryption for wireless fpga node. *International Journal of Communications in Information Sci-ence and Management Engineering*, 12(3):40–46, Mar. 2012.

[164] C.-T. H. T.-F. Lin, C.-P. Su and C.-W. Wu. A highthroughput low-cost aes cipher chip. *Proc. IEEE Asia-Pacific Conf. ASIC*, pages 85–88, 2002.

[165] M. S. T. Maier, D. Droste. S-parameter models for transient simulation in verilog-a. *2013 9th Conference on Ph.D. Research in Microelectronics and Electronics (2013 PRIME)*, pages 249 – 252, 2013.

[166] M. B. T. Raja, V.D. Agrawal. Variable input delay cmos logic for low power design. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 17(10):1534 – 1545, 2009.

[167] R. S. T. Wang, F. Shao and H. Huang. A hardware implement of bus bridge based on single cpu and dual bus. *2008 Int. Symposium on Computer Science and Computational Technology*, pages 17–20, Dec. 2008.

[168] R. S. H. H. T. Wang, F. Shao. A hardware implement of bus bridge based on single cpu and dual bus architecture. *2008 International Symposium on Computer Science and Computational Technology (2008 ISCSCT)*, pages 17 – 20, 2008.

[169] S. R. G. Q. T. Wang, L. Niu. Multi-core ?xed-priority scheduling of realtime tasks with statistical deadline guarantee. *IEEE/ACM Design, Automation & Test in Europe Conference & Exhibition (DATE),*, page 1335C1340, Mar 2015.

[170] A. K. T. Welp, N. Kitchen. Hardware acceleration for constraint solving for random simulation. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, 31(5):779 – 789, 2012.

[171] V. S. U. Deshmukh. Stochastic automata network based approach for performance evaluation network-on-chip communication architecture. *in: proceedings of the 2010 IEEE Computer Society Annual Symposium on VLSI*, page 351C356, July 2010.

[172] Y. Z. S.-W. S. S.-P. U. R. M. F. M. U-F. Chio, H.-G. Wei. Design and experimental verification of a power effective flash-sar subranging adc. *IEEE Trans. Circuits and Syst. II, Express Briefs*, 57(8):607 – 611, 2010.

[173] V. M. E. B. R. G. K. H.-T. W. V. Gourisetty, H. Mahmoodi. Low power design flow based on unified power format and synopsys tool chain. *2013 3rd Interdisciplinary Engineering Design Education Conference (2013 IEDEC)*, pages 28 – 31, 2013.

[174] L. M. Y. W. F. L. J. Z.-X. Y. D. L. W. Teng, J. Sun.

[175] 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. Exp. Briefs*, 60(1):36–40, Jan. 2013.

[176] L. H. W.Hartong and E. Barke. Model checking algorithms for analog verification. *ACM/IEEE Design Automation Conference (DAC)*, pages 542–547, 2002.

[177] G. L. R. C. X. Han, G. Kim. An optical centralized shared-bus architecture demonstrator for microprocessor-to-memory interconnects. *IEEE Journal of Selected Topics in Quantum Electronics*, 9(2):512 – 517, 2003.

[178] J. A. X. Yang, N. Wu. A novel bus transfer mode (as transfer) and a performance evaluation methodology. *Elsevier, Integration, the VLSI Journal*, 52:23–33, July 2015.

[179] J. A. X. Yang, N. Wu. A low-cost and high-performance embedded system architecture and an evaluation methodology. *IEEE Computer Society Annual Symposium on VLSI (2014 ISVLSI)*, pages 240–243, March 2014.

[180] Y. C. N. C. Y. Kim, S. Park. System-level online power estimation using an on-chip bus performance monitoring unit. *in: Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, page 149C154, 2008.

[181] E.-J. C. Y.-S. Cho and K.-R. Cho. Modeling and analysis of the system bus latency on the soc platform. *in Proc. Int. Workshop System-Level Interconnect Predict*, pages 67–74, March 2006.

[182] R. W. Y. Wang, S. Joeres and S. Heinen. Modeling approaches for functional verification of rf-socs: Limits and future requirements. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, 28(5):769 – 773, 2009.

[183] X. L. A. A. M. P. Y. Wang, M. Triki. Hierarchical dynamic power management using model-free reinforcement learning. *2013 14th International Symposium on Quality Electronic Design (2013 ISQED)*, pages 170 – 177, 2013.

[184] L. L. W. W. K. C. J. J.-H. J. S.-Y. A. Y. Zhang, Q. Tong. Automatic register transfer level cad tool design for advanced clock gating and low power schemes. *2012 International SoC Design Conference (2012 ISOCC)*, pages 21 – 24, 2012.

[185] X. Yang.

[186] X. Yang and J. Andrian. A high performance on-chip bus (msbus) design and verification. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst. (TVLSI)*, 23(7):1350–1354, July 2014.

[187] X. Yang and J. Fan. Mixed-signal system-on-chip (soc) veri?cation based on system verilog models. *The 45th Southeastern Symposium on System Theory (2013 SSST)*, pages 17–21, Mar. 2013.

[188] C.-T. K. Y.S.-C. Huang, K.C.-K. Chou. Application-driven end-to-end traffic predictions for low power noc design. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 21(2):229 – 238, 2013.

[189] S.-D. T.-B. L. G. S. Y. C. Z. Hao, R. Shen. Statistical full-chip dynamic power estimation considering spatial correlations. *2011 12th International Symposium on Quality Electronic Design (2011 ISQED),*, pages 1 – 6, 2011.

[190] D. L. X. Z. Z. Liu, Q. Zhu. Off-chip memory encryption and integrity protection based on aes-gcm in embedded systems. *IEEE Design & Test*, 30(5):54 – 62, 2013.

[191] P. Z. Z. Zhang, D. Wu and D. Xie. A flexible vlsi architecture of transport processor for an avs hdtv decoder soc. *IEEE Trans. Consum. Electron. (TCE)*, 52(4):1427–1432, Nov. 2006.

[192] X. Zhang and K. Parhi. On the optimum constructions of composite field for the aes algorithm. *IEEE Trans. Circuits Syst. II. Exp. Briefs*, 53(10):1153–1157, Oct. 2006.

[193] X. Zhang and K. Parhi. High-speed vlsi architecture for the aes algorithm. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 12(9):957–967, Sept. 2004.

VITA

XIAOKUN YANG

2007	M.S., Software Engineering Beihang University Beijing, China
2007	M.S., Electrical and Computer Engineering Florida International University Florida, USA
2007-2008	ASIC Design Engineer, SoC Group PowerLayer MicroSystems Corp. Beijing, China
2009-2010	Sr. ASIC Verification Engineer, WLAN Group China Electronics Corp. Beijing, China
2011-2012	Sr. ASIC Design/Layout Engineer, GPU Group Advanced Micro Devices (AMD) Beijing, China
2016	Ph.D., Electrical and Computer Engineering Florida International University Florida, USA

FIRST AUTHORED PUBLICATIONS

Xiaokun Yang, Jean H. Andrian, (2014). *A High Performance On-Chip Bus (MS-BUS) Design and Verification*, IEEE Trans. Very Large Scale Integr. (VLSI) Syst. (TVLSI), Vol. 23, Issue: 7, PP. 1350-1354.

Xiaokun Yang, Nansong Wu, Jean H. Andrian, (2015). *A Novel Bus Transfer Mode: Block Transfer and A Performance Evaluation Methodology*, Elsevier, Integration, the VLSI Journal, Vol. 52, PP. 23-33.

Xiaokun Yang, Jean H. Andrian, (2015). *An Advanced Bus Architecture for AES-Encrypted High-Performance Embedded Systems*, ACM Trans. Archit. Code Optim. (TACO), Under Review.

Xiaokun Yang, Jean H. Andrian, (2016). *A Configurable and Synthesizable On-Chip Bus Architecture for Integrating Industrial Standard IPs*, The 52nd Design Automation Conference (2016 DAC), Poster Session, June 2016, Accepted.

Xiaokun Yang, Jean H. Andrian, (2014). *A Low-Cost and High-Performance Embedded System Architecture and An Evaluation Methodology*, IEEE Computer Society Annual Symposium on VLSI (2014 ISVLSI), Best Ph.D Forum Paper Award, Mar. 2014, pp. 240-243.

Xiaokun Yang, Jeffery Fan, (2013). *Mixed-Signal System-on-Chip (SoC) Verification Based on System Verilog Model*, The 45th Southeastern Symposium on System Theory (2013 SSST).

Xiaokun Yang, Jean H. Andrian, (2016). *Implementation of A High Performance DBUS for AES-Encrypted Circuits*, IEEE Computer Society Annual Symposium on VLSI (2016 ISVLSI), Under Review.

Xiaokun Yang, Jean H. Andrian, (2016). *Design of A High-Performance and High-Security Circuit for Bio-Feedback Cardio Equipment Using 33-Step Training Table*, IEEE Computer Society Annual Symposium on VLSI (2016 ISVLSI), Under Review.

Xiaokun Yang, Jean H. Andrian, (2015). *An Advanced Bus Architecture for AES-Encrypted High-Performance Embedded Systems*, US Patent, (filed).

Xiaokun Yang, (2013). *A Mixed-Signal Verification System for Sigma-Delta Filters*, Patent, CN 102955871A.