An Adaptive Cryptographic and Embedded System Design with Hardware Virtualization

Chun-Hsian Huang
Department of Computer Science and Information Engineering,
National Taitung University, Taiwan

Abstract— *This work proposes an adaptive cryptographic and embedded system (ACES) design that can adapt its hardware and software functionalities at runtime to different system requirements. By using the hardware virtualization technique in the ACES design, a fixed set of logic resources can be configured as different hardware modules at runtime to support multiple software applications. Further, by taking the advantages of architectural characteristics of FPGAs, the ACES can support high-performance computing for computing-intensive functions such as cryptographic and image processing functions. Experiments with ubiquitous computing applications have also demonstrated that the ACES can accelerate by up to 26.5x the processing time required by using the software solution. Compared to the traditional embedded system design, the ACES can reduce 29% of slice registers and 33% of slice LUTs required for supporting all the five required hardware functions. Through the advantage of system adaptation, the ACES can also dynamically reduce its power consumption at runtime, according to different environmental conditions.*

Keywords: Adaptability, hardware virtualization, partial reconfiguration

1. Introduction

As network technology scaling, transferring information and data between different electronic devices becomes more and more convenient. Further, with the increase in user requirements, mobile ubiquitous computing applications enable information processing to be thoroughly integrated into everyday living. In such a ubiquitous computing environment, services and devices can be dynamically adapted to changing environments.

The target applications of this work are mainly used in the ubiquitous computing environments, especially for the field of image processing and cryptographic applications. To support dynamically changing and unpredictable ubiquitous computing applications, adaptability becomes a key requirement in providing high-performance computing and complete data protection on the network in this work. However, in the most existing dynamic adaptive approaches [1]–[5], only software services and applications can be adapted, and hardware devices support the changing software applications passively. This means that hardware functions cannot be reconfigured at runtime, which also leads to the inefficient use of hardware resources. To be able to not only adapt on-demand functionalities but also provide better system performance, designing an efficient embedded system architecture to meet the dynamic requirements of various environmental situations becomes very important.

This work tries to solve the above problem about system adaptability and performance by proposing an *Adaptive Cryptographic and Embedded System* (ACES) design. Figure 1 gives an example for illustrating the practicability of the proposed ACES. Here, real-time image are captured from the camera and then displayed on the monitor. The filters are used to reduce the effects of the noise in the source images for further image processing applications. The images can also be transferred to a client via the network. To ensure the security of data transfers on the network, all data are first encrypted and then transferred to the client. Based on the effects of noise in the source images and the security requirements of data transfers, the ACES can adapt on-demand its filter and cryptographic functions for providing better *Quality-of-Service* (QoS).

Figure 1 gives the ideal blueprint to apply the ACES to ubiquitous computing environments. It must solve the following issues related to ubiquitous computing, including 1) what method can make hardware adaptable? 2) how to use hardware resources efficiently? 3) how to support high-performance computing and reduce power consumption at runtime?

To make hardware adaptable, the ACES design integrates the dynamic partial reconfiguration technology from Xilinx [6]. Thus, one part of the FPGA device in the ACES is being reconfigured, while other parts can remain operational without being affected by reconfiguration. This shows that the filter and cryptographic hardware functions in the ACES can be dynamically adapted to different environmental requirements. Further, the partial reconfiguration technology can also be considered as the gate-level hardware virtualization technique, using which multiple applications can access a fixed set of logic resources in a temporally exclusive way. Thus, the utilization of hardware resources can be increased significantly. For system performance, computing-intensive functions such as filter and cryptographic functions are implemented in hardware, so that the ACES can take the advantages of architectural characteristics of FPGAs for

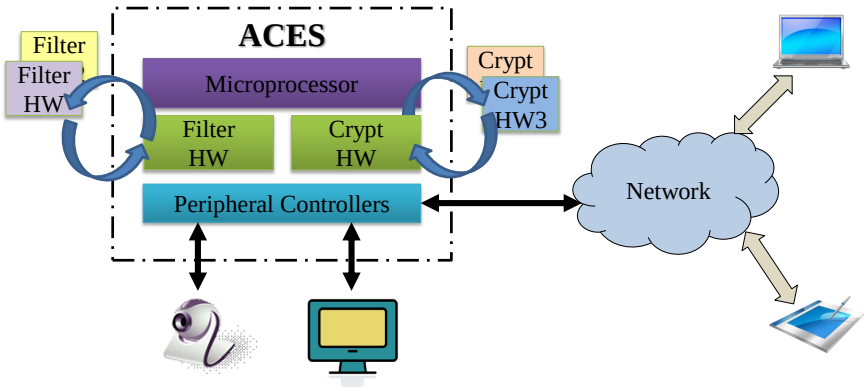


Fig. 1: Application Example

further enhancing system performance. Further, the ACES contains the blank modules to disable the functionality of the partial reconfigurable region in the FPGA. When no requests are received from software applications, the blank module can be used in the ACES to reduce power consumption at runtime.

The rest of this paper is organized as follows. Section 2 introduces the related work. The detailed ACES design is illustrated in Section 3. Section 4 presents our experiments and analyses, and conclusions are given in Section 5.

2. Related Work

In a ubiquitous computing environment, computing devices can be adapted to environmental changes for satisfying user requirements. To support the capability of adaptation more efficiently, Efstratious et al. [1] proposed an architecture that could support adaptive context-aware applications. However, their infrastructure only notified applications about the environmental changes, and application themselves needed to trigger the adaptive mechanism. Instead of the passive application adaptation, Ghim et al. [2] further proposed a reflective approach to dynamic adaptation that could perform adaptation operation triggered by changes in the policy and context. The other existing work [3]–[5] also adopted the software solutions to adapt itself to different system requirements. However, in the above designs [1]–[5], hardware cannot be adapted to different requirements, which also restricts system adaptation and performance.

As for our target cryptographic applications, the corresponding algorithms are usually computation-intensive, hard real-time and non-adaptive to changing network conditions. The algorithms make different tradeoffs between security and complexity. To allow multiple tradeoffs and to adapt to changing network conditions at runtime, a data protective process needs a high-speed and flexible embedded system. T. Wollinger and C. Paar [7] demonstrated the advantages of reconfigurable devices for cryptographic applications in embedded systems, including architecture efficiency, resource efficiency, throughput, and algorithm agility. Lagger et al. [8] also compared a full-software design with a coprocessor design embedded with an FPGA device that could be configured with *Data Encryption Standard* (DES), triple DES, and *Route Coloniale* 4 (RC4) hardware cores. Compared to the software solution, the performance of the FPGA-based design was significantly enhanced due to the specific architecture. In other related researches such as [9], [10], they leveraged the advantages of reconfigurable FPGA to further enhance the performance of cryptographic hardware applications. All the above researches [7]–[10] have demonstrated that reconfigurable FPGAs are very suitable for implementing cryptographic applications.

By using the architectural advantages of FPGAs, cryptographic functions of the ACES are implemented in hardware to support high-performance computing. Compared to the software solutions [1]–[5], the ACES design supports the hardware virtualization technique, so that system adaptability and hardware resource utilization can be enhanced significantly. The details of the ACES design will be introduced in Section 3.

3. Adaptive Cryptographic and Embedded System Design

To support high-performance and adaptive features, the proposed ACES design is realized on an FPGA device, as shown in Figure 2. The capture controller is responsible for capturing real-time images from the camera, while the display controller is responsible for displaying the processing results on the monitor. Before the processing results are transferred to a client via the network, they are first encrypted through the cryptographic hardware function. To reduce the effects of noise in the source images, the filter function is also integrated into the image processing hardware function.

Besides realizing the image processing function and the cryptographic function as hardware circuits for enhancing

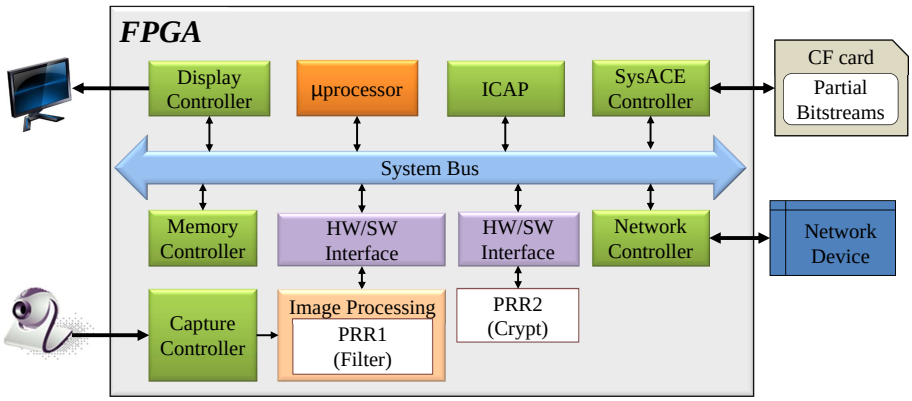


Fig. 2: Adaptive cryptographic and embedded system design

system performance, the ACES further integrates the hardware virtualization technology to increase the utilization of hardware resources. As shown in Figure 2, two *Partially Reconfigurable Regions* (PRRs), namely `PRR1` and `PRR2`, are implemented in the FPGA for configuring the filter function and the cryptographic function, respectively. Thus, the logic resources of each PR region can be reconfigured as different hardware functions, according to system requirements. Therefore, besides the traditional software adaptation, the ACES can also support hardware adaptation.

Based on the partial reconfiguration flow [6], two blank modules are also individually generated to disable the functionalities of `PRR1` and `PRR2`. All the partial bitstreams corresponding to the reconfigurable hardware modules are stored in a CF card, and they are accessed by using the SysACE controller. To support system adaptability, an *Internal Configuration Access Port* (ICAP) [11] controller is also implemented in the ACES for configuring the corresponding partial bitstreams. Through the ICAP controller, the ACES can dynamically adapt its hardware functionalities at runtime, without the user's intervention. To provide efficient hardware/software communication and to support complete system adaptation, a hardware/software communication interface, a virtualizable and hierarchical design, and an adaptation policy are also proposed in the ACES. The details are described in the following sections.

3.1 Hardware/Software Communication Interface

To support seamless data transfers between reconfigurable modules and the microprocessor efficiently, a hardware/software interface component based on the *Intellectual Property Interface* (IPIF) is proposed in the ACES, as shown in Figure 3. It contains a bidirectional buffer and a device interrupt controller. Through the hardware/software interface component, the processing results of the reconfigurable hardware module can be stored in the bidirectional buffer sequentially, while the microprocessor can read the

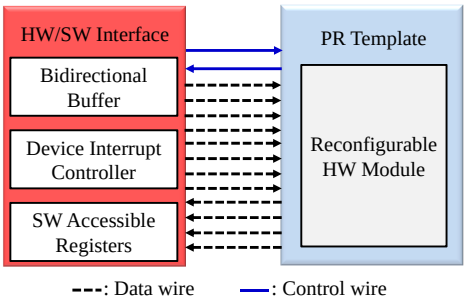


Fig. 3: Hardware/software interface component and PR template

processing results from the bidirectional buffer at the same time. To ensure that all processing results can be read by the microprocessor in real-time, the device interrupt controller is used to notify the microprocessor to read the processing results.

To enhance system scalability, our previously proposed partially reconfigurable template (PR template) [12] is also used in the ACES to ease the integration of user-designed hardware functions with different I/O interfaces. The PR template consists of eight 32-bit input data signals, one 32-bit input control signal, four 32-bit output data signals, and one 32-bit output control signal. To bridge with the interface of the PR template, the proposed hardware/software interface component also contains fourteen software accessible registers for the microprocessor to access the reconfigurable hardware module. Therefore, through the use of the PR template and the hardware/software communication interface component, new user-designed hardware functions can be easily integrated into the ACES.

3.2 Virtualizable and Hierarchical Design

To provide a complete hardware virtualization mechanism, besides the support of the hardware/software interface design as described in Section 3.1, the device drivers corresponding to different hardware functions need to be also implemented

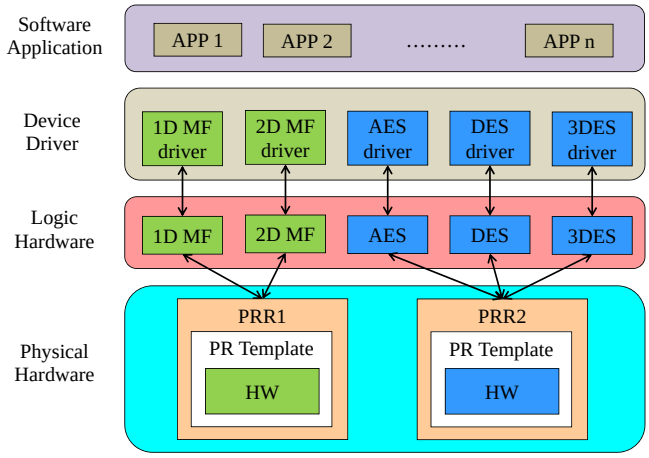


Fig. 4: Virtualizable and hierarchical design

in the ACES. In the design of the device driver, the related *Application Programming Interfaces* (APIs) are provided for software applications to interact with the software accessible registers in the hardware/software interface component. As a result, through the APIs, a software application can easily access the reconfigurable hardware modules.

According to the requirements of our target applications, a software application need to access only one of the filter functions and one of the cryptographic functions at a time instant. When the traditional embedded system design method is used to support multiple functions, all the corresponding hardware designs need be configured in the system at design-time. Although this method enables system performance to be significantly enhanced, because of hardware acceleration; however, system adaptation and the utilization of hardware resources also degrade. To solve the above problem, the ACES is thus realized as a virtualizable and hierarchical design, as shown in Figure 4. Here, besides the software application layer and the device driver layer, the hardware design of the ACES is divided into two layers, including the logic hardware layer and the physical hardware layer.

Through the partial reconfiguration technique, the required filter and cryptographic functions can be dynamically configured in `PRR1` and `PRR2`, respectively. This also indicates that, only two hardware functions are configured in the system at a time instant. The virtualization layer, including the logic hardware layer and the physical hardware layer, abstracts the real hardware characteristics. Furthermore, in the ACES, all the device drivers corresponding to the reconfigurable hardware modules are also provided for software applications. From the viewpoints of software applications, the ACES can support all the hardware functions, even though not all hardware functions are configured in the system at the same time. As a result, through the virtualizable and hierarchical deign, system adaptation and the utilization of hardware resources can be further enhanced.

3.3 Adaptation Policy

In our current implementation, the system adaptation mechanism is realized as a software program executed on the microprocessor. The ACES design contains two types of adaptable hardware functions, including the filter function and the cryptographic function.

The hardware filters are responsible for reducing the effects of noises in the source images. The quality of source images is classified into different levels according to the *Signal-to-Noise Ratio* (SNR). Different hardware filters are individually associated with their corresponding efficiencies for the reduction of noise in the images. With the increase in the effects of noise, the ACES can dynamically configure the corresponding hardware filter to reduce the effects of noise in the source images. In contrast, the ACES can reconfigure the blank module to improve system performance, when the effects of noise in the source images decrease.

The cryptographic functions are responsible for supporting the service of *Secure Socket Layer* (SSL). When a client makes a request for data transfers, the ACES thus negotiates with it to adopt the same cryptographic function for ensuring the security of data transfers on the network. If the negotiation succeeds, the ACES then configures the requested cryptographic function into the FPGA to adapt itself to different security requirements. Additionally, when no requests for data transfers on the network are received, the ACES can also reconfigure the blank module to improve system performance and reduce power consumption. The related experiments will be discussed in Section 4.

4. Experiments and Analyses

To demonstrate the practicability of our proposed method, real applications are implemented in the ACES. In the following sections, we will introduce the experimental setup, the system resource analysis, the power consumption analysis, and the system performance analysis.

4.1 Experimental Setup

The ACES design was implemented on the Xilinx ML605 FPGA development board [13] with a Virtex-6 XC6VLX240T FPGA chip. A soft-core MicroBlaze microprocessor [14] at 100 MHz was integrated into the ACES design. Two hardware median filters, including one-dimensional (1D) median filter and two-dimensional (2D) median filter, and three cryptographic functions, including *Advanced Encryption Standard* (AES), DES, and triple DES, were also implemented in the ACES. Two different sized PR regions, namely a small `PRR1` and a large `PRR 2`, were implemented for the dynamic configuration of median filter functions and cryptographic hardware functions, respectively. As shown in Figure 5, the small `PRR1` configured with 2D median filter and the large `PRR2` configured with triple DES are highlighted for displaying the relative locations

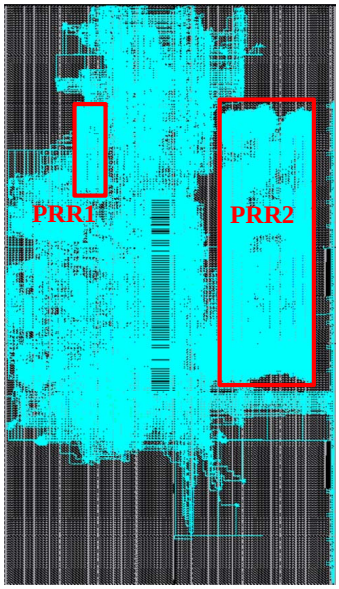


Fig. 5: FPGA implementation result

Table 1: Resource Usage

	#Slice registers	#Slice LUTs
1DMF	45	101
2DMF	730	1,795
AES	1,042	3,627
DES	3,955	6,152
3DES	11,457	18,312

1DMF: one-dimensional median filter. 2DMF: two-dimensional median filter. 3DES: triple DES.

in the implementation result of ACES. Further, a software solution was also implemented and executed on the host computer (Intel Core^{TM} i7-3770 3.40GHz, 32GB RAM) for the comparison with the ACES design.

In the experiments, a point target detection function called PMCE [15] was adopted as the main image processing application. Real-time 320 $\times$ 240 pixel images were captured from the camera for the application of point target detection, which were then encrypted using the cryptographic functions for data transfers on the network.

4.2 Resource Utilization

Compared to a conventional embedded design that requires all the five functions to be implemented and integrated into the system design, the ACES design can support all the five functions by implementing only two PR regions. The resource usages for the five hardware functions, including 1D median filter, 2D median filter, AES, DES, and triple DES, are given in Table 1.

To further compare with the conventional embedded system design, Figure 6 gives a comparison on the numbers of slices registers and those of slice LUTs required for supporting all the five functions. Experimental results show that the ACES needs at most 12,187 slice registers and 20,107 slice LUTs in terms of the Xilinx Virtex-6 XC6VLX240T FPGA. This presents the maximal resource usage by the reconfigurable modules of 2D median filter and triple DES. Compared to the conventional embedded system design, the ACES can reduce 29% of slice registers and 33% of slice LUTs in the Xilinx Virtex-6 XC6VLX240T FPGA. Furthermore, by using the hardware/software interface and the PR template, as described in Section 3.1, new user-designed hardware functions can be easily integrated into the ACES. This shows that, besides having efficient system scalability and adaptation, the ACES can also support a larger number of hardware functions by using the capability of hardware virtualization.

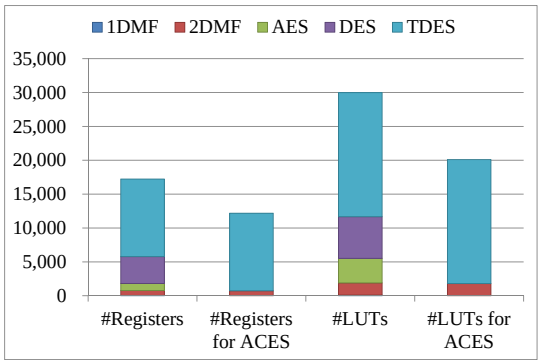


Fig. 6: Resource analysis

Table 2: Power consumption

	Dynamic (W)	Quiescent (W)	Total (W)
1DMF,DES	0.672	4.780	5.452
2DMF,AES	0.765	4.783	5.548
2DMF,3DES	1.020	4.791	5.812
PRR1BM,PRR2BM	0.594	4.778	5.372

1DMF: one-dimensional median filter. 2DMF: two-dimensional median filter. 3DES: triple DES. PRR1BM: blank module for PRR1. PRR2BM: blank module for PRR2.

4.3 Power Consumption

Besides supporting higher resource utilization as described as Section 4.2, the ACES design can also reduce power consumption. To perform the experiment on power consumption, the Xilinx XPower estimator [16] was used to measure the power consumption of the placed and routed netlists for different combinations of hardware functions in the ACES. Here, our measured results, including the dynamic power, the quiescent power, and the total power, in watt (W) are given in Table 2. Considering the worst case of using maximum power for each of the two PRRs, that is, 2D median filter in `PRR1` and triple DES in `PRR2`, the ACES requires 5.812 watt.

Table 3: Configuration Time

	Function	Time (ms)
PRR 1	PRR1BM	174
	1DMF	192
	2DMF	192
PRR 2	PRR2BM	2,009
	AES	2,227
	DES	2,227
	3DES	2,009

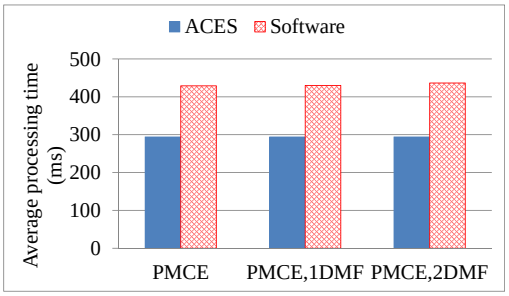
1DMF: one-dimensional median filter. 2DMF: two-dimensional median filter. 3DES: triple DES. PRR1BM: blank module for PRR1. PRR2BM: blank module for PRR2.

When the effects of noises in the source images decrease and no requests for data transfers on the network are received, the ACES can reconfigure the blank modules for `PRR1` and `PRR2` to reduce its power consumption. As shown in Table 2, compared to the ACES configured with 2D median filter and triple DES hardware functions, when the corresponding blank modules are configured in the ACES, the total power consumption can be reduced by 0.44 watt. This shows that, through system adaptation, the power consumption of the ACES can be further reduced at runtime, according to different environmental conditions. This feature also benefits the development of low-power embedded systems.

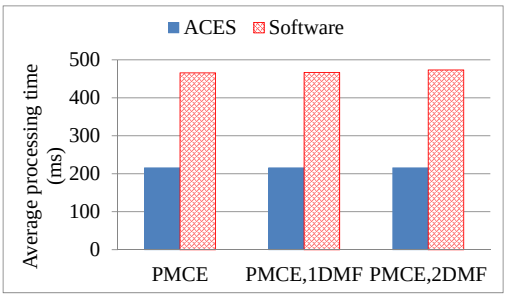
4.4 System Performance

Compared to the conventional embedded system design, for hardware function switching, the ACES contains an additional time overhead, that is, the configuration time. The configuration time for each hardware function is given in Table 3. We can observe that, the configuration times for the hardware functions configured in `PRR1` are approximately the same, while that for the hardware functions configured in `PRR2` are also approximately the same. This is because the configuration time is directly proportionate to the bitstream size, which in turn is directly proportionate to the size of the PR region. To reduce the reconfiguration time overhead, in this work, the configuration prefetch approach [17] is also applied to the ACES.

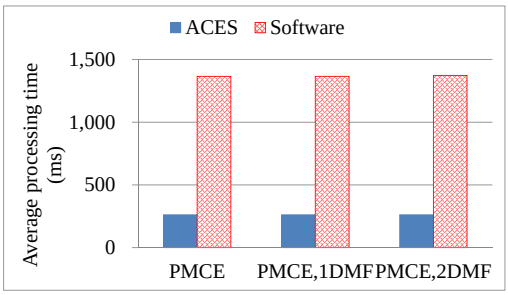
To further analyze system performance, 100 to 1,000 real-time 320 $\times$ 240 pixel images were applied to the software solution and the ACES design. Figures 7(a), 7(b), and 7(c) show the average time to process an image frame by using AES, DES, and triple DES, respectively. Here, each cryptographic function was also individually cooperated with three different image processing applications, including the pure PMCE function, the PMCE function with 1D median filter, and the PMCE function with 2D median filter. We can observe that, compared to the software solution, the ACES can efficiently enhance system performance. According to the experimental results, the ACES can accelerate by up to 1.5x, 2.2x, and 5.2x the times required by using the



(a) Average processing time using AES



(b) Average processing time using DES



(c) Average processing time using triple DES

Fig. 7: Average processing time using AES, DES, and triple DES

software solutions, when the pair of AES and the 2D median filter, that of the DES and the 2D median filter, and that of the triple DES and the 2D median filter, respectively, are configured in the FPGA.

For the current ACES implementation, the data transfers between the microprocessor and the cryptographic function are mainly through the software accessible registers. In order to further analyze the execution process for each cryptographic function, the average time per register access for different numbers of registers were measured as illustrated in Figure 8. We can observe that the average time per register access gradually becomes a constant (196 ns). This is because the operation of register access is mainly through the system bus, and thus the measured time also contains the initialization time over the bus. Considering the number of register access for each cryptographic function and the experimental results as shown in Figure 8, the pure

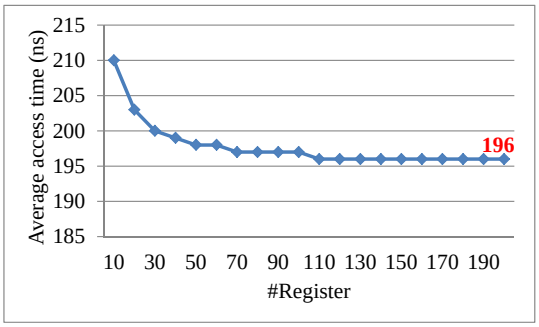


Fig. 8: Average access time per register

execution times required by using the AES, DES, and triple DES can be further calculated. In fact, the pure execution times required by using the AES, DES, and triple DES only take around 7%, 19%, and 29%, respectively, of the measured time as shown in Figures 7(a), 7(b), and 7(c). The experimental results show that, when the time per register access is not considered, the ACES can accelerate by up to 26.5x the time required by using the software solution. This also demonstrates that, the ACES design leverages the architectural features of FPGAs efficiently, so that system performance can be enhanced significantly.

5. Conclusion

The proposed adaptive cryptographic and embedded system (ACES) design can not only provide high-performance computing by using the architectural advantages of the FPGA device, but also can adapt its functionalities to different system requirements. Through the hardware virtualization technique in the ACES, system adaptation and the utilization of hardware resources can be further enhanced. Experiments with real applications have also demonstrate that ACES can accelerate by up to 26.5x the processing time required by using the software solution. Further, through the ability of system adaptation, the power consumption of the ACES can also be reduced at runtime, according to different environmental conditions.

References

[1] C. Efstratiou, K. Cheverst, N. Davices, and A. Friday, "An architecture for the effective support of adaptive context-aware applications," in *Proceedings of the 2nd International Conference on Mobile Data Management (MDM 2001)*, January 2001, pp. 15–26.

[2] S.-J. Ghim, Y.-I. Yoon, and J.-W. Choe, "A reflective approach to dynamic adaptation in ubiquitous computing environment," in *Proceedings of the International Conference on Networking Technologies for Broadband and Mobile Networks (ICOIN 2004)*, February 2004, pp. 75–82.

[3] J.-Z. Sun, "Adaptive determination of data granularity for QoS-constraint data gathering in wireless sensor networks," in *Proceedings of Symposia and Workshops on Ubiquitous, Autonomic and Trusted Computing (UIC-ATC)*, June 2009, pp. 401–405.

[4] S. Hallsteinsen, K. Geihs, N. Paspallis, F. Eliassen, G. Horn, J. Lorenzo, A. Mamelli, and G. Papadopoulos, "A development framework and methodology for self-adapting applications in ubiquitous computing environments," *Journal of Systems and Software*, vol. 85, no. 12, pp. 2840–2859, December 2012.

[5] J. Zhou, E. Gilman, J. Palola, J. Riekki, M. Ylianttila, and J. Sun, "Context-aware pervasive service composition and its implementation," *Personal Ubiquitous Computing*, vol. 15, no. 3, pp. 291–303, March 2010.

[6] Xilinx Inc., "Partial Reconfiguration User Guide - UG702," January 2012.

[7] T. Wollinger and C. Paar, "How secure are FPGAs in cryptographic applications," in *Proceedings of the 13th IEEE International Conference on Field Programmable Logic and Applications (FPL03)*, August 2003, pp. 1–3.

[8] A. Lagger, A. Upegui, E. Sanchez, and I. Gonzalez, "Self-reconfigurable pervasive platform for cryptographic application," in *Proceedings of 16th IEEE International Conference on Field Programmable Logic and Applications (FPL06)*, August 2006, pp. 777–780.

[9] N. Mentens, K. Sakiyama, L. Batina, I. Verbauwhede, and B. Preneel, "FPGA-oriented secure data path design: implementation of a public key coprocessor," in *Proceedings of the 16th IEEE International Conference on Field Programmable Logic and Applications (FPL06)*, August 2006, pp. 133–138.

[10] R. Laue, O. Kelm, S. Schipp, A. Shoufan, and S. Huss, "Compact AES-based architecture for symmetric encryption, hash function, and random number generation," in *Proceedings of the 17th IEEE International Conference on Field Programmable Logic and Applications (FPL07)*, August 2007, pp. 480–484.

[11] Xilinx Inc., "LogiCORE IP XPS HWICAP - DS586," June 2011.

[12] C.-H. Huang and P.-A. Hsiung, "Model-based verification and estimation framework for dynamically partially reconfigurable systems," *IEEE Transactions on Industrial Informatics (TII)*, vol. 7, no. 2, pp. 287–301, May 2011.

[13] Xilinx Inc., "ML605 Hardware User Guide - UG534," October 2012.

[14] ——, "MicroBlaze Processor Reference Guide, Embedded Development Kit - UG081," January 2012.

[15] C.-H. Huang, "An FPGA-based point target detection system using morphological clutter elimination," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2013, pp. 2436–2439.

[16] Xilinx Inc., "XPower Estimator User Guide - UG440 (v13.4)," January 2012.

[17] S. Banerjee, E. Bozorgzadeh, and N. Dutt, "Physically-aware HW-SW partitioning for reconfigurable architectures With partial dynamic reconfiguration," in *Proceedings of the 42nd ACM/IEEE Design Automation Conference (DAC'05)*, Jun. 2005, pp. 335–340.