Defence Science Journal, Vol. 62, No. 1, January 2012, pp. 25-31, DOI: 10.14429/dsj.62.1438

A Flexible Crypto-system Based upon the REDEFINE Polymorphic ASIC Architecture

Ganesh Garga[*], Saptarsi Das[**], S.K. Nandy[**], Ranjani Narayan[*], Chandan Haldar[*], Maheshkumar P. Jagtap[#], and Siba Prasad Dash[#]

[*]*Morphing Machines Pvt Ltd., Bengaluru, India*
[**]*Indian Institute of Science, Bengaluru, India*
[#]*Advanced Numerical Research and Analysis Group, Hyderabad, India*
E-mail: ganesh.garga@morphingmachines.com

The highest levels of security can be achieved through the use of more than one type of cryptographic algorithm for each security function. In this paper, the REDEFINE polymorphic architecture is presented as an architecture framework that can optimally support a varied set of crypto algorithms without losing high performance. The presented solution is capable of accelerating the advanced encryption standard (AES) and elliptic curve cryptography (ECC) cryptographic protocols, while still supporting different flavors of these algorithms as well as different underlying finite field sizes. The compelling feature of this cryptosystem is the ability to provide acceleration support for new field sizes as well as new (possibly proprietary) cryptographic algorithms decided upon after the cryptosystem is deployed.

Keywords: REDEFINE, advanced encryption standard, AES, elliptic curve cryptography, ECC, cryptographic protocols

1. INTRODUCTION

New cryptographic standards are firmly moving towards recommending different cryptographic algorithms for different security functions, all of which are usually needed within the same communicating device. With quality-of-service based communications proliferating, there is also the need to be able to rapidly tune the performance of these accelerators according to communication medium conditions. Achieving this flexibility by using a number of 'fixed function' accelerators available today is not a scalable approach, as changing cryptographic standards could easily render the security functionality out of date, requiring a respin of the entire product. Such circumstances make it an attractive proposition to have a unified cryptographic accelerator that can accelerate at least a canonical set of existing cryptographic algorithms, while still providing some means to support performance tuning as well as new cryptographic algorithms. Of course, it is vital not to lose too much of the high performance that makes specialised accelerators attractive, in the process.

In communications where the highest possible security is absolutely necessary, for instance in national security related communications, moving away from standards based algorithms and devising custom cryptographic algorithms is an option that is desirable and often practiced. It may also be desired to change the crypto algorithm in use at a particular time on-the-fly. If high performance is needed together with proprietary algorithms, one simply cannot obtain a solution in currently available cryptographic accelerators. Designing a unified platform for this purpose requires a careful shift from the usual algorithm-focused way of designing crypto-accelerators.

In this paper, we present the REDEFINE polymorphic ASIC architecture[1,15] as a suitable platform for flexible crypto-accelerators. The overall architecture of the REDEFINE platform is illustrated in Fig. 1. In REDEFINE, specialised hardware units are replaced by more basic hardware units that can be dynamically recomposed to provide different functionalities required to accelerate higher-level applications. Applications described in a high level language, namely C, are broken down into application substructures (called HyperOps) that are then mapped onto a set of basic processing elements (compute elements (CEs) in REDEFINE) interconnected through a NoC. The REDEFINE platform includes its own compiler[1,2], which performs this decomposition in an extremely efficient, hardware-aware manner. Since, application synthesis in REDEFINE is from a high level specification in C, new applications as well as application enhancements decided upon after deployment can be easily realised within the REDEFINE framework, by simply creating a new software (functional) description for it.

Further, the REDEFINE framework allows the customisation of the basic processing units within the architecture, in order to support special instructions accelerating the common low-level operations occurring in all the applications. Processing units thus added are called custom functional units (CFUs). This makes it possible to integrate ASIC-like speed of execution, with the flexibility coming from being able to describe applications in C, which is an ideal combination for unified accelerator for different existing as well as proprietary cryptographic algorithms.

Advanced encryption standard (AES) and elliptic curve cryptography (ECC) algorithms are accelerated on the

Received 5 December 2011, online published 23 January 2012

REDEFINE architecture framework. These two algorithms are chosen since they are instances of two distinct types of cryptographic algorithms- AES involves a large number of bitwise operations, while ECC requires computations over finite fields. We show how REDEFINE can accelerate finite-field computations of arbitrary sizes, which is the most demanding requirement for a flexible cryptosystem.

2. ACCELERATING ADVANCED ENCRYPTION STANDARD ON THE REDEFINE ARCHITECTURE FRAMEWORK

The advanced encryption standard (AES) encrypts/decrypts 128-bit data blocks using 128-bit, 192-bit or 256-bit keys. AES accelerators can be designed to meet a large number of possible area-performance points. When designing a flexible cryptosystem, the challenge is to make use of some of the hardware resources dedicated to obtain a high performance AES implementation, in order to accelerate other cryptographic algorithms. In REDEFINE, we solve this problem by realising the AES application in the following manner:

- The basic processing elements (CEs) are augmented with a CFU that encrypts/decrypts 128-bit data blocks according to the AES algorithm with 128-bit, 192-bit or 256-bit keys. The CFU is designed to provide a moderate throughput.
- The desired level of performance is achieved purely through application software - by writing application code to generate HyperOps that span an appropriate number of such CFUs, thus bringing them together to process the input data. An example mapping of the AES operation onto the REDEFINE architecture framework is shown in Fig. 1, which also serves to visually describe the REDEFINE architecture. A snippet of the corresponding code listing is given in listing 1.
- When the AES HyperOp is smaller than the maximum size of the fabric, it is possible to launch HyperOps of other applications onto the free CEs, thus allowing a throughput driven overlap of different simultaneously running crypto algorithms.

```
while ( (*no_of_blocks) != 0)
{
   start_aes(command_word, store_address, 0, &dummy_output50);
   start _aes(command_word, (store_address+4), &dummy_output51);
   start_aes(command_word, (store_address+8), &dummy_output52);
   start _aes(command_word, (store_address+12), &dummy_output53);
   start _aes(command_word, (store_address+16), &dummy_output54);
   start _aes(command_word, (store_address+20), &dummy_output55);
   start _aes(command_word, (store_address+24), &dummy_output56);
   start _aes(command_word, (store_address+28), &dummy_output57);
   start _aes(command_word, (store_address+32), &dummy_output58);
   start _aes(command_word, (store_address+36), &dummy_output59);
   load_data_0(*(load_start_address + i), *(load_start_address + i + 1),
   *(load_start_address + i + 2), &dummy_output60);
   load_data_1(*(load_start_address + i + 3), &dummy_output61);
   .....
}
```

Listing 1. Code snippet for implementing the AES algorithm on REDEFINE. Here, 10 AES CFUs have been employed for the encryption/decryption task.

The REDEFINE compiler allows the application writer to insert directives which place the different CFU calls (*start_aes* in listing 1) onto different CEs, where they can all be executed in parallel. The throughput obtained for the AES application is listed in Table 1.

3. ACCELERATING ELLIPTIC CURVE CRYPTOGRAPHY ON THE REDEFINE ARCHITECTURE FRAMEWORK

Elliptic curve cryptography (ECC)-based algorithms basically operate on a subset of points over an elliptic curve. The coordinates of these points are defined over an underlying finite field or Galois field. The central operation in all ECC-based protocols is the point scalar multiplication operation[5,10,11]. This operation can be broken down to operations at lower levels as described in Fig. 2. The performance of all ECC-based schemes is finally determined by the performance of multiplication in the underlying finite field[8].

The point addition, point doubling and point multiplication algorithms are described as high level C applications on the REDEFINE framework.This makes it easy to optimise these algorithms depending on the curve and base point chosen.

To generate the results given in Table 1, we have used the Montgomery algorithm for the random base point case, and the fixed base comb method for the fixed base point case[14].

Only the lowest layer of operations, namely the

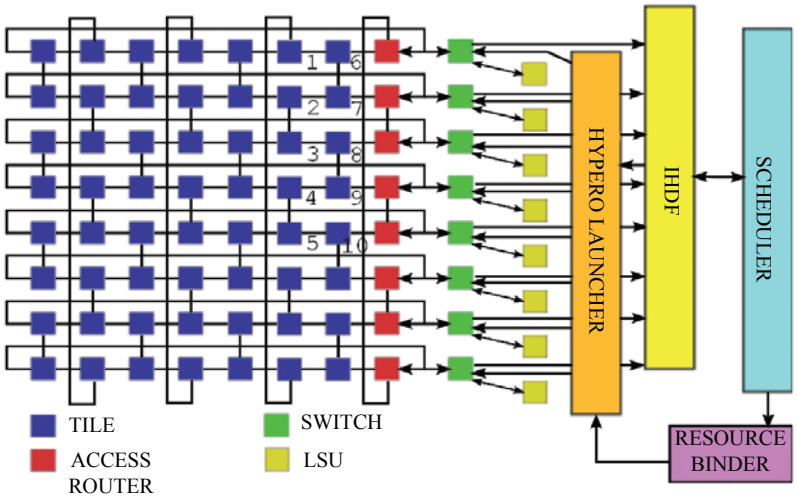

Figure 1. Architecture of the REDEFINE platform. The numbers show the mapping of data blocks onto CEs while running 10 AES instances in parallel. The CEs closest to the load store units (LSUs) are chosen to minimise load/store delays.

Table 1. Performance numbers for AES and ECC on the REDEFINE crypto accelerator

Algorithm	Performance
AES-128 (using 10 CEs of the fabric)	3 Gbps
ECC(over GF(2^{163})) random base point, random curve coefficients	350 ops/s
ECC(over GF(2^{163})) fixed base point, random curve coefficients	700 ops/s

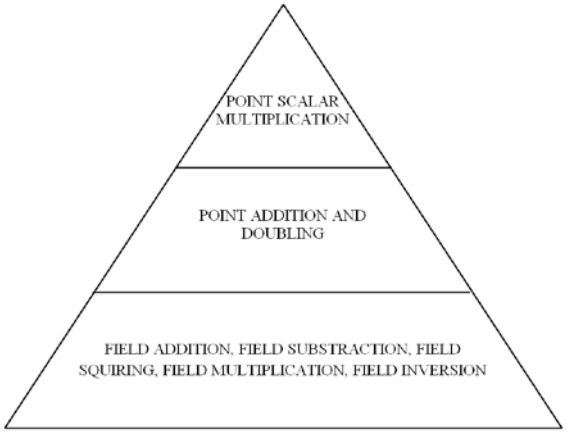

Figure 2. The hierarchy of operations in the elliptic curve point scalar multiplication operation.

finite field operations is accelerated by using CFUs. Thus, we use two CFUs, one for GF (2^m) squaring of a 32-term polynomial and one for polynomial multiplication of two 16-term polynomials with binary coefficients. These CFUs can be used in a scalable manner, i.e., for different field sizes, the higher level application only needs to operate an appropriate number of these CFUs. It is the reduction operation that poses a problem with respect to the flexibility of the cryptosystem, and hence, this issue is considered in greater detail in subsequent sections.

The general purpose method of performing reduction is repeated subtractions (or equivalent), which is too slow to be of use in practice, especially for the finite field sizes of cryptographic importance. To circumvent this problem, fast reduction methods have been developed[3,9]. However, these schemes assume the modulus of the finite field to be a priori decided, and this is where the flexibility of the crypto system wrt different field sizes becomes limited. In REDEFINE, it has been found possible to obtain a 'general purpose' implementation of the fast reduction methods, that can scale to different moduli defined at runtime. The complete method is explained in a related publication[4] . Some of the sections are reproduced here to illustrate the essential concepts.

The fast-reduction method used in the crypto system realisation is Barrett reduction. The underlying finite field considered is GF(2^m), which is a popular choice due to the simplicity of addition and subtraction in that field. Barrett reduction is based on precomputing a constant dependent on the modulus, in order to accelerate the reduction process. Knezevic[3], *et al*. showed that this precomputation step can be eliminated, if the moduli are of some special form. The special form dictates that the degree (say l) of the second most significant term in the irreducible polynomial acting as the modulus should satisfy $l < (n/2)$, where n is the degree of the irreducible polynomial. Equation (1) shows the reduction process in this case[3].

$$Q_1(x) = C_0(x)\ div\ x^m = C_{h,0}(x);\ Q_2(x) = Q_1(x)\,P(x)$$
$$Q_3(x) = Q_2(x)\ div\ x^m = C_{h,0}(x)\,(x^m + x^k + \cdots + x^p + 1)\ div\ x^m$$
$$= C_{h,0}(x) + C_{h,1}(x)$$
$$R_1(x) = C_0(x)\ mod\ x^m = C_{l,0}(x)$$
$$R_2(x) = Q_3(x)\,P(x)\ mod\ x^m = Q_3(x)\,(x^k + \cdots + x^p + 1)\ mod\ x^m$$
$$= C_{h,0}(x)\,(x^k + \cdots + x^p + 1) mod\ x^m + C_{h,1}(x)(x^k + \cdots + x^p + 1) mod x^m$$
$$R(x) = R_1(x) + R_2(x) = C_{h,1}(x)\,(P(x) - x^m) + C_{l,1}(x) \qquad (1)$$

From Eqn (1) it is evident the reduction process requires multiplication of m-bit polynomials. Note that, the module and division operations in the two methods translate to partitioning of the polynomials into lower and higher half and therefore do not require any arithmetic operation.

3.1 Multiplication Operations in Reduction

From Eqn (1) we observe that multiplications of the form $C(x)(x^k + \cdots + x^p + 1)$ form the core of the computations. Therefore, it is necessary to accelerate these multiplications in order to perform fast-reduction. It should also be noted that the only other operations involved in reduction are addition over GF(2^m). Since, there is no carry involved in addition, addition of two m-bit polynomials which span more than one word in a w-bit architecture can be realised as ceil(m/w) w-bit XOR operations. Multiplication on the other hand requires multi-word shift and accumulation of results. Consider the two polynomials $C(x)$ and $P(x)$ of degree m and k respectively. These polynomials can be represented in a w-bit architecture as a collection of m_c and m_p w-bit words respectively. Eqn (2) shows the representation.

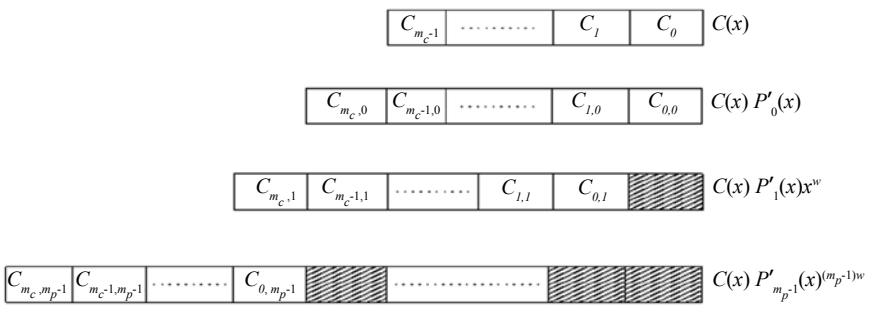

Figure 3. Arrangement of partial products.

$C(x) = \Sigma C_i(x)\, x^{iw};\ 0 <= i <= m_c - 1,$
where m_c = ceil(m/w)
$P(x) = \Sigma P_j(x)\, x^{jw};\ 0 <= j <= m_p - 1,$
where m_p = ceil(k/w) (2)

$C_i(x)\, x^{iw}$ and $P_j(x)x^{jw}$ denote the i-th and j-th words of the polynomials $C(x)$ and $P(x)$ respectively. The product of these two polynomials can be computed as follows:

$C'(x) = C(x)\, P'(x) = \Sigma C(x)\, P'_j(x)\, x^{jw};$
$0 <= j <= m_p - 1 = \Sigma(\Sigma C_i(x)\, x^{iw} P_j(x)\, x^{jw});$
$0 <= i <= m_c - 1$ (3)

A closer look at Eqn (3) reveals that computation of $C_i(x)$ $P_j(x)$ involves computations of the form $C_i(x)\, x^r$. Each of the individual words like $C'_{i,j}(x)$ (Fig. 3) in the product of the entire polynomial $C(x)$ and x^r can be computed as follows:

$C'_{i,j} = C_i << r \mid C_{i-1} >> (w\text{-}r))$ (4)

The individual words like $C_{i,j}(x)$ in the product of $C(x)$ and $P_j(x)$ can be expressed as given by Eqn (5).

$C'_{i,j} = x$ or over all r in $0{:}(w\text{-}1)(C_i << r \mid C_{i-1} >> (w\text{-}r)).p'_{j,r}$ (5)

Note that $p_{j,r}.\, x^r$ denotes the r-th term in the j-th word of the polynomial $P'(x)$ in Eqn (5). The operations of Eqn (5) can be repeated for each of the words in $P'(x)$ to compute the final result. Note that the product of $C(x)$ and each of the words in $P(x)$ is m_c +1 word wide. Hence forward we will refer to products of $C(x)$ with the individual words of $P(x)$ as 'partial products'. It should be noted that these (m_c + 1) word wide partial products need to be aligned to proper word boundaries before they can be added together to produce the final result. Figure 3 shows how the partial products are aligned.

3.2 A Modified Interleaved Galois Field Multiplier as a Hardware Assist for Reduction

A reduction method is only as fast as the underlying multiplication operations. Therefore it is obvious that polynomial multiplication kernels are the candidates for acceleration in a crypto-system. The simplest way of accelerating a $w \times w$-bit polynomial multiplication is to introduce a w-bit polynomial multiplier that produces $2w$-bit results. Therefore each word in the input polynomial $C(x)$ produces a pair of words and these pairs need to be added (i.e., XORed) with proper alignment to compute a partial product. In this subsection we propose a technique for combining the addition operations with the polynomial multiplications. Instead of considering one word of the polynomial $C(x)$ we focus on one word of the partial product (i.e. $C_{i,j}(x)$). It is evident from Eqn (5) that to produce $C_{i,j}(x)$ two words from the polynomial $C(x)$ and one word from $P(x)$ are necessary. Thus the intended operation can be described as a $2w \times w$-bit polynomial multiplication that produces a w-bit result. In this subsection we show that an Interleaved Galois Field (IGF) Multiplier[6] can be modified to support this type of multiplications as shown in Fig. 4. In a shift-and-add IGF multiplier, the multiplicand operand is successively left shifted and the multiplier operand is used to selectively accumulate the results of the left shift operations. The IGF multiplier always produces a reduced result. Reduction over large fields however, requires support for multiplication of polynomials where the result is kept unreduced. This can be achieved by setting the irreducible polynomial to all zeros. This is achieved by masking the irreducible polynomial input to each stage of the multiplier with a one bit control signal (Mode Select signal in Fig. 4). In order to emulate the operations described in Eqn (5) the Modified IGF (MIGF) multiplier inserts the $(w - r)$-th bit from the second multiplicand operand to the LSB of the first multiplicand at the r-th stage of the multiplier. This is enabled by introducing a single AND gate that drives the LSB of the shifted polynomial. As can be seen from Fig. 4, we use the inverted control signal to mask the $(w - r)$-th bit from the second multiplicand operand. This added hardware as shown inside the shaded rectangle in Fig. 4, enables the multiplier to perform two-word shift operations successively which in turn alleviates the need for adding the individual products of the multiplier to form the partial product. The flexibility of the proposed method is immediately apparent from the two facts: (i) the shift value r in Fig. 4 can take any value from 0 to 31. (ii) any two input words can be passed to an IGF multiplier by using appropriate C language statements.

3.3 Hardware Complexity of the Modified IGF Multiplier

As shown in Fig 5, we introduced a set of two-input AND gates in each stage of the MIGF Multiplier to enable two-word shift operations. In a w-bit instance of the multiplier, two sets of w two input AND gates are introduced. The first set of w two input AND gates are used for masking the irreducible polynomial input to the multiplier to zero. The second set of w two input AND gates are used for enabling two-word shift operation. This increase in hardware complexity is compensated by the significant reduction in the number of operations brought about by using this multiplier as a hardware assist for reduction.

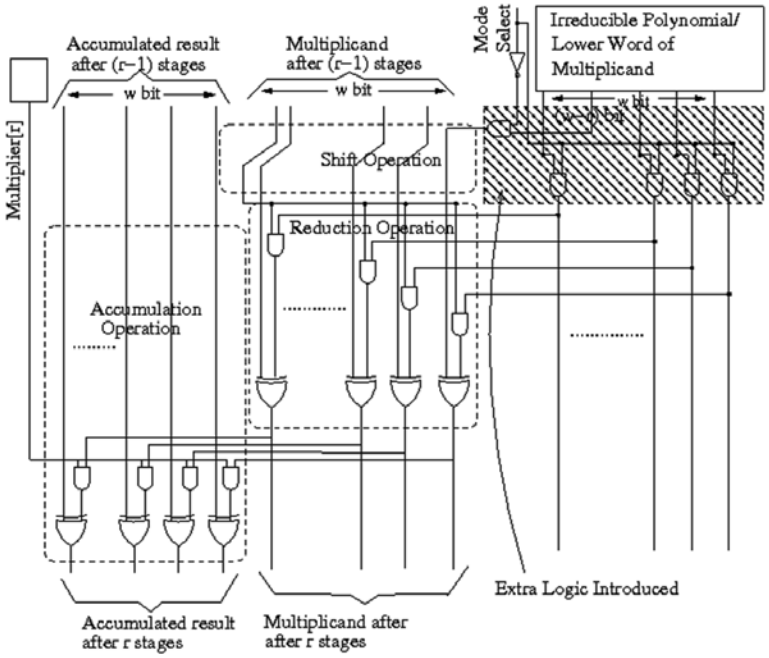

Figure 4. One stage of the modified IGF multiplier.

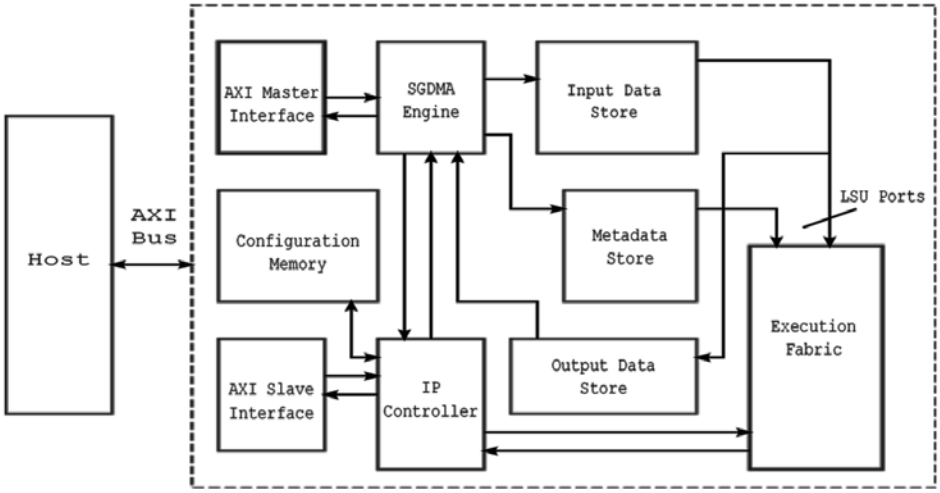

Figure 5. An example interface between the REDEFINE cryptosystem and the rest of a multi-processor system.

The proposed reduction method also leads to a reduction in the basic operation count for finite field reduction, but the associated material is not reproduced here due to space constraints.

4. PERFORMANCE OF THE REDEFINE-BASED CRYPTOSYSTEM

Table 1 details the performance obtained on the REDEFINE-based cryptosystem for the AES and ECC kernels. These results were obtained by running the application on a cycle accurate simulator of the REDEFINE platform described using bluespec system verilog (BSV). The operating frequency of the REDEFINE-based cryptosystem is assumed to be 400 MHz, deriving from synthesis results of the Verilog descriptions of the component modules using Synopsys Design Compiler, using 90 nm technology libraries. The results in Table 1 are comparable to the performance of individual accelerators for the respective algorithms available in the market. As another point of comparison, the OpenSSL implementation of the ECC operation (random base point) achieves around 350 operations/second on a 450 MHz UltraSPARC 2 processor[14]. While the speeds achieved are comparable to those listed in Table 1, the power dissipation is significantly higher due to the higher clock frequency. However, there is scope for achieving even more throughput by utilising the resources available in the REDEFINE architecture framework more fully. For instance, all the CEs available in the computation fabric can be dedicated to AES, in which case one would obtain an ultra-high throughput AES engine with multi-gigabit per second performance.

Using the tools available within the REDEFINE framework, one would be able to implement different complex load-sharing schemes across the CEs (based on the desired speed) without having to change any of the provided hardware. Similarly, the results for the ECC point scalar multiplication with a fixed base point have been obtained with only 256 points of storage. However, the REDEFINE framework can support a large (upto 4GB) size of addressable memory. By using more of the available memory for storing more pre- computed points, the throughput for the fixed base point ECC can be greatly increased.

5. INCLUDING THE REDEFINE CRYPTOSYSTEM AS A COPROCESSOR INSIDE A MULTIPROCESSOR SYSTEM

Author consider how the REDEFINE-based crypto system can be fit as a coprocessor inside a multiprocessor system, wherein the REDEFINE-based crypto system can be driven by one or more of the other processors in the overall system.

The overall system architecture is shown in Fig. 5, where an AXI bus is shown as connecting all of the entities in the system.

One can assume that the programming phase of the REDEFINE-based cryptosystem is distinct from the actual usage of the cryptosystem, as the crypto algorithms mapped onto REDEFINE are not likely to change frequently. Supporting non-streaming applications in the context of Fig. 5 is trivial. For streaming applications, it would be beneficial to make an application stay on the computation fabric for an indefinite amount of time, until end of streaming input is signaled by some external entity. It turns out, that due to the ability to program the REDEFINE-based cryptosystem in C, this feature can be obtained again without changing any hardware, by introducing the while (1) loop. A sample code snippet for a streaming application is shown in Listing 2.

```
while ( *streaming_end == 0)
{
  if ( ( *old_outputs_stored == 1)
  && ( *new_inputs_loaded == 1 ) )
  {
  (perform one iteration of the streaming application)
  }
}
```

Listing 2. A code snippet for implementing a streaming application on the REDEFINE cryptosystem.

The address locations referred to in listing 2 are explained below.

- *streaming_end:* This location indicates how long the

streaming application needs to be executed. The location is set when the 'end of streaming' is obtained from the host processor.

- *old_outputs_stored:* indicates if the outputs from the previous iteration have been stored to the shared memory.
- *new_inputs_loaded:* indicates if the inputs for the next iteration have been loaded into the local data memory of the cryptosystem.

6. CONCLUSIONS AND FUTURE WORK

This paper shows how all the challenges associated with constructing a crypto-accelerator capable of accelerating even 'run-time' defined crypto applications can be met by using the REDEFINE architecture framework as the overall platform for appropriately dividing applications into software and hardware portions. Specifically, a procedure to perform efficient and programmable finite field reduction is discussed. There are quite a few avenues for future work, among which two are listed here:

- For AES-like applications, there is the need to identify an appropriate load-sharing scheme across all the CEs in the fabric, so that the input workload is efficiently divided among all of them.
- Side channel attacks can be easily prevented on the REDEFINE platform, by running a dummy program alongside the main program, so that the electromagnetic radiations from the device get obfuscated. A mature strategy for achieving this is needed.

REFERENCES

1. Alle, Mythri; Varadarajan, Keshavan; Fell, Alexander; Ramesh Reddy C., Joseph, Nimmy; Das, Saptarsi; Biswas, Prasenjit; Chetia, Jugantor; Rao, Adarsh; Nandy, S.K. & Narayan, Ranjani. Redefine: Runtime reconfigurable polymorphic asic. *ACM Trans. Embed. Comput. Syst.*, 2009, 9:11:1–11:48.
2. Alle, Mythri; Varadarajan, Keshavan; Fell, Alexander; Nandy, S. & Narayan, Ranjani. Compiling techniques for coarse grained runtime reconfigurable architectures. *In* Reconfigurable computing: Architectures, tools and applications, edited by Jrgen Becker, Roger Woods, Peter Athanas, and Fearghal Morgan. Springer Berlin/ Heidelberg, *Lecture Notes in Computer Science*, 2009, **5453**, pp. 204-15.
3. Barrett, Paul. Implementing the rivestshamir and adleman public key encryption algorithm on a standard digital signal processor. *In* Advances in Cryptology *CRYPTO 86*, edited by Andrew Odlyzko, Lecture Notes in Computer Science, Springer Berlin/Heidelberg, 1987, **263**, pp. 311-23.
4. Das, Saptarsi; Varadarajan, Keshavan; Garga, Ganesh; Mondal, Rajdeep; Narayan, Ranjani & Nandy, S.K. A method for flexible reduction over binary fields using a field multiplier, *SECRYPT 2011,* Seville, Spain, 2011, pp. 50-58.
5. Eberle, Hans; Gura, Nils; Shantz, Sheueling Chang & Gupta, Vipul. A cryptographic processor for arbitrary elliptic curves over GF(2^m). Technical report no: SMLI – TR – 2003 – 123. Mountain View, CA, USA, 2003
6. Hinkelmann, Heiko; Zipf, Peter; Li, Jia; Liu, Guifang & Glesner, Manfred. On the design of reconfigurable multipliers for integer and galois field multiplication. *Microprocessors and Microsystems - Embedded Hardware Design*, 2009, **33**(1), 2-12.
7. National Institute of Standards and Technology, Information Technology Laboratory. http://csrc.nist.gov/ publications/fips/fips197/fips 197.pdf. (Accessed on 23 June 2010)
8. Karatsuba, A. & Ofman, Yu. Multiplication of multidigit numbers on automata. *Soviet Physics—Doklady*, 1963, **7**(7), 595–96.
9. Knezevic, Miroslav; Sakiyama, Kazuo; Fan, Junfeng & Verbauwhede, Ingrid. Modular reduction in GF(2^n) without pre-computational phase. *In* WAIFI edited by Joachim von zurGathen, Jose´ Luis Imana, and Cetin Kaya Koc. *Lecture Notes in Computer Science*, Springer, 2008, **5130**, pp. 77-87.
10. Peter, Steffen; Langendorfer, Peter & Piotrowski, Krzysztof. Flexible hardware reduction for elliptic curve cryptography in GF(2^m). *In* DATE edited by Rudy Lauwereins and Jan Madsen, ACM, 2007. pp. 1259–264.
11. Satoh, Akashi & Takano, Kohji. A scalable dual-field elliptic curve cryptographic processor. *IEEE Trans. Comp.*, 2003, **52**, 449-60.
12. Hankerson, D.; Menezes A. & Vanstone, S.A. Guide to Elliptic Curve Cryptography, Springer-Verlag, 2004.
13. REDEFINE Crypto Core Data Sheet, Version 1.0, 2010, Morphing Machines, Bangalore, India.
14. Gupta, Vipul; Gupta, Sumit & Chang, Sheueling. Performance analysis of elliptic curve cryptography for SSL.WiSe '02, September 2002, Atlanta, Georgia, USA. pp. 87-94.

Contributors

Mr Ganesh Garga has obtained his MSc(Engg.) from the Indian Institute of Science (IISc), Bengaluru, in 2009. He is currently working as a Senior Member of Technical Staff at Morphing Machines Pvt. Ltd, Bengaluru, India. His research interests area: Cryptographic processors and wireless baseband systems.

Mr Saptarsi Das received his BE from Jadavpur University, Kolkata in 2007 and MSc from IISC, Bengaluru, in 2011. He is currently pursuing his PhD from IISc, Bengaluru. His research interests include: Applied cryptography and reconfigurable computing.

Dr SK Nandy has received MSc (Engg.) in Computer Science and Engineering in 1986 and the PhD (Computer Science and Engineering) from IISC, Bengaluru, in 1989. He is currently working as a Professor in the Supercomputer Education and Research Centre, IISc, Bengaluru. He has over 150 publications in International Journals, and Proceedings of International Conferences. His research interests includes: Low power and high performance embedded systems on a chip, VLSI architectures for reconfigurable systems on chip, and architectures and compiling techniques for heterogeneous many core systems.

Dr Ranjani Narayan obtained her PhD from IISC, Bengaluru, in 1989. Currently she is the CTO of Morphing Machines, Pvt Ltd, Bengaluru. She has many publications in Journals and proceedings of International Conferences to her credit. Her research interests include: Processor architectures, heterogeneous multi-cores architectures, embedded SoCs, and reconfigurable silicon cores.

Dr Chandan Haldar received hid BTech (Honors) and MTech from Indian Institute of Technology, Kharagpur and PhD from IISc, Bengaluru. He is currently working as a Managing Director of Morphing Machines Pvt Ltd, Bengaluru. He is also Chairman and Chief Scientist at Terra Incognitus Systems Research Alliance Pvt Ltd (TISRA). He is an alumnus of the Senior Executive Programme at the London Business School as Aditya Birla Scholar, and a Senior Member of the ACM, IEEE, and IEEE Computer Society.

Mr Maheshkumar P. Jagtap has obtained BE (Electronics) from University of Poona. He is currently working as a Scientist F at Advanced Numerical Research and Analysis Group (ANURAG), Hyderabad. His research interests are: Multi-core processor architectures, multi-core processor based system, flexible cryptographic processor architecture, reconfigurable computing system on chip, and network processor architecture for security and high performance applications.

Mr Siba Prasad Dash has obtained BE(Electronics and Telecommunication) from North Maharashtra University. He is currently working as a Scientist C in the ANURAG, Hyderabad. His research interests are: Multicore processor, flexible cryptography processor architecture, architecture for secure routers and switches.