Cryptographic Hardware Architecture

Savithri Venkatachalapathy
Department of Electrical & Computer Engineering,
Oregon State University, Corvallis, Oregon 97331 -USA.
E-mail: venkasav@ece.orst.edu
May 22, 2002

Abstract— **In this paper we have tried to present a descriptive description of Cryptographic Hardware. Cryptography itself is a math that provides the security in many technologies, including all digital certificate and encryption systems. Either Software or Hardware implementations of cryptography can be used. The Hardware implementation is secure, faster, but comes at an exorbitant price. To this extent the current trends in Hardware implementations have a promising future for cost effective hardware crypto-systemss. In this regard we try to give a summary of Cryptography Processor Architecture, Dynamic implementation of one of the AES standards. For this, we have chosen the Serpernt Block Cipher algorithm. And finally the comparision of the performance of FPGA implementation of five standard AES algorithms and determine their suitability for FPGA implementation.**

KEYWORDS: FPGA, AES, SERPENT BLOCK CIPHER, RC6, MARS, RIJNDAEL, TWOFISH, DSRCP.

I. INTRODUCTION

With the advancements in communication technologies and the indomitable growth of Internet has led to a different mode of bussiness and commerce:E-commerce. Along with this, came the requirement to have secured, portable and energy-efficient connections between the participating entities. To this end the development of Cryptographic algorithms namely RSA, Rijndael, Serpent Cipher etc have eventually created a standard for the cryptographic systems, called the AES standard. We have tried to summarise some of the algorithms and how they can be implemented in Hardware and the performance evaluation of the same.

The paper is organised as follows. Section 1 gives an introduction. Section 2 talks about the Cryptography Processor Architecture with emphasis on domain-specific reconfigurable processing for asymmetric cryptographic applications. Section 3 talks about the Dynamic FPGA implementation of the Serpent block cipher algorithm. Section 4 compares the performance of the FPGA implementation of the five AES algorithms. Section 5 summarises the FPGA implementation and performance evaluation of a hypothetical processor based on an ingenious algorithm. Section 6 draws a conclusion to this scientific chronicle which gave a sojourn of Cryptographic Hardware implemntations. Section 7 is the references and bibliography.

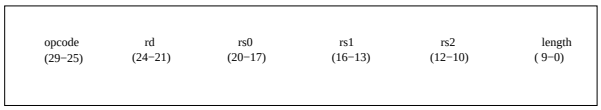
II. CRYPTOGRAPHIC PROCESSOR ARCHITECTURE

The constarint of using sopftware -based techniques in order to achieve the algorithm agility required to maintain compatibility resticts to only providing secure communications with compatible systems. The software approaches are slow, but energy efficient. Hardware based implementations on the other hand are fast, energy and computationally efficient but they are limited on only one type of asymmetric cryptographic algorithm.In this context, a new Domain Specific Reconfigurable Cryptographic Processor (DSRCP) is implemented as described in [4].

In conventional reconfigurable applications such as Field Programmable Gate Arrays (FPGAs). The architecture goals of the device are to provide a large number of small, yet powerfull logic cells, embedded within a flexible programmable interconnect. The overheard associated with making such a general purpose computing device ultimately limits its energy efficiency, and hence its applicaion is in energy-constrained environments. A conventional reconfigurable logic device attempts to cover as much space as possible given its architectural constraints in terms of technology and logic/ routing resources. This results in a considerable amount of overhead that is not necessary given a specific subset of functions. The DSRCP differs from conventional reconfigurable implementations in that its reconfigurability is limited to the subset of functions, called a domain required for asymmetric cryptography.This domain requires only a small set of configurations for perofrming all of the required operations over all possible problem families as defined by the IEEE P1363. As a result the configuration overhead is smaller in terms

of performance, energy efficiency, and reconfiguration time. The Instruction Set Architecture (ISA) of the DSRCP sticks to the IEEE P 1363 description document. It also has other required arithmetic functions is tabulated in order to determine the ISA for the required processor. The resulting instruction format of the DSRCP is a 30-bit word partitioned as shown in Figure 2.

Figure 1: Instruction word of DSRCP.



Architecture of the DSRCP: The processor consist of four main architectural blocks: the global controller and microcode ROMs, the I/O interface, the reconfigurable datapath, and an embedded SHA-1 hash function engine. The inclusion of a hash gfunction was desirable as the key derivation primitives contained wiothin the IEEE P 1363 call for this functionality. DSRCP uses both hardwired and microcode ROM-based control functions. This multi-tiered approach is required as various instructions within the DSRCPs ISA are implemented using other instructions within the ISA.

The microcode approach is chosen due to its simplicity and extensibility as modifications and enhancements of the ISA can be accomplished with a minimal amount of design effort by modifying the microcode ROM contents. The global controller is responsible for disabling unused portions of the circuitryin order to eliminate any necessary switched capacitance. The slowdown strategy is dictated by the current witdh of the processor and enables the datapath to be shutdown in 32 32-bit increments. SET_LENGTH (length) is used to set the current width of the processor. All operands accessed and operated by the datapath are assumed to be the size of the current width of the processor., as set by the SET_LENGTH. This length is used by the control logic to determine the number of iterations that need to be performed by the various operations.

I/O Interface of the DSRCP: Operands used within the processor can vary in size from 8 - 1024 bits, requiring the use of a flexible I/O interface that allows the user to transfer data to/from the processor in a very efficient manner. A 32-bit interface is used which is very well suited to existing processors annd systems which are predominently built upon 32-bit interfaces. The choice enables fast operand transfer onto and off of the processor, requiring at most 32 cycles to transfer the largest possible operand.

Reconfigurable Datapath: The primary componennt of the DSRCP is the reconfigurable datapath, which consists of four major functional blocks an eight word register file, a fast adder unit, a comparrator unit, and the main reconfigurable computation unit. The dayttapath is implemented using a very efficient bitsliced implementation in order to minimize its size and the corresponding wiring capacitance of its control signal generation/distribution. The registetr file is chosen to be eight words as it is the minimum number required to implement all of the functions of the datapath. The number of read and write ports within the register file is dictated by the requirement to be able to perform single cycle, two operand instructions which generate a write-back value. In certain cases two write ports could have proved useful, but the infrequency of the operation does not merit the additional overhead. The fast adder is capable of adding/subtracting two n-bit operands in four cycles using the hybrid carry-bypass and carry-select technique nd optimised for a bitsliced implementation. The unit features a local register to store the previous sum result, a feature thatr s used in modular addition/subtraction and inversion routines.The adder unit can also right shift its result, as required by the modular inversion algorithm used within the DSRCP.

Reconfigurable Logic Cell Design: The DSRCP is capable of performing a variety of algorithms using both conventional and modular integer fields, as well as binary Galois Fileds. These operations are implemented using a single computation unit that can be reconfigured on the fly to perform the required operation. The possible configurations are Montgomery /Multiplication/ reduction, $GF(2^n)$ multiplication and inversion. All other operationa are handled by other units such as the fast adder and the comparator or implemented in the microcode.

III. Dynamic FPGA implementation of the Serpent block cipher algorithm

A Jbits implementation of Serpent Block Cipher in a FPGA is described here. Jbits provides a JAva based Application Programming Interface(API) for the run-time modification of the configuration bitstream. This allows dynamic circuit specialization based on a specific key and mode(encrypt or decrypt). Subkeys are computed in Software and treated as constants in the Serpent datapath. The resulting logic optimization produces a circuit that is half the size and twice the speed of a static, synthesised implementation. With a throughput of over 10 Gigabits per

second, the JBits implementation has sufficient bandwidth for SONET networks. The NIST solicited candidates for a successor to DES,w hich is to be called the Advanced Encryption Standard(AES). AES, like DES is a private key algorithm which uses teh same key for both encryption and decryption. Field Programmable Gate Arrays are frequently used to implement cryptographic algorithms.

Serpent Algorithm: Serpent algorith as given [7] is a substitution-permutation (SP) network that uses 32 rounds. The output of a round i is the input to round(i + 1). The algorithm has two modes of implementations: standard and bitslice. The standard mode operates on individual bits or groups of four bits, while the bitslice mode improves software efficiency by operating on entire 32 bit-words. Serpent Software using the bitslice optimization encrypts at roughly 32 Mbits/sec on a 200 MHz pentium.

Key Scheduling: If required the user supplied key is first padded to 256 bits. This is done by assignin a 1 to the most significant bit and a 0 to the remaining bits. The key is stored as eight 32 bit-words. This is then used to generate the prekeys.

Run time configuration: The run time optimization of FPGAs to the problem instance at hand can have considerable speed and area advantages. For example a dynamic implementation of the DES algorithm exceeds the performance of the fastest known DES ASIC. But, most systems do not exploit the run time reconfiguration (RTR) of SRAM-based FPGAs. This is primarily because there is no support for RTR in the standard design capture, verification and implementation tools. The conventional netlist- based FPGA design flow is the same as for ASICs, and has far too much time and memory overhead to be used in real time or embedded environments. RTR is most easily controlled with a microprocessor. Many systems already make use of one or more microprocessors for those operations thgat do not require hardwrae speeds. Software can directly create or modify the FPGA's configuration with a suitable Application Programming Interface(API). This Serpent model readily supports hardware/software co-design, since the integration of hardware and software occurs early in the development effort.

IV. Performance evaluation of AES algorithms using FPGAs

In this section we look the the comparision of the performance of the FPGA based implementations of five AES algorithms(MARS, RC6, Rijndael,Serpent and Twofish. Among the various time-space implementation tradeoffs, the focus is primarily on time performance. The time performance metrics are throughput and key setup latency. Throughput corresponds to the amount of data processed per unit time while the key set up latency time is the minimum time required to commence encryption after providing the input key. Time performance and area requirement results are as given by [5] is summarised in the Table 1.

Table 1: Performance comparision with ASIC implementations.

AES	throughput	Key-setup
MARS	1.79	4.87
RC6	1.09	33.76
Rijndael	1/1.71	—
Serpent	1/1.35	1/4
Twofish	1.64	1/3

V. Design and implementation of a 2002KCrypto-Processor

Now, in this section we give you a description of the new 2002KCrypto-Processor developed by the Avnee Incorporation. This is conisdered as a major breakthrough in the filed of Cryptography. Given below a brief outline of the 2002KCrypto architecture and the building blocks. The 2002KCrypto core has 4 major building blocks namely: Key-Storage, Comparator, nonary-converter and reconfigurable datapath. The Key-Storage has a maximum storage capacity of several Gigabytes, exceeding the size of any commercially available cryptographic device. Its primary purpose is to store the keys(both public and private). The nanotube technology is used in the manufacture of this high speed storage structure.The keys are encrypted using the nonary-converter block. This block operates on base9 numbers. Every client who registers with this device is given a unique private key and a public key which is encrypted using the nonary-converter and stored in the key-storage. When another clientB wants to set up a secure connection with clientA, then they both send their public keys to this processor, which sends back an encrypted base-9 number which can be decrypted using the private key of the clients. Then the clients send the decrypted code back to the processor. The processor further decrypts it and compares with the already established private key for both clientA and clientB. If they match, a connection is set up or else it is rejected.

VI. Conclusions

Cryptography has come into limelight more than ever due to the increased need for secure connection, at the same time having a reliable and faster connection. In this regard the NIST has defined some pre-set standards which are to be abided by all the Cryptographic Hardware Systems. In this paper we tried to give a bird's eye view of Cryptographic Hardware Architecture, Cryptographic Algorithms and FPGA implementations. FOr further information the reference and bibliography is provided.

VII. References and Bibliography

References

[1] W.P. Choi L.M. Cheng, "Modelling the Crypto-Processor from Design to Synthesis," *Cryptographic Hardware and Embedded Systems (CHES)*, pp. 25–36, August 1999.
bibitemCHES:DES ASIC Perry Roberston et al. D. Craig Wilcox, Lyndon Pierson, "A des asic suitable for network encryption at 10 gbps and beyond," *Cryptographic Hardware and Embedded Systems (CHES)*, pp. 37–48, August 1999.

[2] J.-H. Chung E. Hong and C. H. Lim, "Hardware design and performance estimation of the 128-bit block cipher crypton," *Cryptographic Hardware and Embedded Systems (CHES)*, pp. 49–60, August 1999.

[3] Johann Groschd, "High-Speed RSA Hardware Based on Barret's Modular Reduction Method," *Cryptographic Hardware and Embedded Systems (CHES)*, pp. 191–203, April 2000.

[4] James Goodman and Anantha Chandrakasan, "An Energy Efficient Reconfigurable Public-Key Cryptography Processor Architecture," *Cryptographic Hardware and Embedded Systems (CHES)*, pp. 175–190, April 2000.

[5] ToAndreas Dandalis, Viktor K. Prasanna, and Jose D.P. Rolim, "A Comparative Study of Performance of AES Final Candidates Using FPGAs," *Cryptographic Hardware and Embedded Systems (CHES)*, pp. 125–140, April 2000.

[6] Colin D. Walter, "Data Integrity in Hardware for Modular Arithmetic," *Cryptographic Hardware and Embedded Systems (CHES)*, pp. 204–215, April 2000.

[7] Cameron Patterson, " A Dynamic FPGA Implementation of the Serpent Block Cipher," *Cryptographic Hardware and Embedded Systems (CHES)*, pp. 141–155, April 2000.