IMPLEMENTATION OF ENCRYPTION ALGORITHMS ON TRANSPORT TRIGGERED ARCHITECTURES

P. Hämäläinen[1], M. Hännikäinen[1], T. Hämäläinen[1], H. Corporaal[2], and J. Saarinen[1]

[1]Digital and Computer Systems Laboratory
Tampere University of Technology
Hermiankatu 3 A, 33720 Tampere, Finland
{panuh, markoh, timoh, jukkas}@cs.tut.fi

[2]IMEC/DESICS
Kapeldreef 75, B-3001 Leuven, Belgium
henk.corporaal@imec.be

ABSTRACT

The paper studies a configurable processor architecture, Transport Triggered Architecture (TTA), for encryption algorithm implementations. The automatic TTA design space exploration is applied and configurations with good cost-performance ratio are found. It is shown that TTAs are at least equal to commercial processors in performance. According to earlier studies the performance level is also achieved at far lower cost, which encourages for further development with tuned functionality.

1. INTRODUCTION

Originally microprocessors were meant to be used in general-purpose computers for solving mathematical problems. However, during the years special-purpose embedded systems have become the most important area of the processor market. The demand for flexible and configurable architectures has markedly increased. It is impossible to design an optimized architecture suitable everywhere, but by making it as adaptable as possible without impairing performance, most requirements can be met. One family of processor architectures that are specifically developed considering flexibility and configurability is called Transport Triggered Architectures (TTAs) [2]. In addition to the flexibility and application specific processor design, the architecture also provides support for instruction level parallelism (ILP).

One field that can make use of these special features provided by TTAs is encryption. As telecommunications - both wired and wireless - has become a hot topic, the previously overlooked encryption for achieving service security has also been brought under closer examination. Current communication devices are increasingly embedded systems with real-time service requirements. Therefore, it is important to find as efficient encryption implementations as possible. Most encryption algorithms include operations (e.g., bitwise permutations) that are not efficiently supported in traditional processor hardware. Therefore, those have to be implemented with several basic operations, which is often tricky and inefficient. In addition to special hardware, utilizing parallelism can improve encryption performance.

This paper studies three encryption algorithms, Improved Wired Equivalent Privacy (IWEP), RC4, and Triple Data Encryption Standard (3DES), for TTA implementations. In order to exploit ILP as much as possible while maintaining the architecture general and highly applicable, customization using only basic operations is studied.

The paper is organized as follows. The first section briefly describes the processor architecture and the software framework for the TTA development. Then, the studied encryption algorithms are introduced. Next, the realized implementations and performance results on TTAs are presented. The results are compared to those on Pentium III and Texas Instruments TMS320C6201 processors. Finally, conclusions are given and future work is discussed in the concluding section.

2. TRANSPORT TRIGGERED ARCHITECTURE

The main difference of TTAs compared to traditional, operation triggered processor architectures is the way the operations are executed. Instead of triggering data transports, in TTAs operations occur as a side effect of data transports, i.e., the execution begins when data is written to operand registers. This design implies that only one instruction, *move*, is needed for TTA programming. Therefore, the architecture is also called the *MOVE architecture*. Figure 1 depicts the basic TTA processor structure. [2]

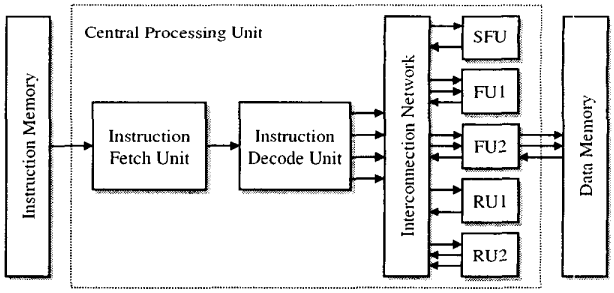

Figure 1. TTA architecture.

The main part of the central processing unit (CPU) is organized as a set of functional units (FUs), special function units (SFUs), and register units (RUs). The data transfers between these units are done through an interconnection network that consists of the desired number of buses. RUs contain the processor's general-purpose registers, and FUs implement the different arithmetic operations needed in program execution. In Figure 1, *FU2* operates as a load-store unit (LSU) handling the data transfers between the data memory and register file. The architecture is very flexible because the number of FUs, RUs, RU ports, buses, and bus connections can be changed unlimitedly. The MOVE compiler supports all these changes. [2]

0-7803-6685-9/01/$10.00©2001 IEEE

As stated, a TTA processor is programmed (in the lowest level) using only one type of operation, *move*. An instruction can consist of either one or more move operations, depending on the interconnection network. If only one move can be executed at a time, only one move can be included in an instruction. By adding buses and connections, more moves can be executed in parallel (provided that there is parallelism in the application) and thus the execution time is reduced. The source and destination of a move can be any register in RU and any FU (or SFU) input or output. [2]

In addition to the flexible basic architecture, the MOVE framework allows the designer to add application specific operations into the instruction set [4]. The new operations are implemented by SFUs. This is a very efficient way to improve performance since quite complicated operations in traditional processors may be trivial in application specific hardware.

The user software tools for MOVE development include a *compiler* to translate high-level programming language (i.e. C/C++) into sequential move code, a *scheduler* to schedule the sequential code and produce parallel code, and a *simulator* to verify and evaluate both the sequential and parallel codes. Because of the flexibility, finding out the best configuration for the application by hand is a very time-consuming and error-prone task. Therefore, a *design space explorer* is also provided. The explorer can be used to test different MOVE configurations for the application. Finally, synthesizable Very high-speed integrated circuit Hardware Description Language (VHDL) can be automatically generated for the chosen configuration using the MOVE Processor Generator (MPG) [1].

3. ENCRYPTION ALGORITHMS

This section describes the three encryption algorithms studied for TTA implementations. The ciphers are Improved Wired Equivalent Privacy (IWEP), RC4, and Triple-DES (3DES). The algorithms were chosen because hardware and software implementations for them were already realized earlier [5]. The implementations have also given ideas what the SFUs that will be developed in future should contain. Furthermore, different implementation methods, including TTA implementations, are studied in order to evaluate the algorithms' suitability for the terminals of a proprietary, multimedia-capable Wireless Local Area Network (WLAN) called TUTWLAN [10].

IWEP was developed at Tampere University of Technology (TUT). It was designed considering hardware implementations for the encryption of time-critical data in TUTWLAN system. Despite its name the cipher is considerably different from Wired Equivalent Privacy (WEP) algorithm included into IEEE 802.11 standard for WLANs [6]. The WEP standard utilizes RC4 stream cipher while IWEP is a simple block cipher encrypting 64-bit blocks of data by mixing them with a 64-bit key. [7]

RC4 is a commercial product used in many applications requiring a strong encryption. This cipher was developed by RSA Data Security, Inc. RC4 operates plaintext a single byte at a time. The idea of the algorithm is to produce an 8-bit pseudo random number series initialized by the given key. An encrypted output is then exclusive-OR (XOR) between the random bytes and the input data bytes. More detailed information on this algorithm can be found in [8].

3DES is probably the most widely used encryption algorithm. It is commonly regarded as a strong, but also rather weighty algorithm. Therefore, as the real-time and other performance requirements are becoming more demanding, an alternative for this algorithm is being sought. 3DES was designed to encrypt 64-bit blocks of data under the control of three unrelated 56-bit keys. It consists of three rounds of DES algorithm used in "encrypt-decrypt-encrypt" order. As the Rijndael encryption algorithm was recently selected for Advanced Encryption Standard (AES), the successor of DES, it is also likely to replace 3DES in new products. An exact description of 3DES can be found in [3] and [8].

4. TTA IMPLEMENTATIONS

This section describes the TTA design space exploration and the achieved results for the presented ciphers. In this case only the basic operations were used, so no SFUs were added to the architectures. Here "basic" means common Reduced Instruction Set Computer (RISC) operations like *add* and *store*. This choice was made to enable proper comparison to off-the-shelve architectures and to keep the configurations as generally applicable as possible. The examinations concentrated on the critical functions of the ciphers. Functions such as initializations and key setup procedures were left outside the study.

The same initial TTA configuration was used in explorations in order to see how differently it changes for each algorithm. Before the initial architecture was created, the algorithms were examined in order to be able to include enough resources for all of them. The initial configuration is very large: eight LSUs, eight Arithmetic Logic Units (ALUs), twenty 32-bit-wide buses (fully connected), four immediate units (IUs), two integer register units (IRUs) both containing 64 registers and eight input and output ports, and one Boolean register unit (BRU) with eight input ports. The ALUs are able to perform addition, subtraction, comparison, shift, and logic operations.

In order to enable comparison with other processor architectures, the throughputs for the same source codes on Intel Pentium III (PIII) and on Texas Instruments TMS320C6201 digital signal processor [12] are also presented. PIII is a contemporary processor used mainly in personal computers. However, when considering embedded systems, it is too expensive and too power-hungry to be utilized in them. Conversely, 'C6201 is one of the most powerful fixed-point signal processors based on Very Large Instruction Word (VLIW) architecture. The structure of 'C6201 is quite close to TTAs because it also has several processing units connected with a communication network. On the other hand, the architecture is fixed and therefore does not provide the flexibility of TTAs.

The following results present the throughputs in megabytes (MB) at 200-MHz clock speed. This clock rate was chosen because it is realistically attainable using automatically synthesized standard cell implementation (i.e. no large VLSI development team required). In addition, the used 'C6201 test board runs at the same speed [12].

One advantage of TTAs is that the performance-cost ratio is always very good, which has been proved in earlier studies [2]. As the following results show, even a simple configuration can be efficient. Therefore, one should notice that in order to emphasize this virtue, the presented results are for the configurations with the

best execution time-cost ratio, not for the configurations with the shortest execution time.

All ciphers were written in C and the same code was compiled for all processors. IWEP was implemented at TUT, the RC4 source code follows the code presented in [9], and 3DES source was obtained from Phil Karn of Qualcomm Incorporated[1]. It is fair to mention that since the codes are portable, they do not take advantage of the special intrinsic instructions of 'C6201 and therefore the results on this processor could be improved.

4.1 TTA Design Space Exploration

Figure 2 depicts the exploration process for IWEP. The design space exploration proceeds the following way. First, the explorer is used for finding the fully-connected architecture with best performance-cost ratio. In the figure the chosen architecture is marked with a circle in the graph on the left. After that, different connection combinations are tested until the fastest configuration is found. This architecture is marked in the graph on the right. One could think that a fully-connected network results the shortest execution time. However, after removing sparsely used connections and taking advance of software bypassing, the bus load decreases implying shorter clock cycle and, since the cycle count remains the same, faster execution [1]. The figure shows that for this algorithm the best cost-performance architecture is also almost the fastest and cheapest before connectivity explorations. The same design space exploration was applied to the other two algorithms.

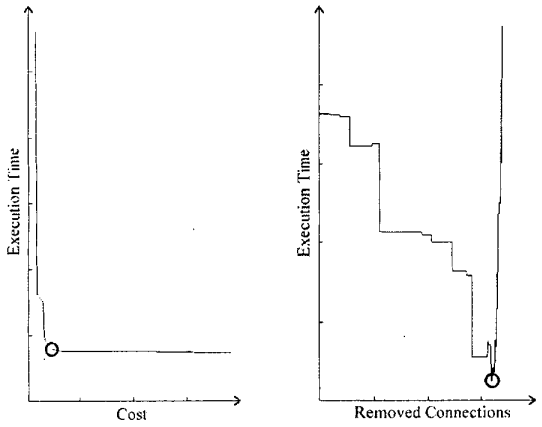

Figure 2. Design space exploration for IWEP.

Figure 3 shows the selected TTA configuration for IWEP after the design space exploration. The figure also presents the bus utilizations in percentage during execution. The architecture corresponds to what was expected. Two LSUs are still included in the final configuration, which means that the architecture takes advantage of instruction level parallelism. LSUs handle only byte-wide data, which eases memory port design. However, because the configuration with the best time-cost ratio was chosen, not all possible operations are executed in parallel. After the resource reduction, more expensive configurations with more ALUs were also available. However, in this case they were not chosen for connectivity exploration.

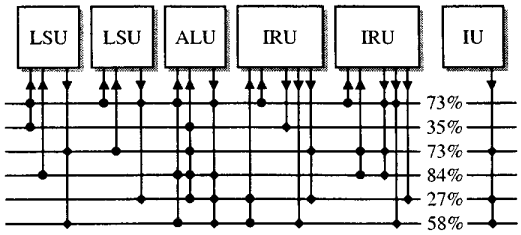

Figure 3. TTA configuration for IWEP.

As it was expected, parallelism was not applicable to the RC4 cipher. Every operation is dependent on the results of the previous ones. Figure 4 depicts the hardware configuration for this algorithm. Because only one ALU is included in the architecture, only one arithmetic operation can be executed at a time.

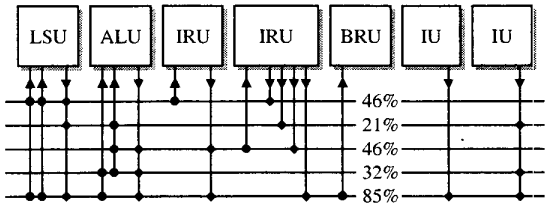

Figure 4. TTA configuration for RC4.

It was also difficult to find parallelism in the 3DES design, as the configuration given by the explorer in Figure 5 illustrates. In this reasonable speed-cost configuration only one ALU and LSU are needed. The number of required buses is also small, the smallest for the three ciphers. Therefore, the architecture is also the cheapest.

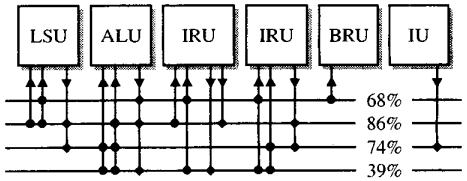

Figure 5. TTA configuration for 3DES.

4.2 TTA Results

Table 1 presents measurements for the TTA configurations of each algorithm. The TTA results are calculated according to the statistics given by the scheduler.

For IWEP the results as well as the compiler generated 'C6201 assembly show that the execution is done in parallel. Since on average two moves are needed for one RISC instruction, up to two instructions are executed simultaneously. As shown, even this basic implementation without any SFUs is faster on TTAs than on PIII and almost equal to the 'C6201 processor.

[1] The source code for 3DES is available at http://people.qualcomm.com/karn/code/des (visited January 22, 2001).

Table 1. Encryption implementation results.

Cipher	*Moves per instruction*	*Throughput (MB/s)*		
		TTA	*PIII*	*'C6201*
IWEP	3.29	8.08	6.20	8.16
RC4	2.12	7.10	6.20	5.71
3DES	2.77	0.64	0.80	0.54

The table shows that on average only one RISC instruction is performed at a time in RC4 execution. Still, even though the execution is sequential, the cipher is faster on TTAs than on the compared processors due to software bypassing

As it can be seen, parallelism is a little better utilized in 3DES implementation than in RC4 implementation. However, the throughput on PIII processor is better than on TTA. This is because on PIII and 'C6201 the inner loop of the source code was fully unrolled. On the contrary, in order to keep the code reasonably small to enable faster scheduling, the loop was not unrolled in TTA implementation. Thus, the scheduler was not able to schedule over successive iterations. By choosing a faster and more expensive configuration, TTA would be closer to PIII despite of the difference in the source. As stated, 3DES is commonly regarded as a heavy cipher, and therefore the throughput should still remain on much lower level than that of IWEP or RC4.

5. CONCLUSIONS AND FUTURE WORK

Because of flexibility, TTAs are extremely suitable for application specific processor design. With careful pre-examination of the source code and help of the software development framework, it is easy to design efficient architectures for different applications. Apart from fixed architectures, in which applications has to be adjusted to the hardware in order to get the best results, in TTAs the hardware can be automatically adjusted to the application. This makes it easier for a designer to achieve better performance, without serious debugging and optimization of the source codes.

Since the results for the architectures are given by the scheduler software, they are all estimates of real hardware implementations. The hardware estimation model that the scheduler uses is created by the designer. Therefore, it is up to his/her skills how close the model is to the real world. For example, in this study the LSU latency was estimated to be two clock cycles and caches were not taken into account.

The presented results show that even with the basic operations TTAs can be faster than other processors of the same clock frequency. In the future, by adding special function units, the results should improve and the throughputs are expected to be close to the full-hardware implementations presented in [5]. Furthermore, the framework also provides automatic loop unrolling, software pipelining, and function inlining, which were not utilized within this study. When applied, these all should improve the efficiency because most encryption algorithms, like 3DES and RC4, contain an inner loop. Moreover, when using block ciphers like IWEP and 3DES, the outer loop can also be made parallel and several blocks of data can be encrypted simultaneously. This should increase the performance, and the limit for the number of parallel encryptions will only be set by the cost of the hardware.

RC4's RISC-oriented design does not provide a reasonable possibility for SFU implementation. In [5] it was verified to be no faster in FPGA implementation than when running on a processor. On the contrary, the TTAs were found very suitable for encryption algorithms like IWEP and 3DES. They both contain operations that are much faster in dedicated hardware than in software. Therefore, this study gives encouragement for further development for these ciphers by adding SFUs and making the outer loop parallel. A follow-up paper will concentrate on the specialization of the TTAs more to the application domain by tuning also the functionality. In addition, it would also be interesting to see how the recently selected AES algorithm performs on TTAs.

6. REFERENCES

[1] Corporaal H., and Hoogerbrugge J., "Cosynthesis with the MOVE Framework," *CESA'96 IMACS Multiconference*, Lille, France, 1996, pages 184-189.

[2] Corporaal H., *Microprocessor Architectures from VLIW to TTA*, John Wiley & Sons Ltd., West Sussex, England, 1998.

[3] FIPS PUB 46-7, *Data Encryption Standard*, Federal Information Processing Standards Publication, National Institute of Standards and Technology (NIST), USA, 1999.

[4] Hoogerbrugge J., *Code Generation for Transport Triggered Architectures*, PhD Thesis, Delft University of Technology, Delft, The Netherlands, 1996.

[5] Hämäläinen P., Hännikäinen M., Hämäläinen T., and Saarinen J., "Hardware Implementation of the Improved WEP and RC4 Encryption Algorithms for Wireless Terminals," *X European Signal Processing Conference* (EUSIPCO 2000), Tampere, Finland, 2000, pages 2289-2292.

[6] IEEE Std 802.11. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*. LAN MAN Standards Committee of the IEEE Computer Society, USA, 1999.

[7] Salli K., Hämäläinen T., Knuutila J., and Saarinen J., "Security Design for A New Wireless Local Area Network TUTWLAN", *9th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications* (PIMRC'98), Boston, USA, 1998, pages 1540-1544.

[8] Schneier B., *Applied Cryptography: Protocols, Algorithms and Source Code in C*, 2nd ed., John Wiley & Sons, Inc., USA, 1996.

[9] Schneier, B., and Whiting D., "Fast Software Encryption: Designing Encryption for Optimal Software Speed on the Intel Pentium Processor," *Fast Software Encryption Workshop 1997* (FSE4), Haifa, Israel, 1997, pages 242-259.

[10] Tikkanen K., Hännikäinen M., Hämäläinen T., and Saarinen J., "Advanced Prototype Platform for a Wireless Multimedia Local Area Network," *X European Signal Processing Conference* (EUSIPCO 2000), Tampere, Finland, 2000, pages 2309-2312.

[11] *TMS320C6000 DSP Platform Technical Documentation*, Texas Instruments, Inc., USA, 2000.

[12] *TMS320C62xx CPU and Instruction Set Reference Guide*, Texas Instruments, Inc., USA, 1997.