Reconfigurable Elliptic Curve Cryptosystems on a Chip

Ray C.C. Cheung, Wayne Luk
Department of Computing
Imperial College London, UK
r.cheung, w.luk@ic.ac.uk

Peter Y.K. Cheung
Department of EEE
Imperial College London, UK
p.cheung@ic.ac.uk

Abstract

This paper presents a System-on-a-Chip (SoC) architecture for Elliptic Curve Cryptosystems (ECC) which targets reconfigurable hardware. A four-level partitioning scheme is described for exploring the area and speed tradeoffs. A design generator is used to generate parameterisable building blocks for the configurable SoC architecture. A secure web server, which runs on a reconfigurable soft-processor and an embedded hard-processor, shows over 2000 times speedup when the computationally-intensive operations run on the customised building blocks. The embedded on-chip timer block gives accurate performance information. The design factors of configurable SoC architectures are also discussed and evaluated.

1 Introduction

Field Programmable Gate Arrays (FPGAs) enable a high degree of parallelism and can achieve orders of magnitude speedup over general purpose processors [5]. Many applications such as signal, image, video, networking and security have been successfully mapped into reconfigurable hardware, for exploring the on-chip parallelism and achieving design flexibility. As the design moves to system-level integration, the control logic and the computationally-intensive logic are usually separated. Configurable System-on-Chip (CSoC) devices [3, 11], which are a combination of embedded microprocessors, memory and embedded programmable logic, have attracted academic research and also resulted in industrial products such as Triscend A7, Xilinx VirtexII Pro and Altera Excalibur. Each of them has one or more embedded microprocessors using different instruction set architectures at different clock speeds, and large programmable logic resources. Previous work mainly focuses on the speedup from using pure FPGAs over processors, and analyses the mapping from different application domains to different configurable resources [14].

In this paper, we aim to explore the huge design spaces provided by CSoC devices, to describe the limiting factors in designing CSoC applications and to predict the dominating factors as the technology grows. It is clear that in this design space, the two extreme cases are the pure software implementation running on a generic processor which has no additional area overhead, and the hardware implementation purely running on configurable logic which gives the best performance but uses the largest area. As shown in Figure 1, the software program can be run on a PC or an embedded processor, or the design could be fully mapped into hardware. On the other hand, there are several design parameters such as the processor speed, the reconfigurable logic speed, the memory speed and the CoreConnect [13] bus speed connecting the processor and user logic. We define Intellectual Property (IP) block as a building block which contains customisable user logic. In this paper, we address the following two questions:

1 How much application logic should we put into the reconfigurable fabric?

2 What are the limiting factors on the performance for a CSoC design; in particular, which factor dominates?

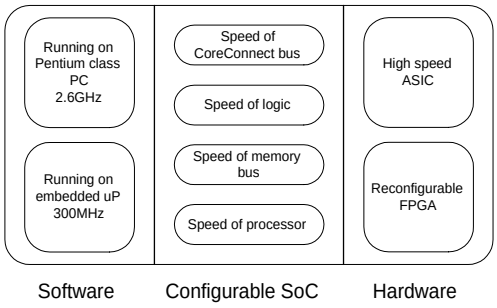

Figure 1: The limiting factors for Configurable SoC.

Elliptic curve cryptography (ECC) is a public key cryptography system superior to the well-known RSA cryptography: for the same key size, it gives a higher security level than RSA [5]. ECC has been adopted in a wide variety of applications from digital certificates in webserver authentication [6] to embedded processors in wearable devices. A major trend of research work is hardware acceleration, and

1530-1591/05 $20.00 © 2005 IEEE

some recent work addresses system integration [6, 7] using the PCI interface and proprietary interconnect. In this paper we propose an SoC approach and evaluate the effect of partitioning in a single chip. Our main contributions include:

- A building block containing a field multiplier where degree of parallelism can be varied (Section 3).
- Customisable IP blocks for four-level partitioning of ECC operations (Section 3).
- An SoC architecture and design generator for ECC IPs incorporating embedded processors (Section 4).
- Performance evaluations of individual IP blocks and the configurable SoC chip (Section 5).

The rest of the paper is organized as follows. Section 2 describes related work on ECC. Section 3 presents the architectures of the relevant customisable IP blocks. Section 4 describes the design generator of our approach, and the system integration between IP blocks and embedded microprocessors. Section 5 evaluates our results and compares the software and hardware designs. Section 6 describes a framework of a secure web server system running on the CSoC chip. Finally, Section 7 summarises our current and future research.

2 Background and Related Work

The difficulty of the underlying Elliptic Curve Discrete Logarithm Problem (ECDLP) makes ECC cryptosystems suitable for applications that need long-term security and low bandwidth measurements. For instance, the NIST has recommended specific curves for implementations [12] and the IEEE has provided detailed specifications for the choices of private key length and fields [5].

ECC research can be first divided into two groups depending on the underlying field representation: prime field, $GF(p)$ and binary field, $GF(2^m)$. Two bases, Optimal Normal Basis (ONB) and Polynomial Basis (PB), are commonly used for manipulating binary fields. It is known that the binary field is more suitable for hardware implementation, and squaring is very fast using ONB. However, point multiplication requires efficient design for field multiplication, inversion, squaring and efficient coordinate systems. In recent years, there has been much software [1] and hardware [2] research work on both $GF(p)$ and $GF(2^m)$ focusing on the performance of point multiplication, and it is obvious that a fast point multiplication design is crucial.

Previous hardware work includes: the first ASIC design with a Motorola M68008 microcontroller which calculates 9 point multiplication per second, an ECC design on a 8051 microprocessor in smart cards, and recent FPGA implementations for ECC designs [9]. System level work has also been reported such as a secure web server [6] and a reconfigurable computer [7]. In this paper, we build a flexible CSoC using configurable logic and embedded processors, and apply it to applications such as a secure web server system.

3 Design Partitioning

This section describes the operations in our ECC architecture using $GF(2^m)$ with ONB. Section 3, 4, 5 use a bottom-up approach, we first break down the ECC operations and put them into IP blocks in Section 3, then describe the design flow and how we construct the hardware architecture in Section 4. An overview of the interactions amongst ECC routines is shown in Figure 2. Our architecture supports point multiplication, addition and substraction, in this paper we focus on the design partitioning of point multiplication. The arrows in the figure show the dependencies amongst different operations.

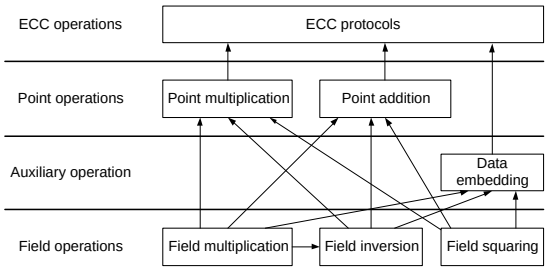

Figure 2: Interactions between different operations.

3.1 Level 1 - Pure Software

In this paper, we adopt the open source software from Rosing [10]. The basic routines such as the field and point multiplications are implemented in C and compiled for both Pentium PC and embedded processors. Table 1 shows the number of field multiplications and field inversions for one point multiplication. By using this information, we can evaluate the effect of replacing the software routine by configurable logic.

field size m	53	113	131	233	270
no. of multiplications	612	1530	1760	3732	4758
no. of inversions	68	153	176	311	366

Table 1: Statistics of field multiplication and field inversion for different field size m.

3.2 Level 2 - Field Multiplication

In our ECC cryptosystem, the field multiplier as shown in Figure 3 is based on our previous work [8]. This field multiplier is repeatedly used by other operations as shown in Figure 2. The datapath of our system for various operations is shown in Figure 4. For the pure hardware architecture, users are able to select the second level paral-

lelism in terms of the number of executable field multipliers up to a maximum four parallel field multipliers. Different scheduling optimizations have been applied to the corresponding designs. Using more than four field multipliers does not bring further speed improvement but induces an area penalty. There is a tradeoff between efficiency and area, and the designer must select the best design.

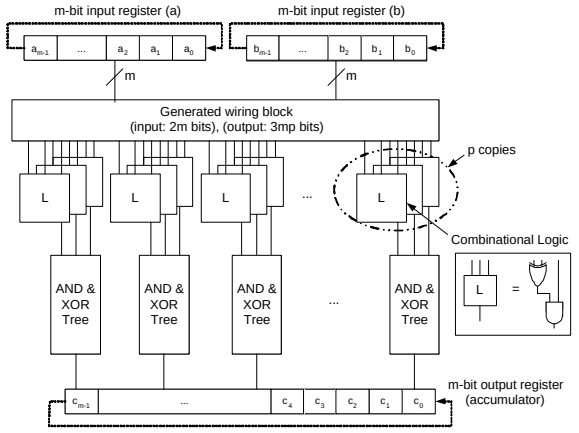

Figure 3: Parallel field multiplier IP block.

Since field multiplication is the most computationally-intensive operation, we separate and build it as a single IP block using the design flow as shown in Figure 5. The inputs to this block are two m-bit values, and another m-bit data are generated at the end of the computation.

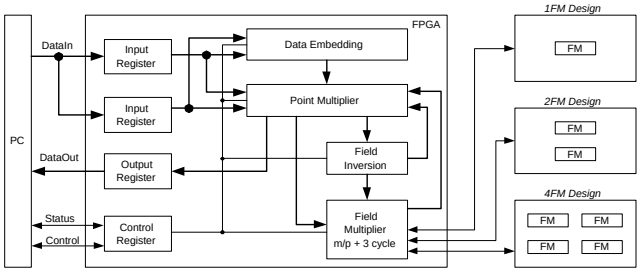

Figure 4: Datapath of the customisable ECC system.

3.3 Level 3 - Field Inversion

In level 3, we develop an IP solely for computing field inversion. The algorithm we used for field inversion is based on the Fermat's theorem which states that in a normal basis:

$$a^{-1} = a^{2^m-2} = \left(a^{(2^{m-1}-1)}\right)^2$$

The pseudo-code for this field inversion can be found in our previous work [8]. The basic idea is to find an inverse m-bit field element by inputting an arbitrary m-bit field element. In this field inversion block, the field multiplication circuit described in level 2 is reused and is placed as a component inside this block.

3.4 Level 4 - Point Multiplication

Point multiplication is the key operation in ECC and it is the most time-consuming process. We adopt the improved Montgomery Scalar multiplication [4] in our design. In our embedded logic core, users are able to preload some of the system options such as the base point and the curve information. This could save time as the basic point multiplication "$Q = k \times P$" where $P = P(x, y)$ that has two components. The bus width of our prototyping platform is 32-bit, as a result, the k and P values are tokenized and then sent between the embedded logic and the embedded processor.

4 Design Generator

In this section, we describe a design generator that produces high-level Handel-C code for arbitrary field size and degree of parallelism. The generated Handel-C is then translated into VHDL; we use a developed wrapper to connect this user logic with the interface connect core. As shown in Figure 5, the customised IP block is thus connected to the bus with its own address space such that the embedded processor can directly send data to it.

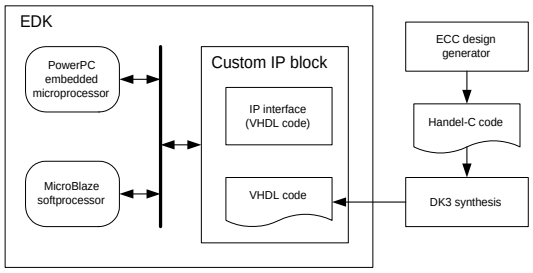

Figure 5: Diagram of the design flow.

The system overview that makes use of embedded processors and configurable logic IP blocks which are generated in the design flow. As shown in Figure 6, the On-Chip Peripheral Bus (OPB) provides a fast link between logic cores such as the configurable user IP block and the on-chip timer. The PowerPC processor is first connected to the high-speed Processor Logic Bus (PLB) and then uses a bridge to connect to the OPB peripherals.

5 Performance Evaluation

The results are divided into two parts. (1) Evaluation of a standalone ECC IP block using the Celoxica RC2000 board containing an XC2V6000 FPGA chip. This compares the speed of point multiplication using different hardware and software. (2) Evaluation of the CSoC design using the Xilinx ML310 board containing an XC2VP30 FPGA chip. This compares the speed of implementing different IPs using configurable logic. The reported timing information and area usage are collected from the Xilinx place-and-route tools and the OPB on-chip hardware timer.

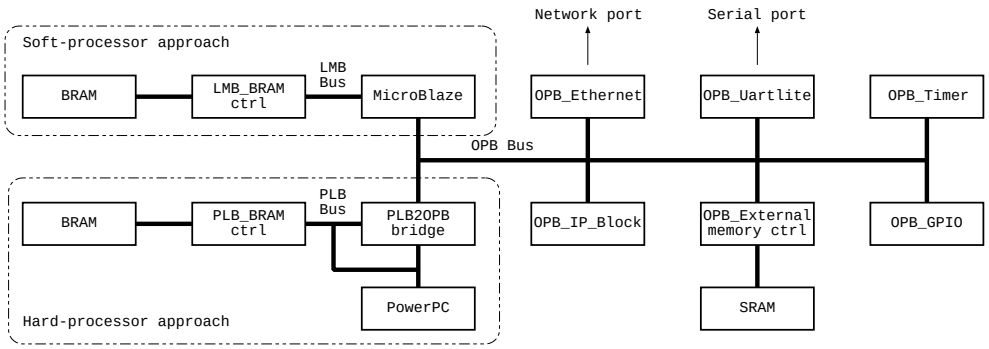

Figure 6: Architectural components in a single chip.

5.1 Individual IP Block

We compare the performance of various software and hardware implementations for point multiplication, which is the bottleneck for ECC systems. We have implemented the software design [10] on a dual-processor Intel Xeon 2.66GHz with 4GB of RAM. The comparison for serial and parallel designs on different m and p values, where p refers to the degree of parallelisation, is presented in Table 2. Note that Place-and-Route (P&R) results refer to those obtained from the Celoxica DK3 and Xilinx ISE 6.2 tools, and measured results refer to those from hardware realisation. We have also verified each design using 300,000 consecutive point multiplications using data sent and received from the PC. The "Speedup" column shows the performance gain of our design over other methods. This gain is due to our highly parallel architecture which does not involve instruction fetch and decode.

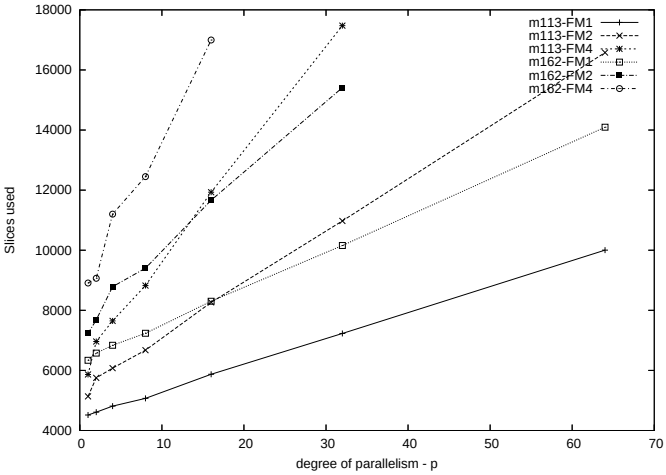

Figure 7: Area usage for various p and m values.

As shown in Figure 7, the area requirements for various designs such as $m = 113, 162$ using 1, 2 and 4 field multipliers (1FM, 2FM and 4FM) show a linear growth when the degree of parallelism p increases. The speed for performing one point multiplication is shown in Figure 8 with respect to four different cases: m is not divisible by p using 1FM, m is divisible by p using 1FM, 2FM and 4FM. The ECC IP block designed in this paper is highly customisable and can cope with different area and speed tradeoffs.

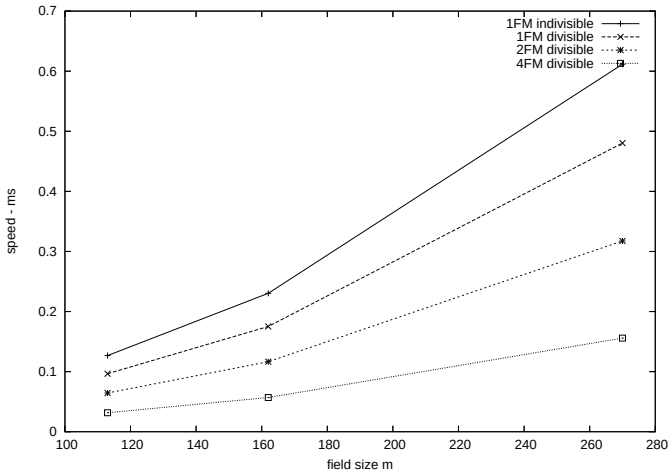

Figure 8: Speed comparison for various p and m values.

5.2 CSoC Design

Figure 9 shows the speed of field multiplication and field inversion for different m values using the embedded PowerPC. We can see that the time spent on one inversion grows faster than that on field multiplication. We also measure the time for transferring one data item and for transferring 1000 consecutive data items using the CoreConnect bus. From Table 3, the overhead of consecutive transfers is smaller. The measured speed by using configurable logic for different levels are given in Table 4, 5 and 6. Note that the cycle count starts once the processor enables the operation and ends when a "done" signal is received from the configurable logic. The software running on the embedded processor is responsible for controlling data transfer.

Parallelism	P&R results		Measured results		Software [10]			Hardware (serial) [9]		
p	clock	time(ms)	clock	time(ms)	clock	time(ms)	speedup	clock	time(ms)	speedup
$m = 113$										
2	42MHz	0.36	56MHz	0.27	2.6GHz	15.99	59.22	31MHz	4.3	15.93
8	42MHz	0.13	56MHz	0.09	2.6GHz	15.99	177.67	31MHz	4.3	47.78
16	42MHz	0.08	56MHz	0.06	2.6GHz	15.99	266.50	31MHz	4.3	71.67
32	42MHz	0.07	56MHz	0.04	2.6GHz	15.99	399.75	31MHz	4.3	107.5
56	42MHz	0.05	56MHz	0.04	2.6GHz	15.99	399.75	31MHz	4.3	107.5
$m = 162$										
2	40MHz	0.83	54MHz	0.55	2.6GHz	45.67	83.04	29MHz*	9.39*	17.07
8	40MHz	0.27	54MHz	0.17	2.6GHz	45.67	268.65	29MHz*	9.39*	55.24
16	40MHz	0.18	54MHz	0.11	2.6GHz	45.67	415.18	29MHz*	9.39*	85.36
32	40MHz	0.13	54MHz	0.07	2.6GHz	45.67	652.43	29MHz*	9.39*	134.14
56	40MHz	0.10	54MHz	0.06	2.6GHz	45.67	761.17	29MHz*	9.39*	156.5
$m = 270$										
2	24MHz	3.28	35MHz	2.24	2.6GHz	196.71	87.82	26MHz*	27.99*	12.50
8	24MHz	0.92	35MHz	0.63	2.6GHz	196.71	312.24	26MHz*	27.99*	44.43
16	24MHz	0.53	35MHz	0.36	2.6GHz	196.71	546.42	26MHz*	27.99*	77.75
32	24MHz	0.35	35MHz	0.24	2.6GHz	196.71	819.63	26MHz*	27.99*	116.63
56	24MHz	0.25	35MHz	0.17	2.6GHz	196.71	1157.12	26MHz*	27.99*	164.65

Table 2: Comparison between our design and the reference designs [9, 10]. The symbol (*) denotes extrapolated results based on published data for different m values.

In Table 5, the required I/O cycles mean that in order to replace the software routine by programmable logic, four read cycles and four write cycles are needed. We can see that if the operation performed in logic is very fast and the I/O transfer cycles will dominate this overhead. In Table 4, there is actually no performance gain when the degree of parallelism is equal to or larger than four. On the other hand, there is no I/O overhead for the result using embedded microprocessor. In Table 4, we use the on-chip timer to measure the number of cycles taken to perform one field multiplication for both embedded processor and the IP block.

	Single read	Singe write	Mult. read	Mult. write
Cycle	100	102	65	58

Table 3: Read/Write cycles using the CoreConnect bus, sharing reduced cycles for multiple Read/Write.

parallelism	1	2	4	8	16
cycle count	297	239	181	181	181
speed (us)	3.15	2.62	1.99	1.92	2.18
area (slices)	2063	2199	2487	2691	3494
embedded PowerPC	117591 cycles				

Table 4: Level 2: Comparison of one field multiplication when $m = 113$ (I/O cycles: 8W+4R).

parallelism	1	2	4	8	16
cycle count	1109	703	471	355	297
speed (us)	13.70	8.78	4.99	3.94	3.87
area (slices)	2708	2966	3251	3528	4299
embedded PowerPC	972098 cycles				

Table 5: Level 3: Comparison of one field inversion when $m = 113$ (I/O cycles: 4W+4R).

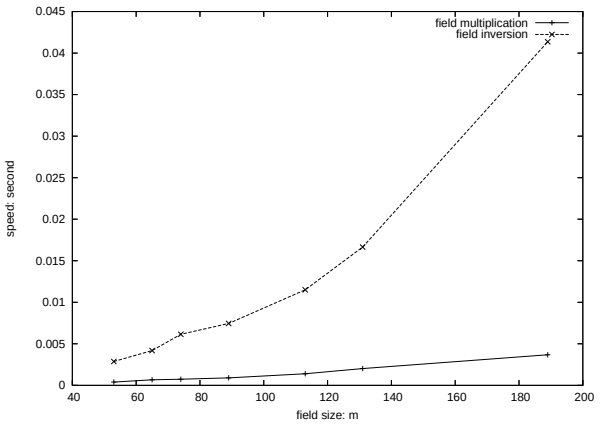

Figure 9: Level 1: OPB Timer block measurement of PowerPC running at 300MHz designs.

parallelism	1	2	16
$m = 53$			
cycle count	19411	16555	3603
speed (us)	206.04	175.63	47.09
area (slices)	3712	3869	4529
embedded PowerPC	19591629 cycles		
$m = 113$			
cycle count	79761	42767	9875
speed (us)	1021.82	570.73	123.01
area (slices)	6001	6344	7677
embedded PowerPC	166950475 cycles		

Table 6: Level 4: Comparison of one point multiplication when $m = 53$ (I/O cycles: 6W+4R) and $m = 113$ (I/O cycles: 12W+8R) using one field multiplier.

5.3 Technology Trend Discussion

The limiting factors of most CSoC systems are the speed of configurable logic, speed of the bus or switch connecting between the embedded processor and the configurable logic, the speed of embedded processors and the memory bandwidth. As shown in our results, the major bottleneck of the CSoC designs is the bandwidth and the latency of the bus connecting the embedded processors and the configurable logic. For instance, an elliptic point in the size 113 field needs to be divided into eight 32-bit words for data transmission. As shown in Table 3, each single R/W operation takes 100 cycles of embedded on-chip timer that is very expensive in high performance applications. We expect that using a wider bus-width in CSoC is the major trend.

Our results show that the clock speed of all synthesized designs is around 70-90MHz which is slower than the embedded processors. The next generation reconfigurable logic is able to achieve up to 500MHz and it will enlarge the forementioned bottleneck. For the memory bus, many systems use the CoreConnect/AMBA bus for connecting the embedded memory and the microprocessors. We expect that a wide low-latency bus, which is tightly coupled the processor, memory and configurable logic will be crucial.

6 Secure Web Server

A secure web server using our CSoC framework is depicted in Figure 10. In this system, a network logic core [13] is added to the OPB bus for communicating between the web server program running on the embedded processors and the network. From Table 6, a speedup factor of 2000 is achieved when we replace the $m = 113$ point multiplication running on the embedded processor by the customised level 4 IP block including the read/write bus overheads.

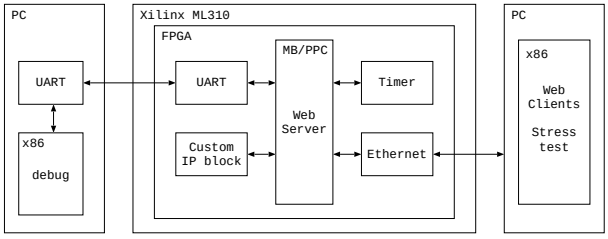

Figure 10: Secure web server overview.

We also illustrate that our CSoC chip can benefit another secure web service. Previous work [6] describes an ECC hardware system that accelerates the Secure Socket Layer (SSL) in a client/server system. The ECC keys are embedded into the generated X.509 certifcates. Their implementation involves an FPGA on a PCI bus at 66.4MHz, and achieves 12.5 times speedup for the Elliptic Curve Key exchange when comparing with pure software implementation. Since the point multiplication in our design is twice as fast, our design could achieve up to 25 times speedup.

7 Conclusions

This paper presents a configurable SoC architecture for Elliptic Curve Cryptosystem and an embedded secure web system using reconfigurable hardware. A four-level partitioning scheme is proposed for exploring the area and speed tradeoff of pure software implementation on embedded processor and the customisable building blocks approach. Our experience shows that performance in CSoC designs is dictated mainly by data transfer speed, hence it is desirable to have a wide bus-width and a wide memory bus with low latency for most embedded system designs. Future work includes CSoC designs for other application domains and the use of run-time reconfiguration.

Acknowledgement. Thanks to Dr T. Todman and the referees for comments, and to the Croucher Foundation for support.

References

[1] D. Hankerson et al. Software implementation of elliptic curve cryptography over binary fields. In *Cryptographic Hardware and Embedded Systems LNCS 1965*, pages 1–24, 2000.

[2] G. Orlando et al. A High Performance Reconfigurable Elliptic Curve for $GF(2^m)$. In *Cryptographic Hardware and Embedded Systems, LNCS 1965*, pages 41–56. Springer, 2000.

[3] J. Becker. Configurable systems-on-chip (CSoC). In *Integrated Circuits and Systems Design*, pages 379–384, 2002.

[4] J. López et al. Fast multiplication on elliptic curves over $GF(2^m)$ without precomputation. In *Cryptographic Hardware and Embedded Systems*, pages 316–327, 1999.

[5] L. Batina et al. Hardware architectures for public key cryptography. *Integration, the VLSI Journal*, 34:1–64, 2003.

[6] N. Gura et al. An End-to-End Systems Approach to Elliptic Curve Cryptography. In *Cryptographic Hardware and Embedded Systems LNCS 2523*, pages 349–365, 2002.

[7] N. Nguyen et al. Implementation of Elliptic Curve Cryptosystems on a Reconfigurable Computer. In *Int'l Conf. on Field Programmable Technology*, pages 60–67, 2003.

[8] N. Telle et al. Customising Hardware Designs for Elliptic Curve Cryptography. In *Int'l Workshop on Systems, Architectures, Modeling and Simulation*, 2004.

[9] P.H.W. Leong et al. A Microcoded Elliptic Curve Processor using FPGA Technology. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, 10(5):550–559, 2002.

[10] M. Rosing. *Implementing Elliptic Curve Cryptography*. Manning Publications Co., 1999.

[11] S. Knapp et al. Field configurable SoC device architecture. In *Custom Integrated Circuits Conf.*, pages 155–158, 2000.

[12] U.S. Dept. of Commerce/National Institute of Standard and Technology. *FIPS 186-2, Digital Signature Standard*, 2000.

[13] Xilinx app. note XAPP433. *Embedded System Example: Web Server Design Using MicroBlaze Soft Processor*, 2004.

[14] Z. Guo et al. A Quantitative Analysis of the Speedup Factors of FPGAs over Processors. In *ACM Int'l Symposium on FPGAs*, pages 162–170, 2004.