A Signing Message Architecture Development for Smart Card Chip Based on Open Sources

Walid Kaaniche* and Mohamed Masmoudi

EMC Research Group, National Engineering school of Sfax, Electrical Engineering Department, Sfax, Tunisia

Abstract: In this paper we present an architecture development to be used in smart card. The architecture is dedicated for digital signature and it is based in the use of the open hard and soft sources. The development of the digital signature is based on the elliptic curve digital signature algorithm (ECDSA) and uses an open source cryptographic library named Miracl. The hardware development of the architecture is based on the use of the LEON2 soft core processor. To make Leon suitable for smart card some changes are made in the open VHDL packages of the LEON2 processor. We first in this paper present the simulation of the developed digital signature on an emulator environment of Leon named TSIM. Secondly, we show the simulation of the enhanced architecture with the developed elliptic curve digital signature in Modelsim environment. Finally an FPGA validation of the architecture implementing a 160-bit ECC over GF (p) is made on an Altera Excalubur development board. In this paper we prove that nowadays some open sources IP cores could be considered as solutions for soc application development.

Keywords: Smart cards, elliptic curve digital signature algorithm, LEON2 processor, FPGA.

1. INTRODUCTION

Today global electronic commerce in the internet is growing to enable profitable and legal trading: authentication, integrity, and Non repudiality of the associated messages are necessary.

- Authentication: a merchant must know the identity of the customer in some case.
- Integrity: the recipient of a message or an order should be sure that the data wasn't altered during its transmission.
- Non repudiality: it is often necessary to assert that a particular person sent an order or message and that no other person could possibly have sent it.

In order to achieve this, a digital signature algorithm with smart cards can answer to all these challenges. In fact, smart cards can be used to securely store secret keys and perform the secret key cryptographic operation, namely signing. So, on this context, the goal of our work was to develop a dedicated signing message chip for smart card based on open source.

In this paper, we first present the hardware core of our chip, the LEON2 SPARC v8 processor. In the second section, we justify the choice of the elliptic curve cryptosystem which will be used in the digital signature. In the third section we briefly explain the arithmetic of the elliptic curve. In the following sections we present the software and hardware design flow of our architecture and the different simulations results.

*Address correspondence to this author at the EMC Research Group, National Engineering school of Sfax, Electrical Engineering Department, Sfax, Tunisia; E-mail: dilaw.ek@planet.tn

2. LEON2 MICROPROCESSOR

LEON2 [1] is a microprocessor which implements a RISC architecture conforming to the SPARC v8 definition [2]. It's a synthesizable core written in VHDL and can be implemented both on FPGAs and ASICs. It's distributed under the terms of the GNU LGPL license so it is an open hardware [3] and it is specifically designed for embedded applications. It was originally developed by the European Space Agency and nowadays it is maintained by Gaisler Research.

The LEON2 32-bit core implements the full SPARC v8 standard. It uses big-endian byte ordering, has 32-bit internal registers, 72 different instructions in 3 different instruction formats and 3 addressing modes (immediate, displacement and indexed). It implements signed and unsigned multiply, divide and MAC operations and has a 5-stage instruction pipeline (Instruction, Fetch, Decode, Execute, Memory & Write). It also implements two separate instruction and data cache interfaces, Harvard architecture.

The VHDL model is fully synthesizable with most of the commonly synthesis tools, it is very configurable and it uses the AMBA-2.0 AHB/APB on chip buses [4], which makes it easy to extend its functionality. All these features make LEON2 an ideal microprocessor for System-on-Chip applications. A block diagram of LEON2 architecture is presented in Fig. (**1**). Many of those blocks are optional and could be removed from the model of our concrete application.

SPARC v8 processor defines three main units, integer unit, floating-point unit and a custom coprocessor, each one with its own 32-bit internal registers. The later two units are optional, not mandatory for the processor to be SPARC complaint.

LEON2 implements the integer unit completely and the interfaces for the other two units in its core. Gaisler Research

1874-1290/08

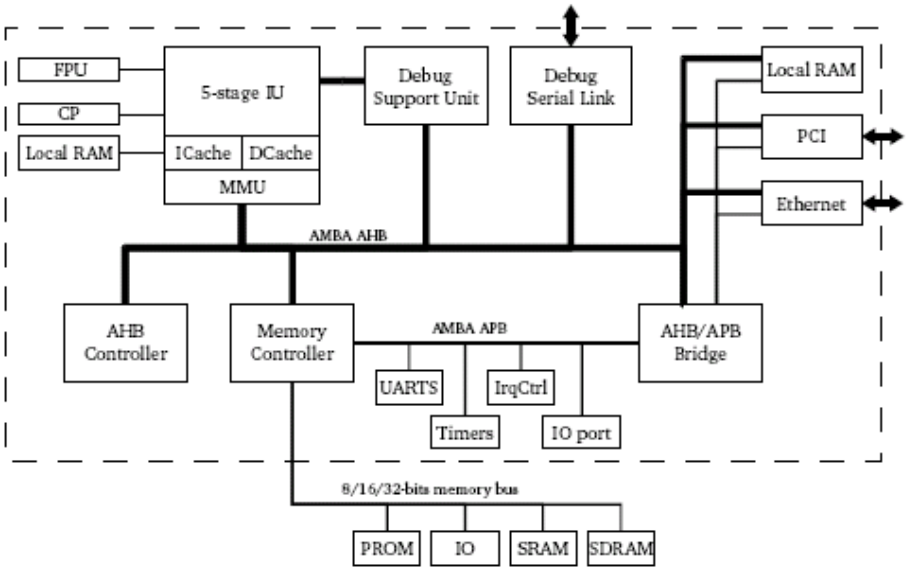

Fig. (1). LEON2 microprocessor architecture.

also has a commercial high performance FPU for LEON2 available [5], fully IEEE-754 compliant.

The LEON2 implements the following features:

- *32 bits RISC microprocessor*
- *SPARC v8 compliant*
- *5-stage instruction pipeline*
- *Multiply/divide/mac operations on hardware*
- *Separated instruction and data caches*
- *Memory management unit, MMU*
- *Memory interfaces for FLASH, SRAM,*
- *SDRAM & PROM*
- *On-chip RAM*
- *Interrupt handler*
- *Interface for a floating point unit, FPU*
- *Debug support unit, DSU*
- *Two 24-bit timers*
- *Two serial port controllers, UART*
- *16-bit I/O port mapped on memory*
- *Watchdog & power-down*
- *Ethernet controller 10/100 MAC*
- *PCI interface*
- *Co-processor interface*

LEON2 also can provide a generic interface for a custom user defined co-processor which work in parallel with the main processor in order to increase performance.LEON2 uses the AMBA-2.0 AHB bus to connect the main processor with high-speed controller like cache and memory ones and other optional units like the on chip RAM or PCI or Ethernet interfaces. In the default configuration LEON2 is the only master of the bus. Another AMBA-2.0 bus is used to access to the most on chip peripherals, the APB bus. It's optimized for simple operation and low-power consumption and it's connected to the AHB (and LEON2) *via* the AHB/APB Bridge, which is the master of that bus.LEON2 external memory access is provided by a programmable memory controller with interfaces to PROM, SRAM & SDRAM chips, providing also memory mapped I/O operation. The controller can decode a map of up to 2 Gbytes.

3. CHOICE OF THE PUBLIC KEY CRYPTOSYSTEM

Nowadays, the most commonly used public key cryptosystem on digital signature is the RSA algorithm which was developed by Ron Rivest, Adi Shamir and Leonard in 1977 [6] which is licensed by over 700 companies and has an estimated base of around 500 million users [7]. Besides the widely used RSA method, public key schemes based on elliptic curves (EC) have gained more and more importance. In 1985 elliptic curve cryptography (ECC) has been first proposed by Miller (1986) [8] and Koblitz (1987) [9]. ECC has become attractive in recent years for the following reasons:

- ✓ ECC always has a shorter key length than any known public key cryptosystems with similar strength of security. Strength of security is said to be in term of the time to break the cryptosystem
- ✓ ECC is probably more secure than RSA. The largest RSA and ECC challenges solved being 512 bits and 108 bit respectively. The solution to 108 bit ECC challenge is believed to be the largest effort ever expended in a public key cryptography challenge. It took four months and involved approximately 9.500 machines. The amount of work required to solve the problem was about 50 times of the 512 bit RSA [10].

Fig. (**2**) shows how long it takes to break the RSA and ECC cryptosystem of different key length [10]. The hard problem of RSA is factorization of a large integer while solving the discrete logarithm problem is needed to break ECC. For the same security level, the key size of ECC is much shorter than RSA's.

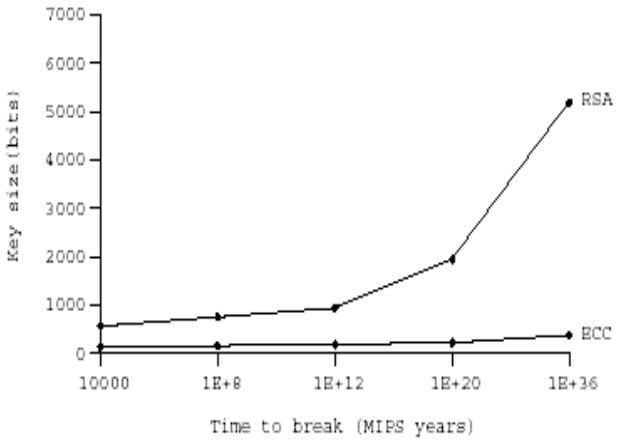

Fig. (2). Comparison of security levels ECC and RSA.

4. BACKGROUND KNOWLEDGE

We call an elliptic curve on R, every curved plane of equation $y^2=x^3+ax+b$, where the discriminate $(4a^3+27b^2)$ of x^3+ax+b is non null. We add with this curve a point at infinity noted O [11-14].

The interest of these elliptic curves is that one can provide them with an operation of commutative group. Let us take P and Q two points distinct from the elliptic curve (and different from the point at infinity O). One plots straight line (PQ) as illustrated in Fig. (**3**). Two cases can occur:

- The line cuts the curve in one 3 points (it is shown that there is at more the 3 points of intersection between a line and the curve). The symmetrical one of this 3rd to the x-axis is P+Q.
- The line cuts the curve only in P and Q this is not possible that if (PQ) is parallel to the y-axis. P+Q=O then is defined (point at infinitum).

If P=Q, one considers the tangent with the curve in P, and one defines P+P as above. Lastly, one prolongs the definition of + by posing P+O=O+P=P. One can then prove that the operation + defines a law of group on the elliptic curve. One can carry out calculations besides, to obtain the co-ordinates of P+Q according to those of P and Q.

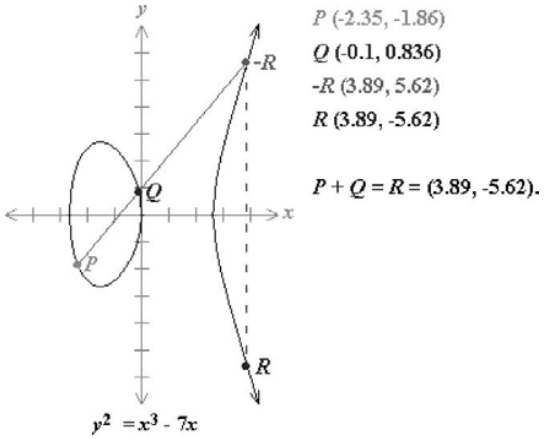

Fig. (3). Example of an EC visualizing the point addition.

Let P = (x1, y1), and $P \neq \pm Q$. with Q = (x2, y2), then P + Q = (x3, y3), and we have:

$$X3 = \left(\frac{Y_2 - Y_1}{X_2 - X_1}\right)^2 - X_1 - X2$$

$$Y3 = \left(\frac{Y_2 - Y_1}{X_2 - X_1}\right)(X1 - X3) - Y1$$

2- Let P = (X1, Y1). Then 2P = (X3, Y3), and we have:

$$X3 = \left(\frac{3X_1^2 + a}{2Y_1}\right)^2 - 2X_1$$

$$Y3 = \left(\frac{3X_1^2 + a}{2Y_1}\right)(X1 - X3) - Y1$$

Generally it is not very convenient to work with the real numbers and the implementation of an elliptic algorithm in the body R requires the use of much memory capacity for the representation of the numbers. Then, which one does, it is that one takes the formulas with multiplications and divisions and one applies them in a body easier to handle. In cryptography, the interests of elliptic curves are those defined over finite fields. There are two fields, Fp and F_2^m, mostly used in ECC. For convenience, we usually use the field Fp in ECC. Therefore, the remainder of this paper will focus on the finite field Fp, where p > 3 and is a prime.

5. PRINCIPLE OF THE ELLIPTIC SIGNATURE OVER FP

The generation of the signature as well as the checking of the latter requires the generation of a pair of keys of which one will be maintained secret and will be used for the signature and the second will be public and thus allow the checking of the signature.

The generation procedure:

1. We select a random number $d \in F_q$
2. We calculate Q = d G with $G = (x_g, y_g) \in E(Fq)$;
3. The public key will be then Q while d will be the private key

According to standard ANSI X9.62 to sign a message m with a part of keys (d, q), we follow the following procedure:

1. We select a random number k such as $1 \leq k \leq q - 1$.
2. We calculate kG = (x, y).
3. We calculate $r = x_1 \bmod n$. If r = 0 then it is necessary to return at stage 1.
4. We calculate $k^{-1} \bmod n$.
5. We calculate e = SHA-1(m). With m the message to be signed.
6. We calculate $s = k^{-1} (e + d\, r) \bmod n$, if s = 0 then we return at stage 1.
7. The signature of the message m is (r, s).

To check the exactitude of a signature, one proceeds as follows:

Table 1. Results of Compilation

Library	Library Files	Library Size	Code Size
Lcrypt.a	Mirdef.h, Miracl.h, Mrcore, Mrarth0, Mrarth1, Mrarth2, Mralloc, Mrio1, Mrio2, Mrxgcd, Mrarth3, Mrrand, Mrmonty, Mrcurv	61 ko	71ko

1. It is checked if r and s are integer in the interval [1, q-1].
2. We calculate e = SHA-1(m).
3. We calculate $w = s^{-1} \bmod n$.
4. We calculate $u_1 = ew \bmod n$. and $u_2 = r\ w \bmod n$.
5. We calculate $X = u_1G + u_2Q$ with X (x_1, x_2).
6. If X = 0 , then we reject the signature. If not, we calculate $v = x_1 \bmod n$.
7. We accept the signature if v = r.

Indeed, if (r, s) is the signature of a message m with $s = k^{-1}(e + dr) \bmod n$. one has then:

$k \equiv s^{-1} (e + d\ r) \equiv s^{-1}\ e + s^{-1}\ r\ d \equiv w\ e + w\ r\ d \equiv u_1 + u_2\ d \pmod n$.

Therefore $u_1G + u_2Q = (u_1 + u_2\ d\ )\ G = kG$, and thus v = r.

6. SOFTWARE DESIGN

Since a common approach for implementing public key cryptography is to use available open-source libraries that offers all basic cryptographic functions. We followed such approach, as a realistic one. In particular, we used MIRACL C library [15]. This library consists of over 100 routines that cover all aspects of multi-precision arithmetic and finite field operations. It includes all the primitives necessary to implement number theoretic based methods for Public Key cryptography.

To compile the Elliptic curve digital signature we use BCC, which stands for Bare-C Cross Compiler, is a C/C++ cross compiler for the LEON2 processor based on the GNU tool chain, the binutils and the standard Newlib library, with full math support and simple I/O operations (non-files).

To develop the digital signature programme, we had to develop a cryptographic library named Lcrypt based on some MIRACL files including all the necessary primitives. The Table **1** shows the needed files and the results of the compilation.

Gaisler Research has released TSIM, a complete LEON2 instruction-level simulator. TSIM can run in standalone mode or connected through a network socket to the GDB debugger, acting like a remote target using a common debugging protocol, so it can be used with another interfaces, like the graphical debugging tool DDD. TSIM is a commercial application but it's also available as an evaluation version for non-commercial uses. The Fig. (**4**), shows the simulation results of our code by TSIM.

7. HARDWARE DESIGN

The design for the hardware of our LEON2 based Smart card application includes the configuration and adaptation of the processor model to our need. So, the first step in the design process is to configure the LEON2 model with a selection of the on chip components it includes, according to our need, and also giving values to the main parameters that defines the behaviour of the core. This could be accomplished first by using a graphical configuration tool provided with the LEON2 model. This tool works on both Windows (with CYGWIN) and Unix/Linux. It is build upon the TCL/TK software, providing some menus that allow the designer to configure how the different units of LEON2 will work and which ones will be implemented in the final hardware. Second, we made some modifications in many leon vhdl files to fulfil the application requirements. The Fig. (**5**) shows the enhanced architecture for smart card based on LEON2.

```
bash-2.05b$ ./tsim.exe

This TSIM evaluation version will expire February 1, 2006


 TSIM/LEON3 SPARC simulator, version 2.0.3 (evaluation version)

 Copyright (C) 2001, Gaisler Research - all rights reserved.
 This software may only be used with a valid license.
 For latest updates, go to http://www.gaisler.com/
 Comments or bug-reports to tsim@gaisler.com

serial port A on stdin/stdout
allocated 4096 K RAM memory, in 1 bank(s)
allocated 2048 K ROM memory
icache: 1 * 4 kbytes, 16 bytes/line (4 kbytes total)
dcache: 1 * 4 kbytes, 16 bytes/line (4 kbytes total)
tsim> load sign
section: .text, addr: 0x40000000, size 67984 bytes
section: .data, addr: 0x40010990, size 2080 bytes
tsim> go
resuming at 0x40000000

r : 533BC354690C3F4673E63ABFBA3DA63C068B3BC99BE0800D
s : 5CD95F2D7E7054680A94D8401D20A04782494F93E699BD6D

Program exited normally.
tsim>
```

Fig. (4). Simulation results of digital signature by TSIM.

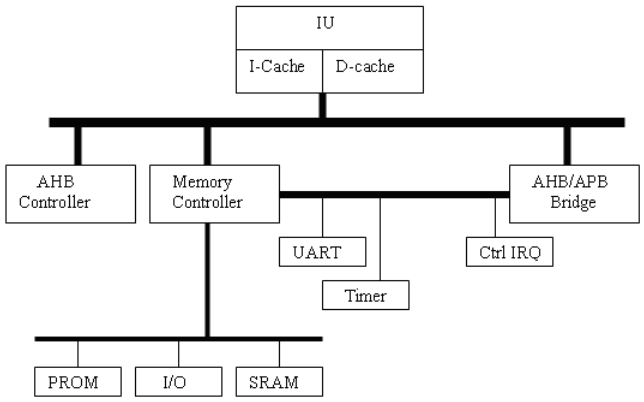

Fig. (5). The enhanced architecture for smart card based on LEON2.

Once the model is configured and enhanced to fit the actual requirements, it is possible to simulate the model by running a test bench running the digital signature algorithm.

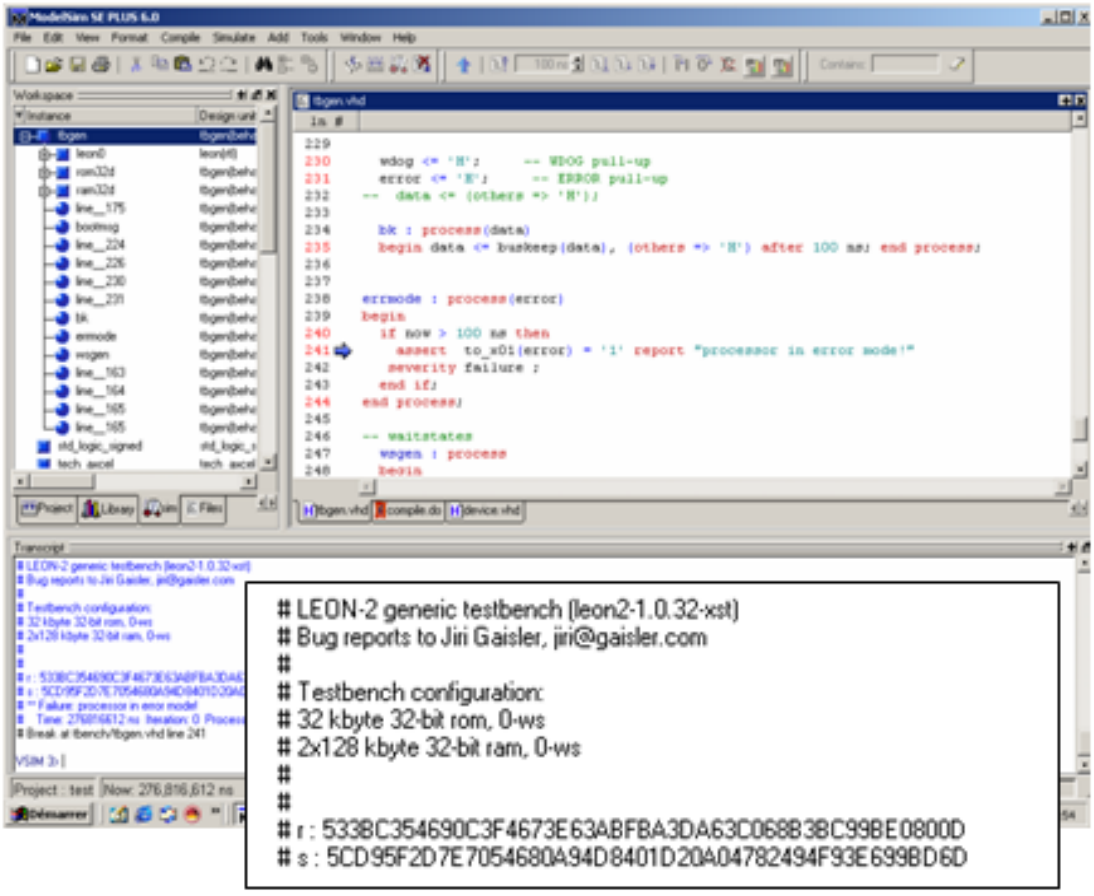

Fig. (6). LEON2 modelsim EC digital signature simulation.

This test bench is provided with the LEON2 VHDL model and it works with many major simulation tools, like Modelsim from Mentor, NCSIM from Cadence2, VSS from Synopsys or GHDL from GNU.

Fig. (**6**) shows the simulation results of the elliptic curve digital signature with the enhanced LEON2 architecture on the Modelsim environment.

8. DESIGN VALIDATION ON A REAL HARDWARE TARGET

LEON2 comes with support for various FPGA evaluation and development boards from different manufacturers, providing several configuration files and user-constraints for each model of board. But we have one board not supported by LEON2 so there's a first step which we had to complete prior to implementing the core for the board. The board used (Fig. **7**) was the "Excalibur Evaluation Kit" from Altera, with an APEX EP20K200E FPGA, and also a JTAG interface, serial port, several LEDs, buttons and switches, with a clock generator of 33,33 Mhz and with SRAM and Flash memory. We constructed a user constraints file (UCF) for that board, to map the FPGA pads (on the Evaluation Kit) to the corresponding pins of the different peripherals and memory banks. At the end of the hardware design flow we obtained a bit stream with the information to program the FPGA device with the LEON2 microprocessor core. We had to download it to the FPGA using the Alera Quartus programming tool through a JTAG cable.

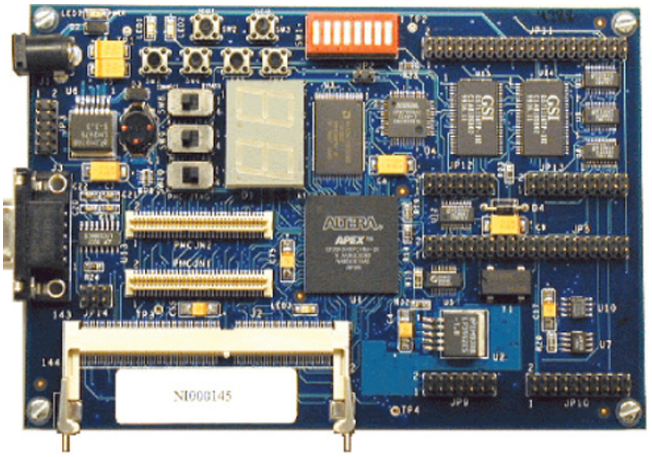

Fig. (7). Excalibur development board.

To download the software and to get it running on the LEON2 we need to use a tool from Gaisler Research which allows to communicate with the processor for a non-intrusive monitoring and debugging, providing full access to internal registers, memories and all the main peripherals. This tool, the LEON2 monitor, named GRMON, has a commercial license and an evaluation version is also available for non-commercial purposes. In order to use the LEON2 monitor we have to include the debug support unit (DSU) in the LEON2 model. This unit will provide a link for the communication

with GRMON, *via* a serial cable or a PCI interface. Fig. (**8**) show the result of the ECDSA turning on the development board using GRMON.

```
using port /dev/ttyS0 @ 115200 baud

processor frequency   :  33.18 MHz
register windows      :  8
v8 hardware mul/div   :  yes
instruction cache     :  1 * 1 kbytes, 32 bytes/line (1 kbytes total)
data cache            :  1 * 1 kbytes, 32 bytes/line (1 kbytes total)
hardware breakpoints  :  2
sram width            :  16 bits
sram banks            :  1
sram bank size        :  128 kbytes
stack pointer         :  0x4001fff8
UART 1 in DSU mode
grmon[dsu]> lo sign4
section: .text at 0x40000000, size 77664 bytes
section: .data at 0x40012f60, size 2080 bytes
total size: 79744 bytes (86.7 kbit/s)
entry point: 0x40000000
grmon[dsu]> run

r : 533BC354690C3F4673E63ABFBA3DA63C068B3BC99BE0800D
s : 5CD95F2D7E7054680A94D8401D20A04782494F93E699BD6D

Program exited normally.
```

Fig. (8). Load and run of ECDSA on the Excalibur development board.

9. CONCLUSION

Open source software, most notably in the case of the Linux operating system is now well established and enjoys widespread use in commercial environments. In the case of hardware however, open source IP has been slow to take off in IC development.

In this paper we prove that nowadays some open source IP cores could be considered as solutions for soc application development due to the maturity of some of them, in the case of the LEON2 processor and the presence of huge amount of support. This could reduce the development cost and reduce total cost of ownership and providing, also the opportunity to break free from the shackles of proprietary systems and expensive upgrades.

REFERENCES

[1] J. Gaisler, "LEON2 Processor user' manual", Gaisler Research. [Online] [Accessed Jan. 7, 2004]. Available: www.gaisler.com

[2] SPARC International Inc, "The SPARC Architecture Manual, Version 8", [Online] 1992, [Accessed Jan. 12, 2004]. Available: www.sparc.org

[3] G. Seaman, "Free Hardware Design: Past, Present & Future". [Online] [Accessed Dec. 17, 2007]. Available: www.opencollector.org/Whyfree/freedesign.html

[4] ARM Limited, "AMBA (tm) Specification, Rev. 2.0", [Online] 1999, [Accessed Jan. 22, 2004]. Available: www.arm.com

[5] E. Catovic, "GRFPU – High Performance IEEE 754 Floating-PointUnit", [Online] [Accessed Jan. 7, 2004]. Available: www.gaisler.com

[6] R. Rivest, A. Shamir and L. Adleman, "A method for obtaining digital signatures and public key cryptosystems", *Commun. ACM*, vol. 21, pp. 120-126, 1978.

[7] [Online] [Accessed Nov. 2, 2006]. Available: www.rsa.com

[8] V. Miller, "Uses of elliptic curves in cryptography", *Advances in cryptology – CRYPTO '85, Lecture Notes in Computer Science,* vol. 218, Springer-Verlag, pp. 417-426, 1986.

[9] N. Koblitz, "Elliptic curve cryptosystems", *Math Comput.*, vol. 48, pp. 203-209, 1987.

[10] [Online] [Accessed Mar. 24, 2007]. Available: www.certicom.com

[11] N. Koblitz, *A course in number theory and cryptography*. Springer Verlag, 1994.

[12] A. Menezes, *Elliptic curve public key cryptosystems.* Kluwer Academic Publishers, Boston, 1993.

[13] L. Blake, G. Seroussi, and N. Smart, *Elliptic curve in cryptography*, Cambridge: Cambridge University Press, 1999.

[14] Certicom Corp, "Current public-key cryptographic systems", Certicom whitepaper, [Online] 2000, [Accessed Jul. 15, 2007]. Available: www.certicom.com

[15] MIRACL, "Multiprecision integer and rational arithmetic C library", [Online] [Accessed Sept. 10, 2004]. Available: www.shamus.ie

Received: July 01, 2008 Revised: July 08, 2008 Accepted: July 14, 2008