A HW/SW Co-design Case Study:
Implementing a Cryptographic Algorithm
in a Reconfigurable Platform

Ivan González, Francisco J. Gómez and Javier Martínez
Laboratorio de Microelectrónica
E.T.S de Informática
Universidad Autónoma de Madrid
Cantoblanco. E-28049 Madrid
E-mail: ivan.gonzalez@ii.uam.es, francisco.gomez@ii.uam.es, javier.martinez@ii.uam.es

Abstract
In this paper, several co-design solutions are presented for the implementation of the cryptographic algorithm IDEA. A reconfigurable platform “Labomat3” based on a processor and a FPGA circuit is used to testing the throughput and relative performances of the proposed designs. Four solutions with different co-design co-dependencies are studied. The available reconfigurable resources in the Labomat3 platform, limit the HW/SW partitioning problem. Only a full IDEA round has been implemented in hardware. The data transference between processor and FPGA is the critical factor to control the throughput. From the studied cases, the best co-design solution presents a performance 125% better than the software solution.

1. Introduction
The evolution of integrated circuit technology is motivating new approaches to digital circuit design[1,2]. Nowadays, a Field Programmable Gate Array (FPGA) [3] circuit includes on a chip memory and reconfigurable resources. A platform based on FPGA supplies system level solution. Complementing the FPGA with a processor increases the design possibilities and there are circuits called System-on-Chip[4,5], integrating core processor and reconfigurable elements.
Several research and commercial platforms are available, to set an example: RC1000-PP [6] from Celoxica (Embedded Solution), CARMEN [7] from Sidsa, Riley-2[8] and Labomat3[9]. Most digital design can be viewed as a collection of several hardware and software components, whose combined operation provides a service. Hardware/Software co-design means meeting system-level objectives by exploiting the synergism of hardware and software through their concurrent design.
Our goal is to study the possibilities of the co-design techniques in a small platform, where these techniques will provide solutions for problems that will not be possible to resolve efficiently without the combination of the HW and SW synergy. The interest of this case resides in the possibility that the future generation of low prize electronic circuits for consumer application will afford the integration of a processor with a reconfigurable part inside.
As a case study, we propose the implementation of IDEA[10] (International Data Encryption Algorithm) using the labomat3 board [9]. This algorithm is used to show the tradeoffs for the suggested architectures developed using co-design techniques.
This paper is organized as follows: section 1 and 2 give a description of the IDEA algorithm and the Labomat3 platform. After that, we present three implementations using a co-design approach. We conclude our paper by pointing out some considerations and future work.

2. The IDEA algorithm as a benchmark
The International Data Encryption Algorithm (IDEA) was proposed in 1991. It is the evolution of an initial algorithm (the Proposed Encryption Standard, or PES[10]) developed by Xuejia Lai and James Massey of the Swiss Federal Institute of Technology in Zurich.
IDEA[11,12] is a symmetric-key block cipher that can encrypt 64-bits plaintexts to 64-bit ciphertexts using a 128-bit key, used for secure communications. The 128-bit key is split into eight 16-bit sub-keys, which are the first eight sub-keys. The digits of the 128-bit key are rotated 25 bits to the left to make a new key that is split into the next eight 16-bit sub-keys. The last step is repeated until those fifty-two sub-keys have been generated. Six sub-keys, $Z_i^n$ to $Z_6^n$, are used in the nth iteration and four sub-keys, $Z_7$ to $Z_{10}$, are used in the final transform.
Encryption and decryption use an identical computational scheme but generate their sub-keys differently from the same 128 bit key. The decryption sub-keys are worked out from the
encryption sub-keys being either multiplicative or additive inverses of them. More details about sub-key generation can be found in [10]. The sub-key expansion process occurs only when a key is chosen and the result can be saved for later use. Thus, the sub-key expansion requires only a negligible fraction of the total computation time.

The sub-key expansion process occurs only when a key is chosen and the result can be saved for later use. Thus, the sub-key expansion requires only a negligible fraction of the total computation time.

The algorithm involves only three simple operations: bit-wise exclusive-or, addition modulo $2^{16}$ and multiplication modulo $2^{16} + 1$. It consists of 8 computational identical rounds followed by an output transformation. The $n$th round receives a 64-bit input block as four 16-bit sub-blocks $X_1^n$ to $X_4^n$, and produces four 16-bit blocks, to $Y_1^n$, $Y_4^n$ as shown in Figure 1.

The operations performed in the $n$th round are:
1. Multiply sub-block $X_1^n$ by sub-key $Z_1^n$.
2. Add block $X_2^n$ and sub-key $Z_2^n$ to obtain $Y_2^n$.
3. Add block $X_3^n$ and sub-key $Z_3^n$ to obtain $Y_3^n$.
4. Multiply block $X_4^n$ by sub-key $Z_{10}$ to obtain finally $Y_4^n$.

IDEA is a popular cryptographic algorithm, for this reason the algorithm has been proposed as a benchmark application for reconfigurable computing[11]. As benchmark is representative of many similar problems. The algorithm has understandable performance metrics and the data flow of IDEA resembles to many signal-processing tasks, which often use 16 bit fixed-point multiplication.

3. The Reconfigurable Platform

The Labomat3 board is a platform for teaching and research on reconfigurable systems developed by the Logic System Laboratory, Swiss Federal Institute of Technology in Lausanne (Swiss).

Figure 2 is a the block diagram [9] of the board The architecture is divided in two main parts:

a) **The processor part** is built around a Motorola 32-bit MC68EN360 processor with integrated Ethernet protocol management [13]. It is connected to 512 KB of boot EPROM, 32 MB of DRAM and 4KB of dual port SRAM.

b) **The reconfigurable part** is built around a XC4013E and a XC6216 FPGA[14]. The XC6216 is connected to 128 KB of additional SRAM. The XC4013E is connected to the dual-port SRAM, which in turn is connected to the processor.

The processor has direct access to the FPGAs as peripherals through its data and address buses. Also is possible data sharing between the processor and the reconfigurable part, by using the 4KB dual-port SRAM.

A large bus of 80 signals interconnects both FPGAs and a subset (52 I/Os) is directly available on 10-pin connectors. The extension bus also holds the global data, the address bus, and the control bus. Interrupt and bus controlling is done by a programmable logic chip (MAX7128)

An on-board programmable clock generator provides a wide range of clock frequencies. Every FPGA receives two different clock signals and each signal can be assigned to a different source. The system's default clock frequency is 25 MHz. Ethernet 10Base-T and RS-232 ports are used to communicate with the outside world.

The MC68360 processor features integrated communication capabilities. Bootstrap code is executed from the EPROM. This code simply downloads the complete operating system and applications via the Ethernet interface. In addition,
there is a large amount of free software tools available for this micro-controller.

Fig. 2. Labomat3 architecture

The Labomat3 platform offers a powerful set of communication interfaces to configure and control the whole board. RTEMS [15], a pre-emptive multitasking operating system is running on board. It contains the drivers for Ethernet and provides a TCP/IP stack, which allows usage of high-level standard TCP/IP protocols.

The highlighted blocks in figure 2, represents the Labomat3 resources used in the IDEA co-design implementation proposed in this paper.

4. System Co-design considerations

Generally, co-design techniques would attempt to implement the time-consuming components in hardware. However, resource limitations can constrain the HW/SW partitioning problem.

The presence of feedback in an algorithm reduces the amount of possible pipelining and parallelism, decreasing the performance of the full hardware solution. Finally, the performance of the co-design solution will depend on the correct definition of HW/SW interfaces[16].

Although the IDEA data flow can be parallelized by processing multiple data blocks through the same pipeline, the use of feedback necessarily place an upper limit on the available parallelism. The most critical component in IDEA implementation is the 16 bit-multiplier. The lacking of a hardware multiplier in the processor architecture penalizes the software solution. A multiplier can be space prohibitive depending on the operand size. Co-design can achieve a compromise between both solutions.

The multiplication modulo $2^{16} + 1$ in IDEA is performed on 16 bit data using the following algorithm:

$$xy \mod (2^{16}+1) = (xy \mod 2^n - xy \div 2^n) \mod (2^n+1)$$

This equation maps efficiently both hardware and software[19].

A typical software implementation of IDEA in C performs data encryption at 16 Mbits/s on a PentiumPro 180 MHz machine. A hardware implementation of the algorithm speeds up the throughput. To the best of our knowledge, the results from to 126,6 Mbits/s (@ 150 MHz) to 1,5 Gbits/s (@ 20 MHz) are found[17][18].

In our case, the study is centered on the relative performance obtained with different HW/SW partitioning solutions, always implemented on the same platform. The resource limitation of this platform does not give comparable results with the previous one. However, the principal interest is to face up the global problem: data fetch up, data processing and data storage.

The interface between the different modules in the Labomat3 platform, plays an important role in the throughput of the designs. The measures of the access time to several elements are summarized in Table I.

<table>
<thead>
<tr>
<th>Element</th>
<th>Average cycles</th>
<th>Time (us)</th>
<th>Normalized Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DRAM</td>
<td>2.5</td>
<td>0.1</td>
<td>1</td>
</tr>
<tr>
<td>4KB DPSRAM</td>
<td>24</td>
<td>0.9</td>
<td>9</td>
</tr>
<tr>
<td>FPGA Access</td>
<td>10</td>
<td>0.4</td>
<td>4</td>
</tr>
</tbody>
</table>

Table I. 32 bit width access time

In order to communicate FPGA with the processor buses, a 32 bit register and the associated control logic have been implemented into the FPGA. The processor can read or write data in the register following the peripheral access protocol depicted in figure 3. An additional cycle-counter included into the FPGA gives the number of passed cycles. This
value agrees reasonably with the values measured with the time functions available in RTEMS OS.

Comparing access time to FPGA with access time to DRAM we obtain a penalization of a factor 4 in Labomat3 platform for accessing to reconfigurable resources. This difference can be explained by the delay associated to the bus controlling chip (MAX7128) managed to hide the timing specifications of bus and FPGA configuration cycles.

5. Software Implementation

Despite its computational challenges, the IDEA data flow is relatively simple. Reference implementations of IDEA are available as a source code in a variety of packages, including the appendix of Applied Cryptography[10] the Pretty Good Privacy package, and the RSAREF cryptographic library. The final criterion for a reasonable benchmark is to have useful metrics. For IDEA, there are two metrics: Latency of a single block encryption, and sustained bandwidth through the device.

For both measurements, the device should operate in Electronic Code Book (ECB) mode, which assures that there is not feedback in the data flow. It may be assumed that the sub-key expansion has already occurred.

The first benchmark metric is the performance of individual computational components, measuring the time (in ms) to encrypt and decrypt a single 512 Kbytes block of data.

The second measure is the maximum sustained bandwidth, typically measured in Mbits/s. This measure is most strongly affected by the size of the device and the availability of parallel computational resources.

By assuming a value of 46 cycles for a multiplier instruction, a theoretical estimation of the throughput for IDEA on a MC68360 processor at 25 MHz, is 0.6 Mbits/s. Running the IDEA algorithm in the processor, the execution measured time for an 1 Mbyte of data was 18711 milliseconds, corresponding to a real throughput of 0.44 Mbits/s. Remark that the measures were performed while the processor was running a multithread OS.

6. Co-design Solutions

Four implementations with different co-design dependencies are studied. Using the Labomat3 platform the fundamental constrains was the available reconfigurable HW resources. In the best case, the HW resources allows a sequential implementation of IDEA based on a full round. The test consists in encrypting and decrypting 512 Kbytes of random data. Hardware resources expended in the implementations, that will be explained later, are summarized in the next table.

<table>
<thead>
<tr>
<th>Unit Implementation</th>
<th>FPGA</th>
<th>CLB’s</th>
<th>Frequency (MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiplier Unit</td>
<td>xc4013e-4-hq240</td>
<td>307 (53%)</td>
<td>27</td>
</tr>
<tr>
<td>Cobinational Round Unit</td>
<td>Xcv1000e-8-bg560</td>
<td>1940 (7%)</td>
<td>25</td>
</tr>
<tr>
<td>Sequential Round Unit</td>
<td>Xc4036ex-2-bg304</td>
<td>1277 (98%)</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>xcv150-6-bg352</td>
<td>1940 (56%)</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>xc2x150-6-fg456</td>
<td>1940 (56%)</td>
<td>14</td>
</tr>
<tr>
<td>Sequential Idea Unit</td>
<td>xc4013e-4-hq240</td>
<td>358 (96%)</td>
<td>10</td>
</tr>
</tbody>
</table>

Table II. Hardware part implemented on FPGA

With different HW/SW partitioning the implementations proposed are:

a) Combination Multiplier Unit Implementation

The analysis of the IDEA algorithm evidences a certain dependence of the multiplier. As a first approach we decide to implement this multiplier as a hardware unit. The register interface, explained in the section 4, was also included.

Our goal is to develop a combinational multiplier, so that the time of operation is one cycle. The result obtained on XC4013 FPGA was an operation clock frequency of 27 MHz with a 53% of occupation. This range of operation speed fits with the 25 MHz microprocessors frequency. However the occupancy is too high to our proposal for developing a full round hardware unit.
The experimental results obtained for the block test were 13861 milliseconds, corresponding to a throughput of 0.6 Mbits/s. This development shows an improvement of 36% with regard to the software solution. Using the multiplier as hardware coprocessor is not a satisfactory co-design solution. However, it makes possible to improve the software solution due to the strong dependence of the IDEA algorithm on the multiplier operation.

b) Combinational Round Unit Implementation
In this solution, the FPGA implementation includes the necessary elements to complete an IDEA round. Figure 5 shows a schematic with 4 multipliers, 4 adders and 6 xor. In this case, an interface of 226 I/O pines would be necessary. The XC4013E FPGA of the Labomat3 platform has not enough number of CLB’s to fits the design. In order to estimate the necessary resources for the pure combinational round design, others FPGA were studied. The results shown in Table II, give information about the relation between speed and resource used. Actually, platforms with those FPGAs are not available in our laboratory, so experimental results expected from these FPGAs are not presented.

c) Sequential Round Unit Implementation
Due to the impossibility to introduce a whole combinational round in the XC4013E FPGA, a new design of a sequential multiplier was proposed. Some implementations in small FPGAs have already been carried out[11] where the multiplier was substituted by a simple adder and rotation multiplier. The size of this multiplier is lower, about 53 CLB’s (9% without including the interface register), but 17 clock cycles are needed to carry out a complete operation. This new design will be clearly worse than the combinational one. The necessary units for the development of a round are schematised in fig 6. Besides the multiplier, there are an adder, an xor unit, and the control logic circuit (Finite-State-Machine) to enable the registers in the appropriated clock cycles. A dual port memory implemented inside the FPGA store the six 16-bit sub-keys and the four 16 bits input data.

The result has been satisfactory, having a complete round with a 96% occupancy resources and it works at 10 MHz. The number of clock cycles necessary to complete a round was 87 cycles. The experimental results obtained for the block test were 10241 milliseconds, corresponding to a throughput of 0.82 Mbits/s. The improvement obtained is 86% better than the software solution.

d) Sequential Idea Unit Implementation
In this solution, the implementation of a full IDEA unit is achieved by sequential operation of the previous round unit. The new design needs a FSM control with more states and more intermediate register. Also all sub-keys involved in the full algorithm must be stored in register. There is not enough space to implement a Dual port memory into the FPGA with capacity to allocate this number of register. Another solution is to use the 4Kbyte DSPRAM of the Labomat3 board to store all sub-keys. The results having a complete idea unit is now a 59% occupancy resources and it works at 15 MHz. The number of clock cycles necessary to complete a round was 104 cycles.
The experimental results obtained for the block test were 8312 milliseconds, corresponding to a throughput of 1.0 Mbits/s. The improvement obtained in this case is 125% better than the software solution.

**Fig. 6.** Sequential Round Unit

7. Conclusions and Future Work

The combination of programmable resources, and level software programs, in the Labomat 3 platform provides a system that can be used to experiment with co-design techniques. A dedicated system comprising both hardware and software components, has been developed for the Idea algorithm. Results obtained in three different implementations show better performance than the SW solution. A HW design effort is accomplished by an increase in the throughput. From the studied cases, the best co-design solution represents a performance 125% better than the software solution. The data transference between processor and FPGA is an important factor to control the throughput. Future work will try to resolve the HW limitation resources by using new powerful board as RC1000PP from Embedded Solutions. In this platform, solutions with a most important HW contribution in the HW/SW partitioning could be studied.

References


