Implementation of Low Power Scalable Encryption Algorithm

K.J. Jegadish Kumar
Assistant Professor
SSN College of Engineering
Kalavakkam-603110 Chennai, India

S. Salivahanan
Principal
SSN College of Engineering
Kalavakkam-603110
Chennai, India

K. Chenna Kesava Reddy
Principal
Jyothismathi College of Engineering and Technology
Shamirpet-500078, India

ABSTRACT
SEA – Scalable Encryption Algorithm is a block cipher based symmetric encryption scheme, particularly intended for resource constrained devices. SEA proposes low computational cipher schemes, that is, miniaturized code size, memory and power, developed for processors with a restricted instruction set. SEA is parametric with plain-text, key and microprocessor size, and found to be powerful with the grouping of encipherment or decipherment and derivation of the keys. SEA was primarily meant for software implementations in microcontrollers, smart cards and small embedded systems. In this article, we look into the performance investigation of modified SEA with efficient modular adder in a Field programmable gate array (FPGA) device. For this reason, a loop based iterative design of the block cipher is realized on FPGA. Apart from its minimum cost, the proposed modified design is entirely flexible with any parameters and acquires advantage of generic VHDL coding. The efficient modular adders implementation based modification in SEA achieves lower area, power consumption and considerably higher throughputs on the target platform VIRTEX-4, xc4vl25 and SPARTAN-3, xc3s1400.

Keywords: Block ciphers, constrained applications, FPGA implementation.

1. INTRODUCTION
Symmetric encryption schemes designed for resource constrained devices have only a limited history. Tiny Encryption Algorithm (TEA) is an example of cipher designed especially for resource constrained devices. TEA is commonly known as Yuval's proposal [1,2]. Earlier cipher does not give efficient resistance to differential and linear cryptanalysis attacks. Block ciphers in recent days, like the Rijndael Advanced Encryption Standard (AES) concentrates on deciding a trade-off in information security, hardware/software complexity, and overall efficiency. Consequently, there arises a requirement for a new cryptosystem that endows with apt solution for resource constrained systems. Embedded applications that are basic building structures posses important features and a challenge for new cryptosystem like Scalable Encryption Algorithm (SEA) [1,2].

1.1 SEA: An Overview
The purpose for SEA [1, 2] is to implement in restricted resources; the architecture proposed is parametric with respect to plain-text, cipher-text, key and the processor size. Since the architecture is parametric in nature, it has flexibility and can be implemented in all platforms with minimum code change. Most algorithms perform differently on different platforms but SEA is an exception as it allows obtaining an optimal method of ciphering that suits any given microprocessor and the security of this cipher is tailored according to its key size [2]. Since it operates on a limited resource processor, it only does some basic operations (i.e. XOR, AND, OR, mod 2^b addition).

1.2 Literature Survey
Though there are many cryptographic algorithms, most of them require high or moderate processing power, like Advanced Encryption Standards (AES) [3-6], Data Encryption Standard (DES) [7], Tiny Encryption Algorithm (TEA) [8,9], Extended TEA (XTEA) [10]. But these encryption algorithms cannot be implemented in a resource constrained system due to various complexities involved like i.e. Non Scalability, Processor Intensive, and Security Level.

AES (Rijndael) [3-6] comprises three block ciphers, all the block ciphers vary depending on the number of bits. AES is a predetermined block cipher having 128 bits. It has different key size of 128, 192 and 256 bits. AES requires four 256 entry, 32 bit tables, so totally 4096 of memory which equals 1kilo byte for each table. AES is more processor intensive and is non scalable, so it cannot be implemented on constrained systems. Though there are efficient implementations of AES, there are still non-scalable for need of any processing platform.

DES [7] is based on symmetric key algorithms of bit size 56. DES is the classic symmetric key encryption algorithm that receives a predetermined length sequence of plain-text bits and alters through a sequence of complex tasks performed in a different bit string known as cipher-text bits using the same key. Though DES is not a secured encryption, it is widely used in a mode of operation as per Federal Information Processing Standard (FIPS-81). DES is more processor intensive, non scalable and is breakable by Linear Cryptanalysis.

TEA [8] or Yuval's proposal [9] is notable for its simplicity and implementable on various platforms (scalability) and works on block size of 64 bits with a key size of 128 bit. When crypt analyzed with equivalent key, each key gives three other keys. So in terms of security TEA is insecure. XTEA [10] was an advanced version of TEA, mainly aimed at improving all the security glitches. XTEA has complex key scheduling and rearrangement of Shift XOR and addition operations. XTEA is vulnerable to related key differential attack. Like SEA, HIGHT

[11] is also another Block Cipher for resource constrained systems, but its non scalable and consumes more number of gates and the throughput and operating frequency are much less when compared to SEA (48, 8). So, implementing SEA for constrained systems is a better option.

This paper is organized as follows. The introduction and literature survey are provided in section 1. Section 2 describes parameters, definitions and basic operations for implementation of SEA. Section 3 illustrates hardware implementation strategies. In section 4 implementation results are presented. Finally, in section 5 conclusions based on the results are made.

2. IMPLEMENTATION OF SEA

Majority of recent private key cipher design resulted in tradeoffs in the cost of execution and their performances. However, the objective is to effectively implement on a wide range of platforms. SEA is a different method and considers a perspective in which it has inadequate throughput and resources. The cipher was primarily targeted as a design to provide cost effective implementation and certification routine for processors with a restricted instruction set [1]. In addition, opposite to block ciphers that are widely used, it considers the parameters such as bus sizes, key and plain-text. Hence it can be tailored straightly to a variety of security requirements and implementation environments. In contrast, in comparison with the solutions that runs traditional ciphers like Yuval's proposal or TEA (Tiny Encryption Algorithm), SEA additionally promotes a resistance to cryptanalysis [2]. When put into practice, SEA was demonstrated to be a proficient answer for microcontrollers and related applications. Although, SEA is an efficient hardware implementation, its performance is yet to be explored. This paper therefore proposes to investigate the interest of this algorithm to be modified for area and power in constrained applications. The investigation begins with an exploration of the quality of a cost effective FPGA implementation of SEA [1] and our progress is to modify SEA using efficient modular adder in [12-14] to reduce the hardware complexities in terms of area and power.

2.1 Algorithmic Description

2.1.1 Basic Operations

Owing to its optimality constraints, $SEA_{n,b}$ [1,2] is based on a restricted number of basic operations chosen for their accessibility in the given device. This can be classified into different categories as given in Standert et.al [1,2]

1. Basic XOR $\oplus$

$$\oplus: \mathbb{Z}_2^{\frac{n}{2}} \times \mathbb{Z}_2^{\frac{n}{2}}: x, y \to z = x \oplus y \Leftrightarrow$$
$$z(i) = x(i) \oplus y(i), \quad 0 \le i \le \frac{n}{2} - 1$$

2. S-Box: $SEAn_{,b}$ make use of 3-bit substitution table stated as:

$$S_T: = [0;\ 5;\ 6;\ 7;\ 4;\ 3;\ 1;\ 2],$$

and is evaluated as in following expressions

$$S: \mathbb{Z}_{2^b}^{n_b} \to \mathbb{Z}_{2^b}^{n_b}: x \to x = S(x) \Leftrightarrow$$
$$x_{3i} = (x_{3i+2} \wedge x_{3i+1}) \oplus x_{3i}$$
$$x_{3i+1} = (x_{3i+2} \wedge x_{3i}) \oplus x_{3i+1}$$
$$x_{3i+2} = (x_{3i} \vee x_{3i+1}) \oplus x_{3i+2},$$

$\wedge$: bitwise AND

$\vee$: bitwise OR.

3. Word rotate R: For n_b-word vectors the word rotate is expressed as:

$$R: \mathbb{Z}_{2^b}^{n_b} \to \mathbb{Z}_{2^b}^{n_b}: x \to y = R(x) \Leftrightarrow$$
$$y_{i+1} = x_i, \quad 0 \le i \le n_b - 2$$
$$y_0 = x_{n_b - 1}$$

4. Bit rotate r:

$$r: \mathbb{Z}_{2^b}^{n_b} \to \mathbb{Z}_{2^b}^{n_b}: x \to y = r(x) \Leftrightarrow$$
$$y_{3i} = x_{3i} \ggg 1$$
$$y_{3i+1} = x_{3i+1}, \quad y_{3+2i} = x_{3i+2} \lll 1, \quad 0 \le i \le \frac{n_b}{3} - 1$$

where $\ggg$: right shifts

and $\lll$: left shifts.

5. Addition modulo 2^b $\boxplus$: $\boxplus: \mathbb{Z}_{2^b}^{n_b} \times \mathbb{Z}_{2^b}^{n_b} \to \mathbb{Z}_{2^b}^{n_b}: x, y \to z = x \boxplus y \Leftrightarrow$
$$z_i = x_i \boxplus y_i, \quad 0 \le i \le n_b - 1$$

In the following sub-section, complete description of the algorithm which can be referred to [1, 2] is briefed. It starts with the vital parameters, and then highlights its basic operation. Then the sequence of key generation is described.

2.1.2 Encryption/Decryption and Key Generation

The encrypt round F_E, decrypt round F_D and key round F_K are defined as:

Encryption Round $\mathbf{F_E}$:

$$[L_{i+1}, R_{i+1}] = F_E(L_i, R_i, K_i) \Leftrightarrow R_{i+1} = R(L_i) \oplus r\big(S(R_i \boxplus K_i)\big),$$
$$L_{i+1} = R_i$$

Decryption Round $\mathbf{F_D}$:

$$[L_{i+1}, R_{i+1}] = F_D(L_i, R_i, K_i) \Leftrightarrow R_{i+1} = R^{-1}\Big(L_i \oplus r\big(S(R_i \boxplus K_i)\big)\Big),$$
$$L_{i+1} = R_i$$

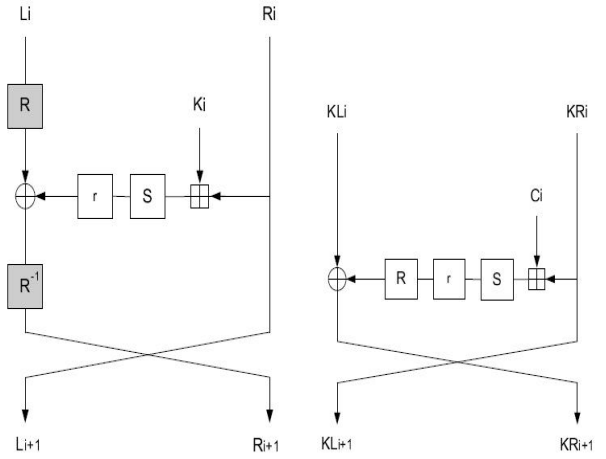

Figure.2.1. Encryption and decryption and key scheduling [1, 2]

Key Scheduling Round $\mathbf{F_K}$:

$$[KL_{i+1}, KR_{i+1}] = F_K(KL_i, KR_i, C_i) \Leftrightarrow KR_{i+1} = KL_i \oplus R\left(r\left(S(KR_i \boxplus C_i)\right)\right),$$
$$KL_{i+1} = KR_i$$

2.1.3 Cipher Description

This ciphering is based on the number of rounds n_r and uses iterative based loop design. The pseudo code given in Figure 2.2 illustrates the necessary steps for encrypting a plain-text.

where, P: Plain-text,
C: Cipher-text
K: Key
and all these three are parameterized by bit size n. Taking into account the parametric b-bit words, the operations are done in the cipher.

Since n_r is odd, referring to Figure 2.2 for key scheduling and encryption, the value of n_r must be rounded up or down. Rounding up or ceil is denoted as ⌈⌉ and rounding down or floor is denoted as ⌊⌋.

```
C=SEAn,b(P,K)
{
% initialization:
        L0&R0 = P;
        KL0&KR0 = K;
% key scheduling:
for i in 1 to ⌊nr/2⌋
[KLi,KRi] = FK(KLi−1,KRi−1,C(i));
switch KL⌊nr/2⌋, KR⌊nr/2⌋;
for i in ⌈nr/2⌉ to nr − 1
[KLi,KRi] = FK(KLi−1,KRi−1,C(r − i));
% encryption:
for i in 1 to ⌈nr/2⌉
[Li,Ri] = FE(Li−1,Ri−1,KRi−1);
for i in ⌈nr/2⌉ + 1 to nr
[Li,Ri] = FE(Li−1,Ri−1,KLi−1);
% final:
C = Rnr&Lnr ;
switch KLnr−1, KRnr−1;}
```

Figure 2.2: Pseudo Code description [1, 2]

2.1.4 Recommended Number of rounds

The expression $n_r = [3(n/4) + 2((n/2b) + b/2)]$ evaluates the number of rounds, n_r, required to sufficiently secure against conventional attacks. i.e, calculation of n_r is related to resist either differential or linear attacks adding to twice the number of rounds to get absolute diffusion. Consecutively, it prevents statistical and structural attacks. The value of n_r must be always odd, if not, 1 must be added to make it odd [1,2].

3. HARDWARE IMPLEMENTATION

3.1 Implementation Description

The first investigation step to the [1] hardware implementation of SEA proposes to take a look at a straightforward implementation of the algorithm on an FPGA platform, achieving a round/clock cycle and denoted as the loop implementation. It is known that the S-boxes and the mod 2^b adder are the operators that cost more in hardware implementation; but the operators like Word Rotate and Bit Rotate blocks in the cipher are realized by swapping wires. As per the specifications of SEA [1], the key generations consist of two multiplexers to switch the right and left part of the key when the algorithm reaches half. The execution is done by the suitable control signal called Swap. The switch controlled multiplexer supplies the looping function with the right part of the key during the execution of first half of round and pass on its left part after the switch. The Generic Loop Architecture is simple and only changes in the location of the R and R^{-1} Block. In this paper, we mainly focus to describe [12] a light weight Modular adders to modify modulo 2^b addition operator in SEA so as to achieve considerable low power optimization at the synthesizable VHDL design level.

3.2 Implementation of Modified SEA with efficient Modular adders

Addition modulo m of x and y $\in \{0,1,2,\ldots..m-1\}$ is given by the equation:

$(x + y) \bmod m = x + y$, if $x + y < m$, and

$= x + y - m$, if $x + y \geq m$, (3.1)

which can be simply realized using a suitable basic arithmetic operators. But, their implementation is complex, as they require more space and speed. The algorithm described allows to liberate the implementation cost and thus results in powerful hardware operators. Here, the equation $k = [\log_2 m]+1$ designates the number of bits which are essential to predetermine output and inputs of a modulo m adder. Fundamentally an addition modulo m can be performed by three methods [12]:

1. Table based operator method
2. Hybrid based operator method
3. Adder-Based operator method.

Adder-Based Operators: Implementing Equation (3.1) and illustrated by Algorithm 3.1 directs to attain architectures in Figure 3.1 and [12] delivers in detail the proof of validation of this method. This algorithm implementation resulted in use of two carry-propagate adders and a Multiplexer and is considered to be appropriate for FPGAs.

Algorithm 3.1 Addition Modulo m.

Determine k $\forall\ 2^{k-1} < m < 2^k$

Assign $s_0 \leftarrow x + y$

Assign $s_1 \leftarrow (s_0 \bmod 2^k) + 2^k - m$

if the carry-out bit of s_0 or s_1 is one **then**

$(x + y) \bmod m \leftarrow s_1 \bmod 2^k$

else

$(x + y) \bmod m \leftarrow s_0 \bmod 2^k$

end if

The architecture of implemented algorithm 3.1 is portrayed in Figure 3.1. Our proposed modulo addition algorithm leads to smallest circuits, in turn reduced overall circuit complexity of SEA.

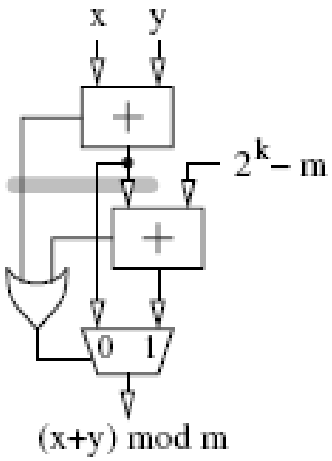

Figure 3.1

4. IMPLEMENTATION RESULTS

The results were derived by synthesizing the algorithm using Xilinx ISE 9.2i tool on VIRTEX-4 platform device XC4VLX25 with speed grade-12 and XC3S1400, SPARTAN-3 platform with speed grade -4. XPower Analyzer tool was used to analyze the power consumption of the implementation. The implementation was done for variants bit data (n) and a processor word size (b). We achieved reduction in number of slices (Figure 4.1), high throughputs (Figure 4.2) and increase in work frequency (Figure 4.3) in implementation compared to implementation of [1]. Throughput /Area ratios are also shown in Figure 4.4. Our implementation of SEA exhibited consumption of small area and move toward at the cost of increased throughput and reduced power consumption as in Figure 4.5. As a result, it can be well thought-out as the attractive substitute for constrained devices.

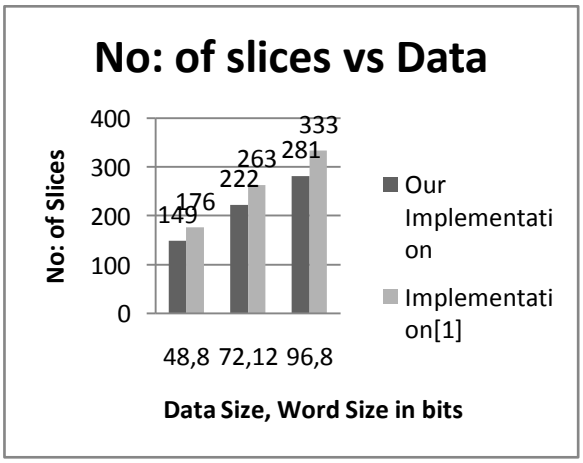

Figure 4.1

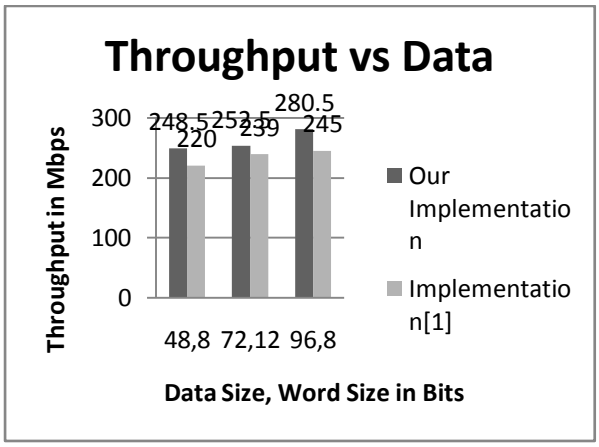

Figure 4.2

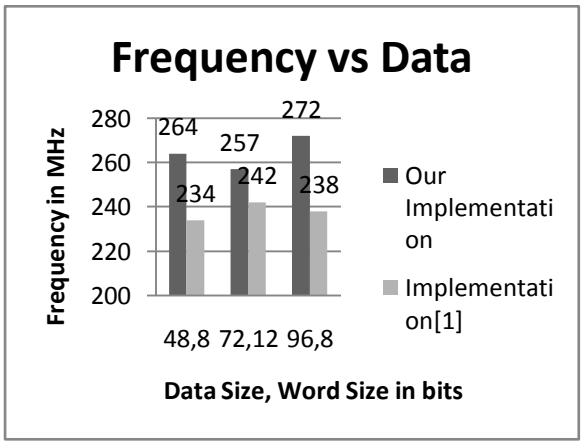

Figure 4.3

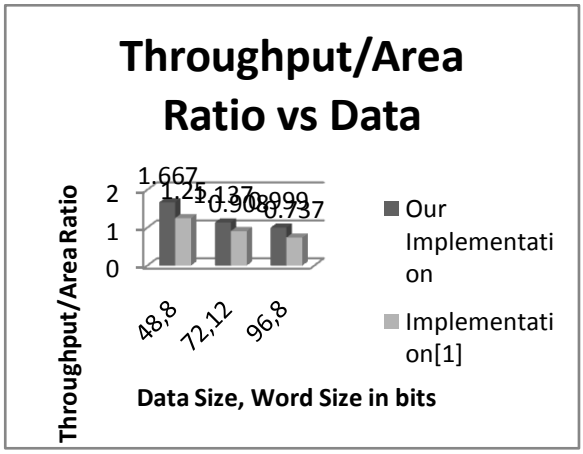

Figure 4.4

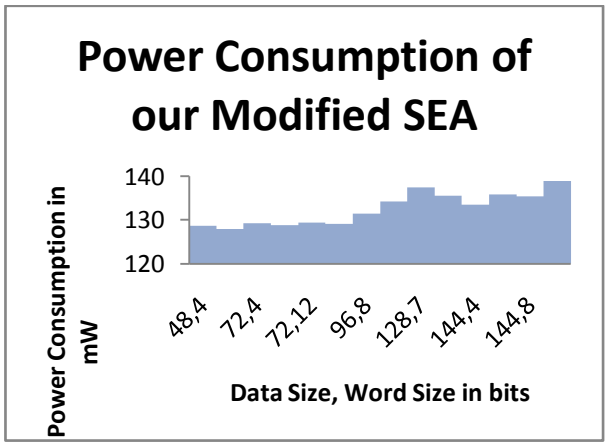

Figure 4.5

5. CONCLUSION

SEA was originally proposed for efficient implementation in software. Considering the need for efficient implementation in hardware, with the new design criteria, it was found to have better solutions as compared to software implementation. Through the hardware investigation of the SEA, we show that this modular symmetric algorithm, targeted for low-resources software solutions, can interestingly respond to constrained hardware needs. We first demonstrated that the scalability of this algorithm can be kept in the hardware description language(VHDL). The simple iterative loop design achieves interesting performance in area and power reduction, improvise throughputs in FPGA. In addition, we analysed the power consumed by the SEA module for different variants in data block and word size. It is also significant to highlight a number of merits in SEA compared to recent block ciphers, specifically its simplicity, scalability(re-implementing SEA for a new block size does not necessitate to re-write code), and being a splendid design of encryption and decryption.

6. ACKNOWLEDGEMENT

We are grateful to the experts who have contributed towards development of our work. We acknowledge Shankar Kuhan, Thirumurugan and Praveen V for all their help during the design phase.

7. REFERENCES

[1] F.Mace, F.X Standert, J J Quisquater "FPGA implementation(s) of a Scalable Encryption algorithm" IEEE Transactions on VLSI Systems, Vol.16, 2008, pp. 212-216.

[2] Francois-Xavier Standaert, Gilles Piret, Neil Gershenfeld, Jean-Jacques Quisquater "SEA a Scalable Encryption Algorithm for Small Embedded Applications" in Proc. CARDIS, 2006, pp 222-236.

[3] J. Daemen and V. Rijmen, The Design of Rijndael. New York: Springer-Verlag, 2001.

[4] Advanced Encryption Standard, FIPS PUB 197, Nov. 2001.

[5] N. Pramstaller and J. Wolkerstorfer, "A universal and efficient AES co-processor for field programmable logic arrays," in Proc. FPL, 2004, pp. 565–574.

[6] Francisco Rodriguez-Henriquez, N.A. Saqib, A. Diaz-Perez, Cetin Kaya K09, "Cryptographic Algorithms on Reconfigurable Hardware", Springer Series on Signals and Communication Technology, 2006.

[7] Data Encryption Standard, FIPS PUB 46-3, Oct. 1999.

[8] D.J. Wheeler, R. Needham, TEA, a Tiny Encryption Algorithm, proceedings of FSE 1994, Lecture Notes in Computer Science, Vol 1008, pp 363-366, Leuven, Belgium, December 1994, Springer-Verlag.

[9] G. Yuval, "Reinventing the travois: Encryption/MAC in 30 ROM bytes," in Proc. Fast Softw. Encryption (FSE), 1997, pp. 205–209.

[10] J.P. Kaps, "Chai-Tea, Cryptographic Hardware Implementations of XTEA, The 9th International Conference on Cryptology in India – INDOCRYPT 2008, LNCS 5356, pp. 363-375, 2008.

[11] D. Hong et al., "HIGHT: A New Block Cipher Suitable for Low-Resource Device," Proceedings of CHES 2006, Lecture Notes in Computer Science, Vol. 4249, pp. 46-59, Yokohama, Japan, October 2006.

[12] Beuchat, J.-L.; Lab. De l'Infonnatique du Parallelisme, "Some Modular adders and multipliers for Field programmable Gate arrays", in Proc. Parallel and Distributed processing symposium 2003.

[13] J.-L. Beuchat. "Modular Multiplication for FPGA Implementation of the IDEA Block Cipher", Technical Report 2002-32, Laboratoire de l'Informatique du Parall´elisme, Ecole NormaleSup´erieure de Lyon, 46 All´ee d'Italie, 69364 LyonCedex 07, Sept. 2002.

[14] J.-L. Beuchat and A. Tisserand. Small Multiplier-based Multiplication and Division Operators for Virtex-II Devices.In M. Glesner, P. Zipf, and M. Renovell, editors, *Field-Programmable Logic and Applications – Reconfigurable Computing Is Going Mainstream*, number 2438 in Lecture Notes in Computer Science, pages 513–522. Springer, 2002.