Design of A Novel Asynchronous Reconfigurable Architecture for Cryptographic Applications*

Kang Sun
College of Computer
Science and Technology
Zhejiang University
Hangzhou, 310027, China
sunk@vlsi.zju.edu.cn

Xuezeng Pan
College of Computer
Science and Technology
Zhejiang University
Hangzhou, 310027, China
xzpan@cs.zju.edu.cn

Jiebing Wang, Jimin Wang
College of Computer
Science and Technology
Zhejiang University
Hangzhou, 310027, China
wangjm@vlsi.zju.edu.cn

Abstract

Cryptographic algorithms are usually compute-intensive and more efficiently implemented in hardware than in software running on general-purpose processors. However, systems which use hardware implementations have significant drawbacks: they are unable to respond to flaws discovered in the implemented algorithm or to changes in standards. By taking advantage of FPGA technology, some work offers high performance and flexible solutions for cryptographic algorithms. But FPGAs still have some drawbacks. To overcome these shortages of FPGA, such as redundant routing resources which increase chip area and power consumption, a novel asynchronous reconfigurable cryptographic engine (ARCEN) is introduced. In this architecture, reconfigurable cryptographic array is the kernel. It routes signals asynchronously between adjacent cells through Neighbor-to-Neighbor wires with 4-phase handshaking protocol. Computation circuit for reconfigurable cell is developed with modified DSDCVS logic. On the implementation of cryptographic algorithms such as AES, the architecture shows a better performance than FPGA.

1. Introduction

Cryptographic algorithms can be implemented in hardware by ASICs (Application Specific Integrated Circuits) or in software by software-programmed microprocessors. Due to the diversity of applications, cryptographic machines have to meet the enormous computing demands of the algorithms. Moreover they also have to be flexible enough to adapt to diverse security parameters (e.g., cryptographic algorithms, cryptographic operation mode, key length). Such flexibility is also crucial for adapting to the evolving requirements of state-of-the-art algorithms and standards. ASIC-based solutions, lacking flexibility, provide effective performance but they can only offer a fixed number of algorithms to system designers. Software solutions can provide the required flexibility but they are inadequate for high speed encryption applications.

Reconfigurable computing is intended to fill the gap between hardware and software, achieving potentially much higher performance than software, while maintaining a higher level of flexibility than hardware. Reconfigurable hardware is a general term that applies to any device which can be configured, at run-time, to implement a function as a hardware circuit. It typically consists of a set of computing elements connected by communication medium. Both the computing elements and the communication medium are programmable. The computing elements can be either fine-grained or coarse-grained, which can exploit fine grain and coarse grain parallelism available in the application. Exploiting this parallelism provides remarkable efficiency compared to conventional microprocessors. And the configurability provides significant flexibility compared to ASICs.

In the past several years, there has been a growing body of work on using reconfigurable devices to implement cryptographic algorithms. Reconfigurable implementations of DES and RSA have all achieved significant speedups over general-purpose processors[1]-[3]. In some recent research work, reconfigurable hardwares have been integrated into System-on-a-Chip (SoC) as cryptographic engines. In [4], a reconfigurable elliptic curve cryptosystems on a chip has been designed and the experiment results show over 2000 times speedup when compared with the general-purpose processor solutions. The SHA-2 hash algorithm family has also been implemented successfully on reconfigurable hardware[5]. To show flexibility and adaptability, the cryptographic engine for IPSec architectures[6]

*This work is supported by Natural Science Foundation of Zhejiang Province (Grant No. Y105355).

Proceedings of the First International Multi-Symposiums on Computer and Computational Sciences (IMSCCS'06)
0-7695-2581-4/06 $20.00 © 2006 IEEE

can be employed to implement all the 5 candidate cryptographic algorithms of advance encryption standard (AES) with different parameters. All the work has shown that cryptographic algorithms can run on reconfigurable hardware efficiently. However, all the work mentioned above employ FPGAs (Field-Programmable Gate Arrays) as reconfigurable cryptographic engine. Although FPGAs can provide good performance and flexibility, they still have some drawbacks. Firstly, as fine-grained reconfigurable device, FPGAs usually need a large amount of configuration data, which will lenghthen the configuration time. Secondly, as general-purpose reconfigurable device, FPGAs require abundant routing resources in order to adapt to various applications, which will increase the chip area and power consumption[7].

Some reconfigurable fabrics different from FPGAs are also developed to implement cryptographic engines. Cryptographic algorithms are usually compute-intensive applications. Compared with FPGAs, some coarse-grained reconfigurable architectures may be more suitable for this field. PipeRench is one of these architectures. It supports hardware virtualization, and it is optimized to create pipelined datapaths for word-based computations[8]. Experiments show that PipeRench is well suited to many cryptographic tasks. GARP[9] and Chimaera[10] are also examples of such work.

In this paper, we introduce a novel adaptive cryptographic engine. Different from the stated preceding work, we develop a coarse-grained asynchronous reconfigurable architecture with high-performance for cryptographic applications. Without global clocks, asynchronous circuits can be totally prevented from the inferiority of clock-skew. And the problem of clock tree power consumption has been eased off. Moreover, in asynchronous circuits, there is no global timing signal which can be used as a reference clock, thus timing and power analysis attack are consequently expected to be more difficult[11]. The architecture we developed is called ARCEN (Asynchronous Reconfigurable Cryptographic ENgine).

The rest of this paper is organized as follows. Section 2 describes the architecture of our cryptographic engine. Section 3 presents the implementation details of this device. Section 4 analyzes the performance of this architecture and provides the experiment results. At last, Section 5 is the conclusion.

2 System Architecture

The proposed asynchronous cryptographic engine (ARCEN) is shown in figure 1. It consists of five main components:

- A random number generator, which is mainly used for generating secret keys.

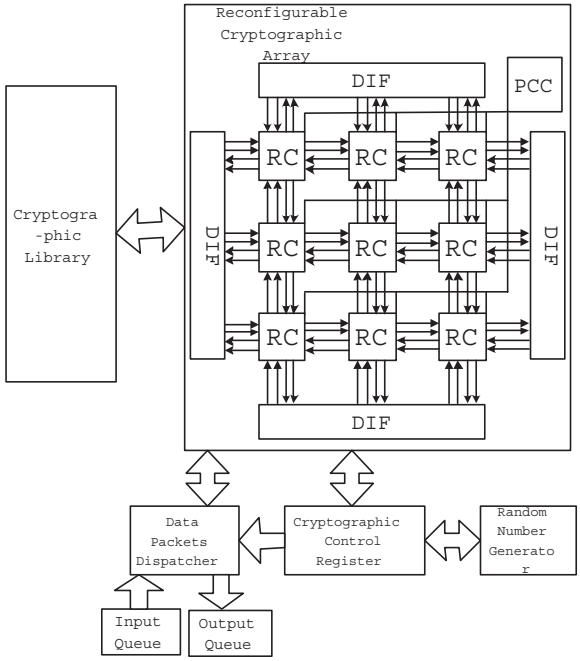

Figure 1. Architecture of ARCEN

- A data packets dispatcher, which is responsible for data moving between reconfigurable array and the outside. Before executing a task, the dispatcher will select data packets from input queue, take cryptographic parameters out from control register, and send them to the reconfigurable cryptographic array. After processing, it will get data packets from the array and send them to the output queue.

- A cryptographic control register, which is the control unit of the system.

- A reconfigurable cryptographic array, which is the computation core of the whole system.

- A cryptographic library, which stores the ARCEN configuration of cryptographic algorithms.

The reconfigurable cryptographic array (RCA) is the core component of this system. With this component, ARCEN can be dynamically adapted to cryptographic algorithms of different secret keys.

The RCA is SRAM-based, asynchronous reconfigurable computing device and consists of logic cells surrounded by NN (Nearest Neighboring) channels (see figure 1). It consists of a parallel configuration controller (PCC), four data input interfaces (DIP) and a set of reconfigurable logic cells (RC). PCC is used for configuring the reconfigurable cells. DIPs are responsible for data transferring. Because of the regularity of cryptographic algorithms, the routing

resources are the NN connections between the logic cells. The routing requirement can be satisfied by NN connections in most cryptographic applications. Even a small amount of long wires exist, they can be substituted with several NN connections. Compared with FPGA, this architecture can save more routing area. The datapath of the logic cells is 8-bit wide and the cells communicate with the nearest neighboring cells from ESWN (East, South, West, North) directions through a pair of data-wires which are 8-bit wide, dual-rail encoded. Since there is no clock in RCA design, logic cells have to use asynchronous hand-shaking protocol when communicating. Besides dual-rail encoded data-wires, NN channel between each couple of the nearest neighboring cells needs a pair of acknowledge-wires to implement the asynchronous communication.

The structure of RCA logic cell is shown in figure 2. Each logic cell is composed of input router, function unit and output router. Input router channels from physical input ports of four nearest neighboring directions (EIN, SIN, WIN, NIN) to three internal logical channels (A, B, C) as the inputs of the function unit. The input router is implemented by several MUXs.

Output router is the part that channels data from inside to the physical output port of logic cell (EOUT, SOUT, WOUT, NOUT). Besides being outputted through function unit (F, Z), input data from the four nearest neighboring input ports of logic cell (EIN, SIN, WIN, NIN) can also be channeled to the output ports directly through output router. It means that logic cell can be used not only for logical/arithmetic functions, but also as route resource.

Function unit is the crucial part of logic cell, composed of a series of various operation modules that take charge of specific logical/arithmetic functions. In order to design an efficient function unit, we have enumerated some common operations in most ciphers and discussed how they map onto reconfigurable hardware.

- Simple arithmetic operations such as addition and subtraction appear frequently in cryptographic algorithms. These operations can be realized by integrating arithmetic unit into the function unit.
- Narrow and unusual bit-widths operations often appear in stream ciphers. These operations are often implemented inefficiently in coarse-grained reconfigurable systems. They are usually mapped on coarse-grained hardware at the cost of wasting the data path width of some logic cells.
- Multiplication is a difficult operation to perform in hardware. It can be realized by many different ways. In our reconfigurable hardware, it will be performed by the adders and shifters in a group of logic cells.
- Logic operations such as XOR appear frequently in ciphers. They can be mapped on the hardware by integrating logic operation unit into the function unit.
- Most block ciphers include S-boxes. They are table lookup operations. Reconfigurable hardware can store S-box values in external scratch memory.
- Rotation and shifting operations in cryptography can be implemented by integrating shifters in the function unit.

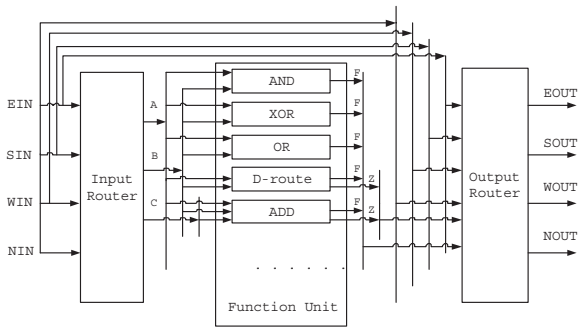

Figure 2. The structure of logic cell

The RCA as mentioned, its function unit is designed to support up to 10 operations. Among these 10 operations, ”and”, ”or” ”xor” ”shift” operations are used for basic logical functions; ”add” and ”sub” operations take charge of fundamental arithmetic functions; ”zero”, ”one”, ”2-1mux” and ”d-router”(dynamic router) are supposed to provide some control logic resources in some special applications.

3 Implementation

This section details the implementation of the reconfigurable cryptographic array (RCA). As mentioned in section 2, the key of RCA design is the function unit built by asynchronous circuit, which can be separated into control circuit and operation circuit. In following sections, each of them will be discussed respectively.

3.1 Asynchronous Handshake Protocol

Since there is no clock in RCA, logic cells have to use handshake protocol to send and receive data on NN channels. In our design, we choose 4-phase handshake protocol (see in figure 3). In 4-phase handshake protocol, only the rising of ”Req” signal can inspire the transfer process, but its control circuit is simple, and this protocol is well suited for DCVSL (Differential Cascode Voltage Swing Logic)[12].

We adopt dual-rail encoding technology to implement 4-phase handshake protocol with simple control circuit. In

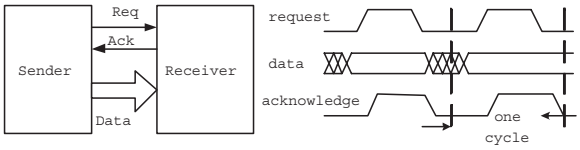

Figure 3. 4-phase asynchronous handshake protocol

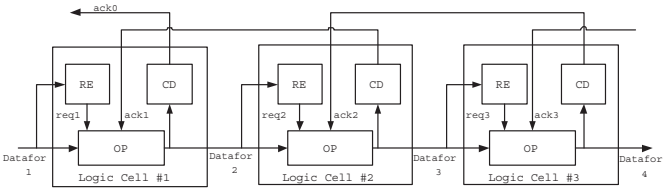

Figure 4. structure of asynchronous micropipeline

dual-rail encoding, each bit of data is implemented by two wires, of which one is the original data, and the other is the complement of it. There are 4 states of these two wires, "01", "10", "00" and "11". Both "01" and "10" represent valid data and inspire the "Req" signal rising, while the state of "00" means invalid data and set the "Req" signal falling. "11" is an illegal state, which is not supposed to appear.

3.2 Control Circuit Design

In short, the role that control circuit plays is to implement dual-rail encoding 4-phase handshake protocol. After configuration, logic cells in RCA form the structure of asynchronous mircopipeline, which is indicated in figure 4. In each logic cell, there are three main parts, namely, OP, CD and RE. OP is the operation circuit taking charge of computation logic, which is implemented by DCVSL and will be discussed later, and the other two parts constitute the control circuit. RE (Request Enabler) is used for generating req signal, whose active falling edge can make the operation circuit pre-charged (reset). Meanwhile, CD (Completion Detector) is used for detecting the state of operation circuit and generating ack signal, which is asserted high when operation circuit has finished an evaluation or low when it has finished a pre-charge phase. The functioning process of the synchronous micropipeline will be analyzed in detail in section 4.

The following terms are used in the discussion of latency issues:

- $t_{op(n)}$: evaluation time of a computation of operation circuit in logic cell n.

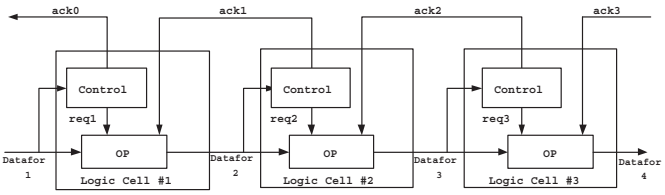

Figure 5. Asynchronous micropipeline with improved control circuit

- t_{cd}: processing time of CD (Completion Detector).
- t_{re}: processing time of RE (Request Enabler).
- $t_{pre(n)}$: pre-charge time of operation circuit in logic cell n.
- $t_{d(n)}$: delay of signal transition on wires between logic cell n and logic cell n+1.
- $t_{cyc(n)}$: the time interval which logic cell n should be ready for evaluating a new data after it has started to evaluate the previous one.

The $t_{cyc(n)}$ of asynchronous micropipeline which is shown in figure 4 can be described as equation (1):

$$t_{cyc(n)} = t_{op(n)}+t_{op(n+1)}+t_{pre(n)}+2t_{re}+2t_{cd}+2t_{d(n)}+t_{d(n+1)} \quad (1)$$

Since $t_{op(n)}$ and $t_{pre(n)}$ are specified by operation circuits, and $t_{d(n)}$ is determined by the routing resources of RCA, the factors which can reduce $t_{cyc(n)}$ are t_{re} and t_{cd}. We place the CD part before operation circuit, and parallel the parts of RE and CD, which can merge t_{re} and t_{cd} into the same time interval. The improved structure of asynchronous micropipeline is shown in figure 5. In this case, both t_{re} and t_{cd} are replaced by $t_{control}$, then the $t_{cyc(n)}$ can be described as equation (2):

$$t_{cyc(n)} = t_{op(n)} + t_{pre(n)} + 2t_{control} + 2t_{d(n)} \quad (2)$$

Obviously, only when the following constrain is satisfied, can this improved structure of asynchronous micropipeline function.

$$t_{cd} + t_{d(n)} > t_{op(n)} \quad (3)$$

Figure 6 shows the improved control circuit design. As mentioned in section 2, the logic cell is designed to support up to 3-input (A, B, C) operations, so we should consider the req signal of each input respectively. Figure 6(a) indicates the circuit generating the req signal of input A, which is extended into A+ and A- by dual-rail encoding. When both A+ and A- are low (represent invalid data), reqA is

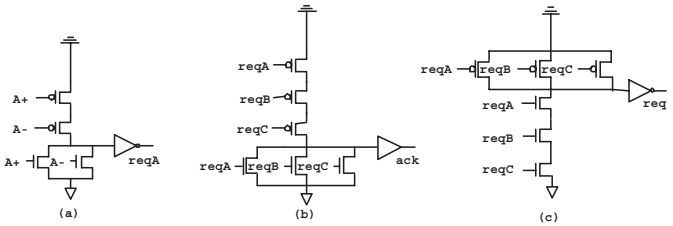

Figure 6. Implementation of improved control circuit

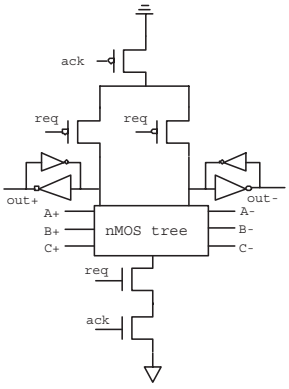

Figure 7. DSDCVSL used for operation circuit

driven to high. Otherwise, when valid input datum A arrives (A+, A-=10 or 01), reqA remains low. In short, when each valid input data for operation circuits arrives, its related req signal asserts low. In figure 6(b), the circuit generating ack signal is presented. Only when all the req signals (reqA, reqB, reqC in 3-input computation) assert low, which represents all the valid inputs (A, B, C) have arrived, can ack signal be active to acknowledge the previous logic cell. It is totally opposite in the circuit for generating req signal for operation circuit, stated in figure 6(c). Only when all the req signals are high, which means all the 3-inputs are invalid, can req signal be asserted to low to pre-charge the operation circuit. Otherwise, the signal of req remains high to keep the operation circuit working.

3.3 Operation Circuit Design

Since well suited for dual-rail encoding, DCVSL is taken into consideration for implementing operation circuit in RCA. We propose a new DSDCVSL (Data-Driven DCVSL) for synchronous pipeline with the advantages of high-speed, simple-control and low-cost[13][14]. The usage of DSDCVSL structure for operation circuit is shown in figure 7. There must be a series of circuits in a logic cell to support various operations. They can be implemented by NMOS

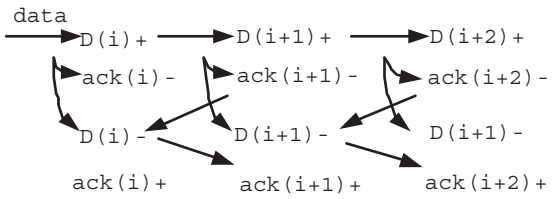

Figure 8. STG of asynchronous micropipeline

tree with different logic functions. In figure 7, the ack and req signals are responsible for setting the state of operation circuit.

4 Evaluation and Results

4.1 Performance Evaluation

The functioning process of asynchronous micropipeline formed by logic cells in RCA is shown in figure 8 by signal transition graph (STG). In figure 8, the transition of ack is represented by ack+ and ack-, of which "+" means the transition from low to high and "-" means high to low. D+ and D- represent the state of operation circuit, of which "+" means evaluation and "-" means pre-charging. (i) represents the Logic Cell of stage i in asynchronous pipeline, while (i+1) and (i+2) represent its successive two logic cells. The arrows dedicate the constrain relationships of signals of neighboring logic cells. For instance, the arrow between ack(i+1)- and D(i)- means that before the operation circuit of logic cell i being pre-charged, the signal of ack from logic cell (i+1) to logic cell i must be inspired from high to low. The minimum cycle of one logic cell from the asynchronous micropipeline has been stated in equation (2). The latency of total asynchronous micropipeline, which is the main measurement of RCA, can be described as equation (4).

$$T = \sum_{i=1}^{n} \left(t_{op(i)} + t_{d(i)} \right) \tag{4}$$

4.2 Experiment Results

To show the effectiveness of our cryptographic engine, we have conducted several tests of simulation, which are based on the technology library of SMIC in 0.18 CMOS process. We choose the 5 candidate algorithms of AES mentioned in [6] as our test suite. All these ciphers are mapped on the hardware manually. The hardware size is 16×16. Table 1 contains the throughput and estimated power consumption reported by Synopsys HSPICE. Figure

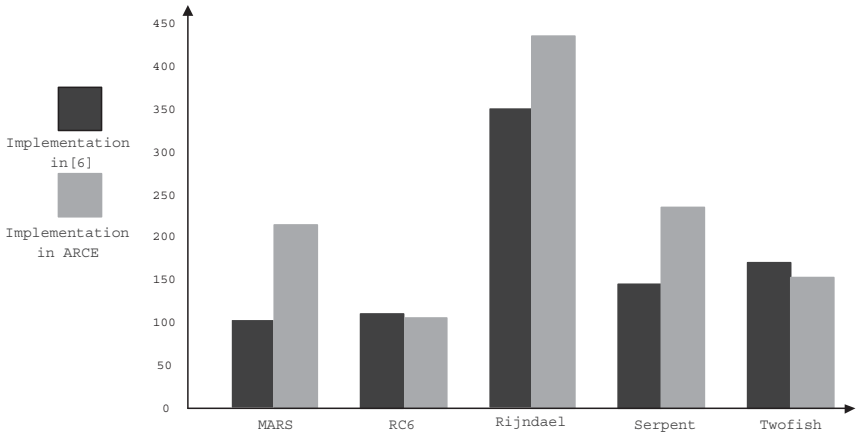

Figure 9. Estimated throughput comparisons of our ARCEN and [6]

9 describes the comparison of throughput with [6]. Table 2 shows the area requirement of our method and [6] respectively. Since architecture of these two devices are different, we only list the number of cells (or slices) consumed by these two solutions.

Table 1. Time and power performance implementation results of ARCEN.

Cipher	Throughput (Mbits/sec)	Estimated Power (mW)
MARS	213.20	103.35
RC6	100.38	95.58
Rijndael	439.23	108.75
Serpent	232.65	97.61
Twofish	153.62	125.49

Table 2. Hardware area implementation results

Cipher	ARCEN (logic cells)	ACE presented in [6] (FPGA slices)
MARS	143	6896
RC6	172	2650
Rijndael	212	5673
Serpent	113	2550
Twofish	238	9363

5 Conclusions and Future Work

We introduce a novel asynchronous reconfigurable cryptographic engine (ARCEN) and discuss the design of ARCEN to achieve high performance. By advancing the completion detector of control circuit, a modified structure of asynchronous micropipeline based on DSDCVSL is proposed. The analysis and experiment results have proven that ARCEN functions well with both high-performance and low-power consumption.

Future work includes designing a dynamic partial reconfigurable architecture to achieve further savings in area and to reduce the reconfigure time of the device. Furthermore, how to implement reconfigurable hardware by asynchronous circuits is another focus of our research work. More experiments and further improvement are needed to make the most advantage of the asynchronous circuits in power design. And we will go deeper into the research of using asynchronous circuits to defend timing analysis and power analysis attack, thus to enhance chip security.

References

[1] K. W. Tse, T. I. Yuk, and S. S. Chan. Implementation of the data encryption standard algorithm with FPGAs. Proceedings of the 3th International Workshop on Field-programmable Logic and Applications, Oxford, England, September 1993. p.412-419.

[2] J. Leonard and W. H. Mangione-Smith. A case study of partially evaluated hardware circuits: Key-specific DES. Proceedings of the 7th International Workshop on Field-programmable Logic and Applications, London, UK, September 1997. p.151-160.

[3] M. Shand, J. Vuillemin. Fast implementations of RSA cryptography. Proceedings of the 11th Symposium

on Computer Arithmetic, Windsor, ONT, Canada, 29 June-2 July 1993. p.252-259.

[4] Ray C.C. Cheung, Wayne Luk, Peter Y. K. Cheung. Reconfigurable Elliptic Curve Cryptosystems on a Chip. Proceedings of the conference on Design, Automation and Test in Europe, Vol. 1, Munich, Germany, March 07-11, 2005. p.24-29.

[5] N. Sklavos, O. Koufopavlou. Implementation of the SHA-2 Hash Family Standard Using FPGAs. The Journal of Supercomputing, Vol. 31, No. 3 March 2005. p.227-248.

[6] A. Dandalis, V. K. Prasanna. An Adaptive Cryptographic Engine for Internet Protocol Security Architectures. ACM Transactions on Design Automation of Electronic Systems, Vol. 9, No. 3, July 2004. p.333-353.

[7] R. W. Hartenstein, T. Hoffmann, U. Nadeldinger. Design-Space Exploration of Low Power Coarse Grained Reconfigurable Datapath Array Architectures. Proceedings of the 10th International Workshop on Integrated Circuit Design, Power and Timing Modeling, Optimization and Simulation, Gottingen, Germany, September 13-15, 2000. p.118-128.

[8] R. R. Taylor, S. C. Goldstein. A High-Performance Flexible Architecture for Cryptography. Proceedings of the the 1st Workshop on Cryptographic Hardware and Embedded Systems, Worcester, MA, USA, August 12-13, 1999. p.231-245.

[9] J.R. Hauser and J. Wawrzynek. Garp: A MIPS processor with a reconfigurable coprocessor. Proceedings of the the 5th IEEE Symposium on FPGA-Based Custom Computing Machines, April 16-18, 1997. p.24-33.

[10] S. Hauck, T. W. Fry, M. M. Hosler, and J. P. Kao. The Chimaera reconfigurable functional unit. Proceedings of the the 5th IEEE Symposium on FPGA-Based Custom Computing Machines, April 16-18, 1997. p.87-96.

[11] Z. C. Yu, S. B. Furber, L. A. Plana. An investigation into the security of self-timed circuits. Proceedings of the 9th International Symposium on Asynchronous Circuits and Systems. IEEE Computer Society, 2003. p.206-215.

[12] C. S. Choy, J. Butas, J. Povazanec et al. A New Control Circuit for Asynchronous Micropipelines. IEEE Transaction on Computer, vol. 50, No. 9, 2001. p.992-997.

[13] S. Mathew, R. Sridhar. A data-driven micropipeline structure using DSDCVSL. Proceedings of the IEEE 1999 Custom Integrated Circuits Conference, San Diego, CA, USA, May 16-19, 1999. p.295-298.

[14] S. Mathew and R. Sridhar, Data-driven self-timed differential cascode voltage switch logic, Proceedings of the 1998 IEEE International Symposium on Circuits and Systems, Vol: 2, May 31-June 3, 1998. p.165-168.