Low power implementation of Trivium stream cipher

Mora Gutiérrez, J.M[1]. Jiménez Fernández, C.J.[2], Valencia Barrero, M.[2]

[1]Instituto de Microelectrónica de Sevilla, Centro Nacional de Microelectrónica(CSIC). Sevilla,España,
[2]Departamento de Tecnología Electrónica. Instituto de Microelectrónica de Sevilla /Universidad de Sevilla. Sevilla,España

Abstract. This paper describes a low power hardware implementation of the Trivium stream cipher based on shift register parallelization techniques. The design was simulated with Modelsim, and synthesized with Synopsys in three CMOS technologies with different gate lengths: 180nm, 130nm and 90 nm. The aim of this paper is to evaluate the suitability of this technique and compare the power consumption and the core area of the low power and standard implementations. The results show that the application of the technique reduces power consumption by more than 20% with only a slight penalty in area and operating frequency.

1 Introduction

Cryptography provides techniques, mechanisms and tools for secure private communication and authentication on Internet and other open networks. It is almost certain that in the coming years every bit of information flowing through a network of any kind will have to be encrypted and decrypted. All devices connected to a network should therefore incorporate mechanisms that implement cryptographic functions to ensure safe transfers. With this in mind, it is necessary to design and implement hardware structures which are suitably efficient in terms of area, operating frequency and power consumption. An additional challenge is that implementations must be constructed to withstand cryptographic attacks launched against them by adversaries who have access to both primary channels (communication) and secondary channels (power consumption, electromagnetic radiation, etc.).

Cryptographic algorithms are either symmetrical - those based on the existence of a secret key - or asymmetric - those based on the existence of pairs of public and private keys. Both types play important roles in current applications. Public key-based cryptography was invented by Diffie and Hellman in 1976 [1]. Asymmetric algorithms use a different key to encrypt and decrypt (a public key and private key). The public key can be known to all, while the private key is known only by the receiver of the message. The message is encrypted using the public key and can only be decrypted using the corresponding private key. With this type of cryptography there are

no problems with key distribution, but the high complexity of the algorithms requires large computing resources and hardware solutions are complex and involve high power consumption.

In private key-based cryptography, the key for encrypting and decrypting is the same and it is secret. Private key-based algorithms can be divided into two groups: block ciphers and stream ciphers. Block ciphers encrypt blocks of data of fixed length while stream ciphers encrypt an amount of data of arbitrary length. These types of algorithms are problematic in that the keys must be distributed between the sender and the receiver. However, they have the advantage that they are fast and their implementations are simple, so they are used both to encrypt and to decrypt large amounts of data, as in applications requiring low-complexity algorithms (for example short range wireless encryptions). From the hardware point of view, the resulting implementations are of very low complexity. This makes them ideal for portable devices with low computing capacity and very high power consumption restrictions.

These needs led the European Union to launch an initiative known as eSTREAM [2][10], proposing new stream ciphers which, in both software and hardware, would meet current needs in the field of stream ciphers. The initiative has now identified and published four new algorithms specially designed for implementation in software (HC-128, Rabbit, Salsa 20/12 and Sosemanuk) and three designed to give optimal performance in hardware (Grain, Mickey and Trivium)[3].

Secure data transfers are also becoming increasingly necessary on devices with low computing capacity and which also require low power consumption. For these devices the best option is to use secret key cryptographic algorithms. Of all the possible algorithms available we have chosen the Trivium stream cipher, because it is endorsed by the eSTREAM project and is simple to implement. Furthermore, power consumption can readily be reduced in its architecture through the technique of logical shift register parallelization. With this technique we have obtained implementations with more than a 20% drop in power consumption and virtually no losing performance.

This paper is organized as follow. Section 2 briefly describes the specification algorithm, Trivium. In section 3 the low power implementation is described, showing the synthesis and power consumption reports and comparing them with the standard version. Finally, Section 5 concludes the paper.

2 Trivium Specification

Trivium is a synchronous stream cipher designed to generate up to 2^{64} bits of key stream from an 80-bit secret key and an 80-bit Initialization Vector (IV). The architecture of this cipher is based on a 288-bit cyclic shift register accompanied by an array of combinational logic (AND, OR and exclusive-or) to provide its feedback [4][10].

Firstly the cipher's 288-bit internal state is initialized using the secret key and the initial value, which are loaded into the internal state register. The state is then updated four times 288 without producing key stream bits. This is summarized in the VHDL code below:

```
state(92 downto 0) <= X"0"  & key; -- first register
state1(76 downto 93) <= X"0" & iv; -- second register
state(287 downto 177) <= "111" & X"0"; -- third register

state(92 downto 0) <= state(91 downto 0) & t3
state(176 downto 93) <= state(175 downto 93) & t1;
state(287 downto 177) <= state(286 downto 177) & t2;
```

After that, the key stream generation consists mainly of an iterative process which updates some bits in the state register with logic operations to generate one bit of key stream. The VHDL code description is given by the following lines:

```
k1 <= state(65) XOR state(92);
k2 <= state(161) XOR state(176);
k3 <= state(242) XOR state(287);

a1 <= state(90) AND state(91);
a2 <= state(174) AND state(175);
a3 <= state(285) AND state(286);

t1 <= k1 XOR a1 XOR state(170);
t2 <= k2 XOR a2 XOR state(263);
t3 <= k3 XOR a3 XOR state(68);

keystream <= k1 XOR k2 XOR k3;
```

The schematic representation of the Trivium radix-1 algorithm (which outputs one key stream bit in every clock cycle) consists of three circular shift registers of different lengths and any combinational logic to perform the addition (exclusive-or cells) and multiplication (and cells) operations as the schematic representation version shown in fig.1. The length of each shift register is different; the first register has 93 bits, the second 83 bits and the third 111 bits. The key stream is the result of exclusive-or operations on some bits in the shift register [9].

The main cost (area and power) of this implementation is in the registers. Trivium radix-1 is a good choice for evaluating the validity of the power reduction technique. Other radix versions allow fast, power-efficient implementations with the same number of storage elements but involve more combinational logic and more complex designs [4].

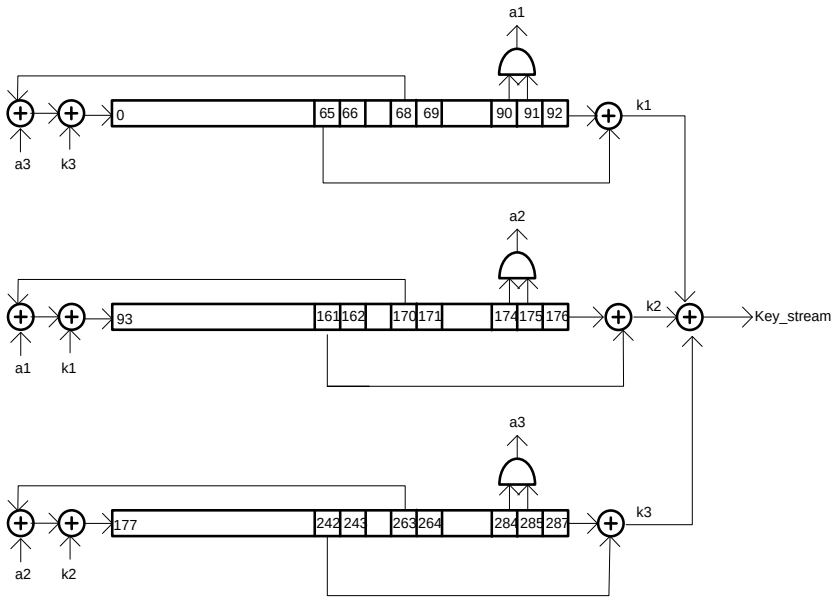

Fig. 1. Trivium schematic representation.

3 Power reduction in Trivium by Logic parallelization

The Trivium low power implementation presented here is based on shift register logic parallelization [5], [6], [7]. The main idea of this technique is to divide a shift register in two, each half-length being like the original. During shift register operation, the data coming in even cycles is stored in one register and that arriving in odd cycles is stored in the other. One of the registers is called the “even register” and the other the “odd register”. In this way, the clock which triggers the loading of each of these registers has a frequency which is half the original frequency.

Figure 2 graphically shows the structure of the register division. The odd group is synchronized with the clock rise time, while the even group is synchronized with the clock fall time. In every clock cycle each shift register stores a new bit and places another bit in its most significant position. Of the most significant bits supplied by the registers even and odd bit must be selected that provides the output. To do so, a multiplexer is used, controlled by the clock signal of the registers.

Power reduction is achieved because in each clock cycle only half of the data is shifted. Thus, each piece of data has to pass through half the number of flip-flops to reach the output. However, it is necessary to introduce additional circuitry to divide the clock, and a multiplexer to select the data in the output. This technique is therefore especially effective in reducing power consumption in medium-sized or large shift registers.

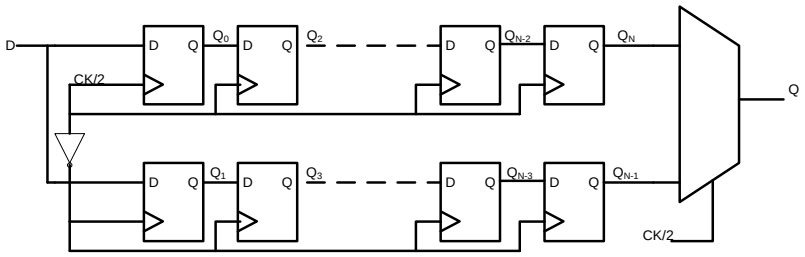

Fig. 2. Parallel Shift Register.

The application of this technique to the Trivium implementation requires some modifications. Firstly, as shown in Fig.1, the Trivium implementation has three different circular shift registers. With the parallelization technique these are separated into six shift registers, as is shown in Fig 3. Each shift register is divided into an odd and even shift register with half bits. The length of each shift register is indicated in the figure inside the register.

Secondly, generation both of the input bits in each of the shift registers and the output bits depends on the bits stored in different positions in the shift registers. These positions, depending in turn on the clock cycle, will match a bit set in the even or in the odd register. It is therefore necessary to introduce a logic that allows these bits to be selected correctly.

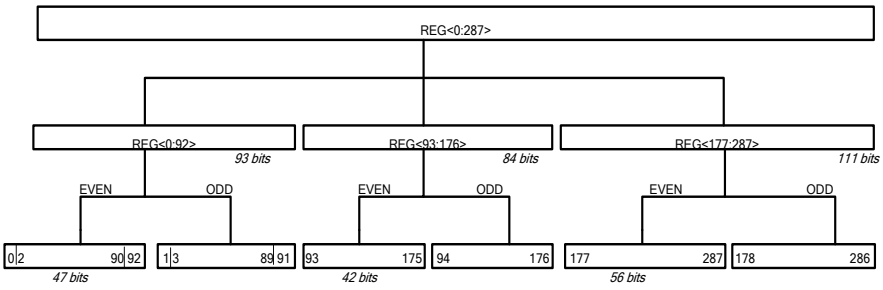

Fig. 3. Even and odd shift registers.

As shown in Fig. 4, this logic basically means introducing multiplexers which, using the clock signal as the selection signal, will select the bit to be taken from the shift register.

Thirdly, in the first clock cycles both the key and initialization vector must be loaded in parallel. In this implementation, the load must be maintained in parallel, but taking into account that the even registers are loaded with the rising edge and the odd registers are loaded with the falling edge of a clock with half the frequency of the original clock. To solve this loading problem we decided to use two clock cycles,

with the even registers being loaded in the first cycle and the odd registers being loaded in the second.

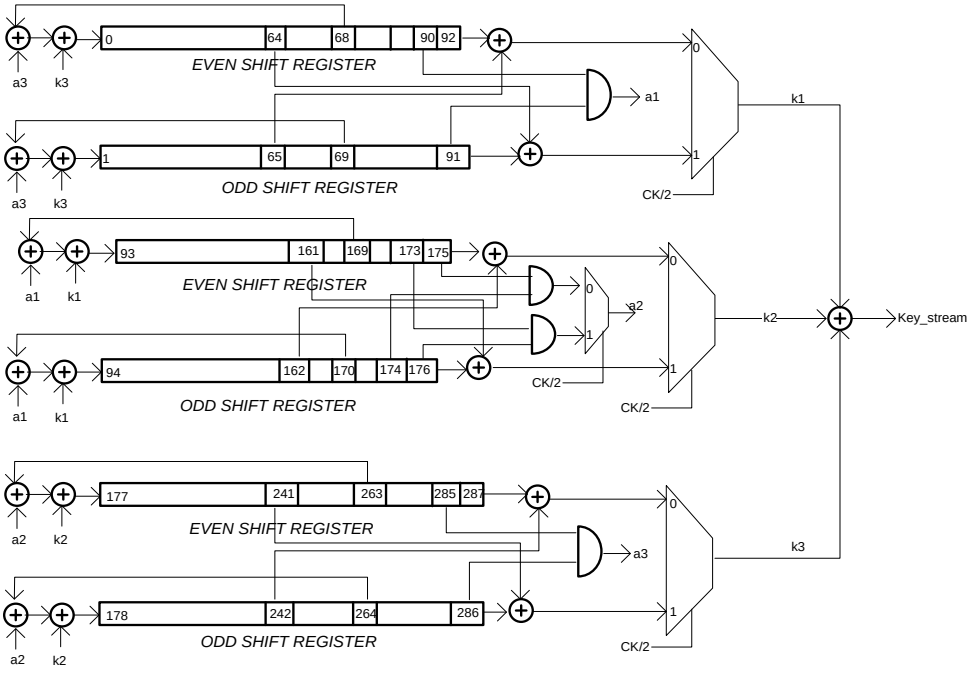

Fig. 4. Schematic of Low Power Trivium implementation.

Other low power Trivium implementations of were described beforehand. The implementation described in [8] uses a clock gating technique and a radix-16 datapath. Although the clock gating technique is widely used, it is not very useful when applied to shift registers. In fact, in this article no data comparisons between low power and standard implementations are shown.

3.1 Trivium implementations

The low-power and standard versions of Trivium were described and designed using VHSIC Hardware Description Language (VHDL). The resulting implementations were verified using the Mentor Graphics ModelSim simulation environment, with test vectors using a key presented in the reference files of Trivium [2]. The technologies used were CMOS 180 nm, 130 nm and 90 nm. The standard version of Trivium was also implemented, so the results obtained by both implementations can be compared.

The timing and area reports for the low power (LP) and standard (TRIV) versions provided by the Design Vision synthesis tool are shown in Table 1. The timing and area data depends on the standard cell library of each technology. Our aim is not to

compare the results of the three technologies, but to check that a significant reduction in power consumption is achieved in them.

The low power version uses more standard cells (4-7%) than the standard version, although the number of cells decreases when the technology is smaller. This is because the synthesis tool chooses the cells from the cells available in the library, and the libraries are different in the three technologies. The non-combinational logic area must be quite similar in both designs, because the number of flip-flops do not change (only the flip-flops for the clock division are introduced). But in 130 and 90 nm the synthesizer uses different flip-flops, with less area in the low power version than in the standard version, so the non-combinational area is similar or less. In conclusion, when the technology decreases the gate size the cell count rises slightly by 6.6% in 180nm and by 3.5% in 90 nm in the low power version and the cell area penalty is not as huge as might be expected since the cell area increases by 6.4% in 180 nm and by 2.2% in 130nm but decreases by 0.6% when the technology is 90nm because it uses flip-flops with less area from the technology library.

With respect to delay, table 1 shows that both implementations achieve the 40 ns restriction imposed on the clock. The multiplexer delay slows the low power implementation down slightly in comparison with the standard implementation.

Table 1. Synthesis Report.

Report		*180nm*		*130nm*		*90nm*	
		TRIV	***LP***	***TRIV***	***LP***	***TRIV***	***LP***
Timing(ns)	***Critical Path Length***	2.72	1.08	2.20	0.34	1.35	0.18
	Critical Path Slack	37.19	38.86	37.60	39.50	38.58	39.73
	Critical Path Clk Period	40.00					
Cell Count		604	644	608	637	602	623
Area(μm^2)	***Combinational Area***	8265	8634	2724	3401	1747	2191
	Noncombinational Area	14694	15795	9677	9281	5644	5154
	Cell Area	22959	24429	12401	12682	7392	7345

3.2 Power Consumption

Power consumption in the design was calculated with Synopsys tools and a switching activity file in SAIF (switching activity interchange format) format, generated with simulations in Modelsim with typical case timing analysis for a desired clock rate of 20 MHz. Power modelling was performed using the typical foundry values for each process.

Power dissipated can be divided in the Synopsys´s report into cell leakage (static) and dynamic power. Static power is the power consumed by a gate when it is not switching. It is caused by currents that flow through the transistors when they are turned off and it mainly depends on the size of the circuit. Dynamic power is the power dissipated when the circuit is active. Dynamic power is further divided into net switching power and cell internal power.

Net switching power is the power dissipated by net interconnects and gate capacitance. The amount of net switching power depends on the switching activity (and therefore related to the operating frequency) of the cell. The more logic transitions in the cell output, the higher the switching power. Lowering the circuit size and reducing the supply voltage also directly reduces dynamic power. Cell internal power is consumed within a cell by charging and discharging internal cell capacitances. Internal power also includes short-circuit power. In these technologies static power is much lower than dynamic power.

When comparing the power consumption of the two implementations, shown in Table 2, we noticed that the low power version always has lower cell internal power consumption than the standard version: 41% lower in 180 nm, 33% in 130 nm and 25% in 90 nm. Cell internal power consumption therefore decreases with smaller technologies. However, net switching power increases in the low power version (16%, 77% and 70%) compared to the standard version due to the rise in the number of net connections. These different switching power values increase in the three technologies due to the cells chosen by the synthesis tool for the combinational part.

The most useful comparison was made when the total dynamic power of the standard and the low power Trivium designs was evaluated. The results show that dynamic power consumption in the low power version is lower, in all technologies, than dynamic power consumption in the standard version. In 90nm it decreased by a factor of about 18%; in 130 nm, by a factor of 23% and in 180 nm by 29%.

Table 2. Power Report.

Power@20MHz	***180nm***		***130nm***		***90nm***	
	TRIV	***LP***	***TRIV***	***LP***	***TRIV***	***LP***
	Core Voltage = 1.8V		***Core Voltage = 1.2 V***		***Core Voltage = 1.2 V***	
Cell Internal (μW)	868	550	218	145	203	152
Net Switching (μW)	139	162	18	33	16	27
Total Dynamic (μW)	1007	712	236	178	219	179
Cell leakage (nW)	89.9	102.9	249.7	290	17.6	19.27

4 Acknowledgments

This work was partially funded by the following projects: Cripto-Bio, Junta de Andalucía (Andalusian Regional Government) (P08-TIC-03674); MOBY-DIC, EEC (FP7-ICT-2009-4) and CITIES, MCINN (TEC2010-16870/MIC).

5 Summary and conclusions

In this paper we have described and compared a new low power Trivium implementation based on logic parallelization techniques. We have synthesized the imple-

mentation in three technologies with different gate lengths - 180nm, 130nm and 90 nm - to compare area penalty and power consumption results.

The area's penalty and cell number obtained with this technique is very low (less than 6%) while the improvement in dynamic power consumption is quite high (more than 18%). For instance, in 180 nm a reduction of 29% was obtainable in dynamic power with an area penalty of only about 6.4%. The best solution was in 90 nm, where dynamic power was reduced by a factor of 18% and cell area decreased slightly by a factor of 0.6%.

We think this solution might be used in designs where the slight increase in cell number requirements is less important than power reduction. In lower technologies it could be used for passively-powered applications like RFID tags, smart cards and Bluetooth products.

6 References

1. Diffie, W. y M.E.Hellman. "New directions in cryptography", *IEEE Transactions on Information Theory 22* (1976), pp. 644-654.
2. eSTREAM: ECRYPT Stream Cipher Project, http://www.ecrypt.eu.org/stream/
3. Ver http://www.ecrypt.eu.org/stream/D.SYM.3-v1.1.pdf
4. C. De Canniere y B. Preneel, "Trivium, A Stream Cipher Construction Inspired by Block Cipher Design Principles", eSTREAM, ECRYPT Stream Cipher Project. http://www.ecrypt.eu.org/stream/papersdir/2006/021.pdf
5. C**.** Piguet et al. "Logic Design for Low-Voltage / Low- Power CMOS Circuits", Proc. of the 1995 International Symposium on Low Power Design, Dana Point, CA, April.
6. T. Schneider et al., "Low-Voltage Low-Power Parallelized Logic Modules", Proc. PATMOS95, Paper S4.2, Oldenburg. October 4-6, 1995, Germany.
7. C. Piquet, "Low-Power CMOS Circuits technology, Logic Design and CAD Tools", CRC Press, 2006.
8. Martin Feldhofer, "Comparison of Low-Power Implementations of Trivium and Grain", eSTREAM, ECRYPT Stream Cipher Project. http://www.ecrypt.eu.org/stream/papersdir/2007/027.pdf
9. A Practical Introduction to Hardware/Software Codesign. Patrick R. Schaumont. Springer Science+Business Media, LLC 2010.
10. New Stream Cipher Designs. The eSTREAM Finalists. Matthew Robshaw Olivier Billet (Eds.). Springer 2008.