School of Electrical, Electronic & Computer Engineering

UNIVERSITY OF
NEWCASTLE UPON TYNE

Dual-Rail with Alternating-Spacer Security Latch Design

D. Shang, A. Yakovlev, A. Koelmans, D. Sokolov and A. Bystrov

Technical Report Series

NCL-EECE-MSD-TR-2005-107

2005

Contact:
d.shang@ncl.ac.uk
alex.yakovlev@ncl.ac.uk
albert.koelmans@ncl.ac.uk
a.bystrov@ncl.ac.uk

Supported by EPSRC grant GR/S81421

NCL-EECE-MSD-TR-2005-107
Copyright © 2005 University of Newcastle upon Tyne

School of Electrical, Electronic & Computer Engineering,
Merz Court,
University of Newcastle upon Tyne,
Newcastle upon Tyne, NE1 7RU, UK

http://async.org.uk/

DUAL-RAIL WITH ALTERNATING-SPACER SECURITY LATCH DESIGN

D. Shang, A. Yakovlev, A. Koelmans, D. Sokolov, and A. Bystrov

School of EECE, Merz Court, University of Newcastle upon Tyne
{}@ncl.ac.uk

Abstract

We present a new design for a dual-rail & dual-spacer latch which exhibits totally symmetrical switching behaviour, which guarantees data independent power consumption and is therefore suitable for use in secure systems. We compare the simulation results with the latest security latches.

1 Introduction

In recent years there has been an increased awareness of information technology security-related issues. More and more people are shopping and banking at home. The use of home and recreational computers has increased dramatically, and the majority of these have Internet access to resources such as email, newsgroups and on-line shopping and banking. However, this has also increased the number of people with an ability to compromise data [1]. This has led to a very high percentage of traffic that requires safeguarding. One of the most fundamental and widespread tools used in providing Internet security is encryption.

Encryption based *security algorithms* are now widely used to protect data during transfer. Unfortunately, those algorithms can be compromised using today's powerful computers. New hardware implementation methods have been proposed to protect sensitive data from attack. Some crypto-systems have been reported in [2]. However, this kind of hardware crypto-system can be attacked using new attacking methods, as a result of *side-channel information leakage* [3]. For example, *differential power-analysis* (DPA) attacks [4] exploit the fact that the power consumption of a device executing a cryptographic algorithm is correlated to the intermediate results of the algorithm. This correlation allows an attacker to extract the secret key used by the system [5].

Hence, the goal of countermeasures against DPA attacks is to completely remove or at least to reduce this correlation. There are two possible approaches to solving this problem[5].

The first approach is to try to make the power consumption of a device independent of the data that is being processed. The countermeasures based on this approach are called *hardware countermeasures*. Typical examples are detached power supplies, logic styles with a data-independent power consumption, noise generators, and the insertion of random delays. Each of these countermeasures reduces the correlation between the data flow and the power consumption. Typically, several hardware countermeasures are combined. This can reduce the correlation down to a level that makes DPA attacks impractical.

The second approach, called *masking*, is to randomize the intermediate results occurring in a cryptographic algorithm. The power consumption of a device processing randomized intermediate results is uncorrelated to the actual intermediate results. Masking can be applied either at the algorithm level or at the gate level.

The first approach is currently the most popular. Special coding methods are used for balancing the power consumption. For example, the use of *dual-rail encoding* together with self-timed design techniques has been proposed in [12, 13]. Whilst this method is undoubtedly promising, it has been found that the dual-rail encoding method cannot fully guarantee the power balancing property[6].

In this paper, we focus on the removal of existing drawbacks of the dual-rail encoding method. The rest of the paper is organised as follows: section 2 provides the design of a new dual-rail secure latch; section 3 compares the latch with existing designs; and section 4 presents the conclusions and future work.

2 Dual-rail encoding problems

The dual-rail code[7] uses two rails with two valid signal combinations {01, 10}, which encode the value 0 and 1 respectively. This code is widely used to represent data in self-timed circuits. Normally, self-timed dual-rail circuits use a *return-to-zero* protocol, which allows only transitions from *all-zeros* {00}, a non code-word, to a code-word, and vice

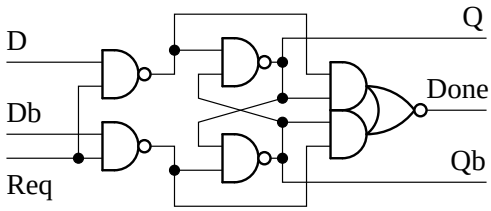

Figure 1: Schematic of the standard dual-rail latch.

versa. The all-zeros state, called a *spacer,* is used to indicate the absence of data, and therefore separates one code-word from another.

Dual-rail circuits exhibit a symmetry in their switching behaviour that reveals little information through their power consumption and electromagnetic emissions. This means that they exhibit inherent robustness against side-channel leakage attacks, by making DPA more difficult.

The return-to-zero protocol certainly helps to balance the switching activity at the level of dual-rail nodes. It depends on the assumption that the power consumed by one rail in a pair is the same as in the other rail, which causes the overall power consumption to be independent of the data bits. There are, however, two problems with this approach:

Problem 1: The standard dual-rail latch, shown in Fig. 1, consumes different amounts of power when storing different data values, even though it is symmetrical. For example, if the initial value of the latch is 0 (i.e. 01 in dual-rail), and the value 1 is written, the latch consumes 2.22407e-13 Joule; when the value 0 is written, it consumes 1.41238e-13 Joule. The difference is 0.81179e-13 Joule (note: throughout the paper, the power consumption is measured by calculating the average of the integrated current between the rising edges of the *req* signal). This difference reveals whether a new value is the same as its predecessor or not. The main reason is for this difference is that, although a return-to-zero spacer protocol is used, the latch cannot be forced into the spacer state. When new data is the same as old data, some gates do not switch, resulting in a lower power consumption than when writing a different data value.

Problem 2: It is difficult to build a dual-rail gate which consumes the same power regardless of data processed because 1) the physical realization at the transistor level is usually not symmetrical; 2) uneven output load and parasitic parameters[8]. A simple example shows this clearly. Suppose we have an AND gate with *a* and *b* as its two inputs. Suppose *b* stays at 0, and *a* changes from 0 to 1, stays at 1 for 5ns, and then changes to 0. Power consumption in this case is 5.257e-15 Joule. When the roles of *a* and *b* are reversed, the gate consumes 4.257e-15 Joule. The difference is caused by the fact that the AND2 gate is not symmetrical at the transistor level. In addition, it is obvious that the dual-rail gates consume different power due to uneven load at the two rails, even if the gates themselves are symmetrical at the transistor level.

3 Dual-rail alternating spacer security latch design

To solve the first problem, we force the latch to return to the spacer state before writing new data, in order to remove all traces of the previous value. The reasonable assumption is made that when new data is written into the latch, the old data has already been used, so it can be safely overwritten.

3.1 Return to zero spacer latch

One possible implementation, shown in Fig. 2, was published in [9]. Before new data is written, the latch is forced into the all-zeros spacer state; only then can the new data be written. The resulting power consumption is as follows. When writing a 0, the latch consumes power 7.01310e-13 Joule; when writing a 1, 6.82956e-13 Joule. The difference is 0.18354e-13. Compared to the standard dual-rail latch (see section 2), the difference is very small.

To solve problem 2, all the gates can be designed such that 1) the physical realisation at the transistor level is symmetrical; 2) both output rails have the same loads and the same parasitic parameters. To make the gates symmetrical at the transistor level, the best solution is to design them specially for this purpose. It also means that a special cell library must be generated. We do not address this problem in this paper.

3.2 Alternating spacer latch

In order to make the switching behaviour symmetrical, the return-to-zero protocol should be extended. The extended protocol uses *two* spacers, namely {00} and {11}, which are used in strictly alternating fashion. This means that *both* rails follow the switching pattern 0-1-0 during a cycle, regardless of the data stream. We now proceed to develop a design

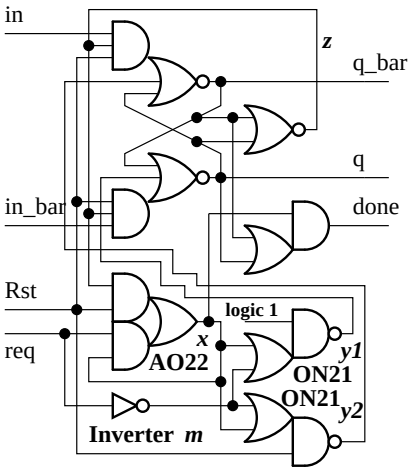

Figure 2: Schematic of the return-to-zeros spacer latch.

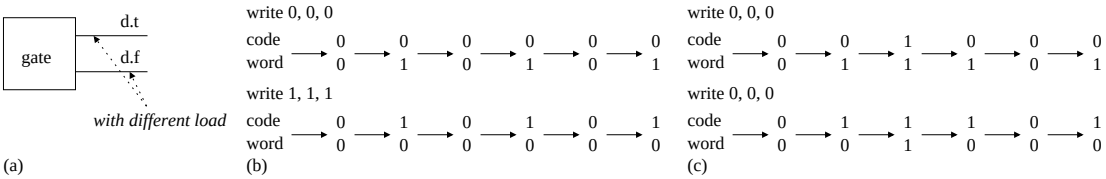

Figure 3: Dual rail switching behaviour

that conforms to this altenating spacer protocol. We base our design on the master-slave latch, which is very popular and is widely used in the kind of architecture shown in Fig. 4, in which the latched data is fed back to the combinational logic.

The alternating spacer protocol was originally used to balance the power consumption of asymmetrical dual-rail gates such as the ones used in the NCL-X design style[8]. We find that it is helpful even for components which are symmetrical at the gate level, but where there is an uneven load or parasitic parameters (as explained above). Fig. 3 shows an example. This is a simple dual-rail gate, which is symmetrical at the gate level. The two rails of the output are assumed to have different loads due to fabrication variations. If the single spacer protocol is used, when the output of the gate is 0, 0, and 0 again, only the *d.f* rail is charged. When the output of the gate is 1, 1, and 1 again, only the *d.t* rail is charged. Because of the different loads in each rail, the power consumption is different. However, if the alternating spacer protocol is used, both *d.t* and *d.f* rails are charged regardless of the output. This means that the same amount of power is consumed (we do not consider the current leakage).

We designed a new security latch that uses the alternating spacer protocol was designed. As our work is focused on staying as close to the standard industry design flow as possible, we selected the architecture shown in Fig. 4, which is based on the master-slave latch.

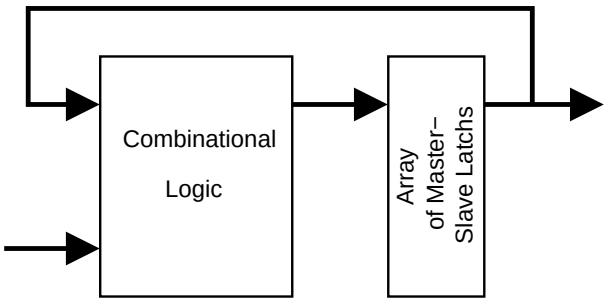

Figure 4: Popular architecture in digital systems.

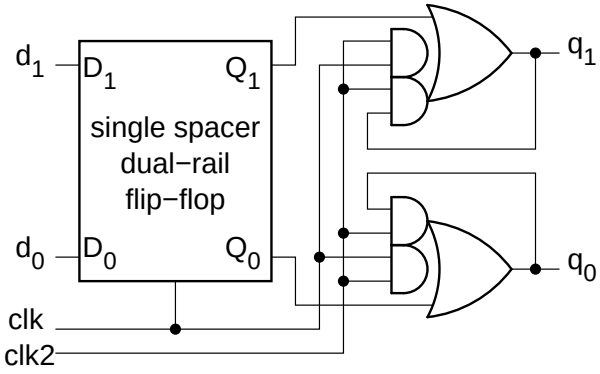

Figure 5: The diagram of the dual-rail alternating spacer latch.

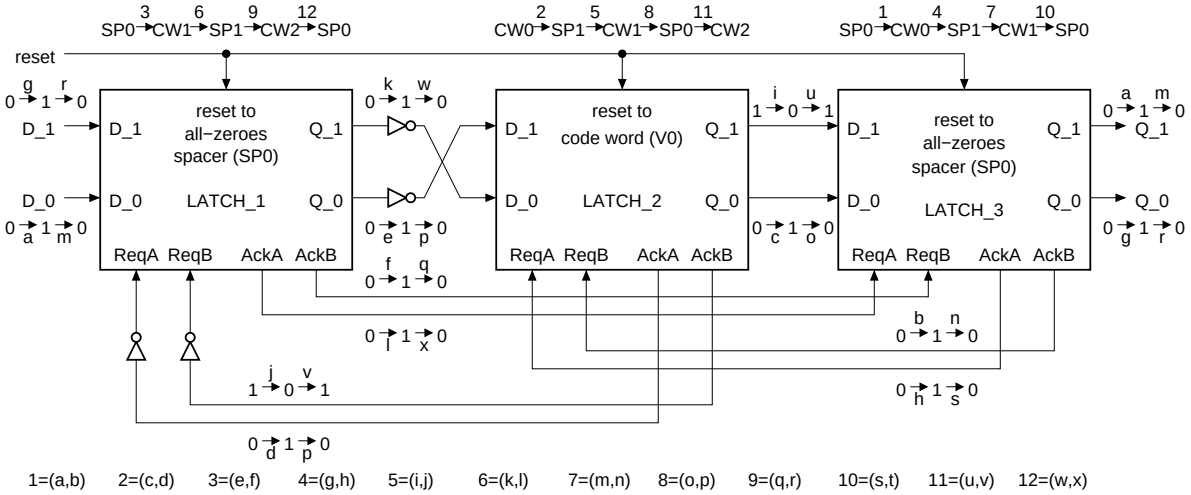

Figure 6: SI dual-rail dual-spacer latch.

One dual-rail alternating spacer master-slave latch (DDMSL) architecture is reported in [10]. The block diagram of the latch is shown in Fig. 5. It consists of two parts. On the left is a dual-rail single-spacer latch, shown in Fig. 2. On the right is a converter, which converts the single-spacer protocol to a dual-spacer protocol. When writing a 0, this latch consumes 1.02004e-12 Joule; when writing a 1, it consumes 1.01727e-12 Joule, a difference of 0.0277e-13 Joule. Since it makes use of a single-spacer latch, it suffers from the above discussed problem. To enjoy the benefits of the alternating spacer protocol, a new latch has to be designed.

The Petri net specification in Fig. 9 shows the operation of the dual-rail single-spacer latch[8]. Two clock signals, *clk* and *clk2*, are used to realise the dual-spacer protocol. The relationship of these two clock signals is $f(clk)$=$2 * f(clk2)$. The converter works as follows: when *clk*2 is low and *clk* is high, the latch stays at the {00} spacer; when *clk*2 is high and *clk* is high, the latch stays at the {11} spacer. Only when *clk* is low, a new data value is stored in the latch. In order to obtain the power-balancing benefits provided by the dual-spacer mechanism, we designed an SI dual-rail alternating spacer master-slave latch that does exhibit the required characteristics. The block diagram of the latch is shown in Fig. 6, and its STG specification in Fig. 7. It consists of three separate latches, which ensures that a data value is never overwritten and that two code-words are never stored simultaneously. Its implementation is shown in Fig. 8.

Initially, the internal latches are reset as follows: LATCH_1 and LATCH_3 are initialised to an all-zeroes spacer, and LATCH_2 is initialised to a code-word depending on what type of single-rail flip-flop is being replaced. After the reset, LATCH_3 switches to a code-word and signals LATCH_2, which then consumes the spacer from LATCH_1. LATCH_3 then consumes the code-word at its input. The inverters between LATCH_1 and LATCH_2 alternate the spacer coming from LATCH_1 and are crossed to maintain code-word polarity.

The annotation in Fig. 6 attempts to explain the operation in terms of switching. The high-level protocol (spacer to code-word) is depicted above the latches. The abbreviations SP0 and SP1 represent the all-zeros and all-ones spacers respectively; CW0 and CW1 represent the code-words. The switching sequence between the spacers and code-words is marked by the numbers above the arrows. The switching of individual wires is shown next to them, and the switching sequence is indicated by letters according to alphabetical order. The relationship between the protocols is shown at the

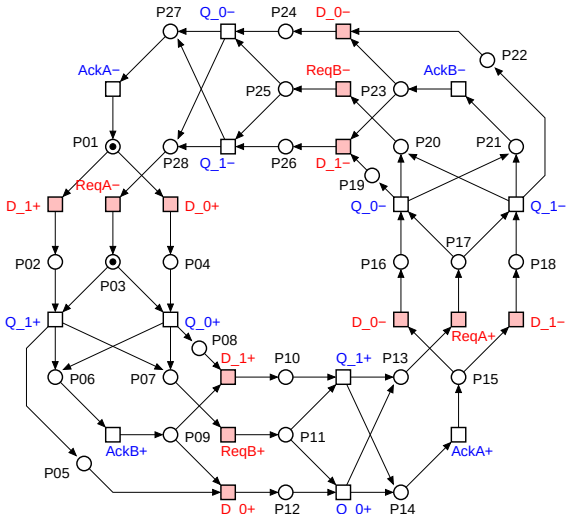

Figure 7: The STG for dual-rail dual-spacer latch.

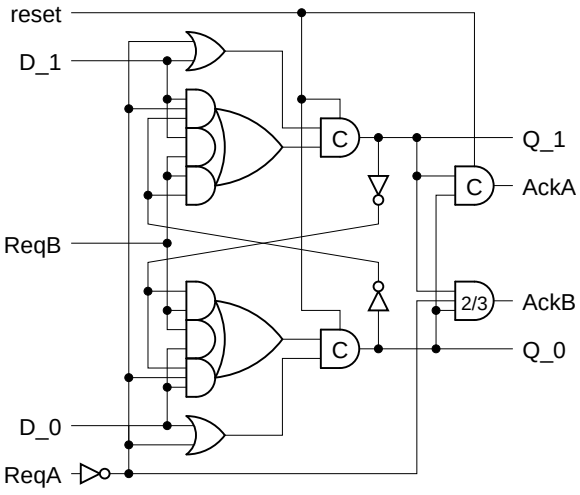

Figure 8: Schematic of the internal latch.

bottom of the figure.

3.3 New dual spacer latch design

We now design a 'proper' dual-rail dual-spacer latch. In the same fashion as DDMSL, we focus on implementing a master-slave latch, because this kind of latch can be used in the popular architecture shown in Fig. 4 in digital systems, in which the data, after computation, is latched into an array of master-slave latches and fed back into the combinational logic for the next stage of the computation.

The same design idea as developed in the single return-to-zero spacer latch, published in [9], is employed in the new latch design. This means that before writing data, the latch is forced into a spacer state, and then new data is written. As for the master-slave latch, this mechanism is arranged as follows: before writing new data, the master is forced into a spacer state, and then the data is latched into the master. After that, the slave is forced into the opposite spacer state. Meanwhile, the *req* signal is withdrawn. Then the data is propagated to the slave. The STG is shown in Fig. 9.

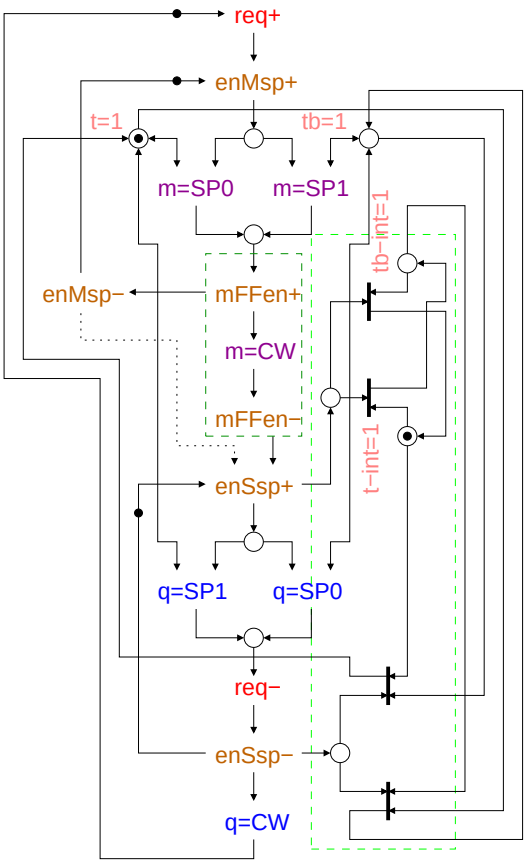

Figure 9: The Petri net specification of the dual-rail dual spacer latch.

The STG specification can also be synthesised (using the Petrify tool) to generate a Speed-Independent (SI) circuit. However, SI circuits have a high area overhead because of the required completion detection logic. So, instead of a SI solution, a compact solution with simple timing assumptions is sought.

To satisfy the requirements used in the architecture shown in Fig. 4 and implement the alternating spacer protocol, the latch consists of a master-slave latch, a toggle and a controller. The master-slave latch is the main part, and is used to propagate and store data. The toggle is used by the controller to determine the spacer for the master and slave latches. On the rising edge of the *req* signal (*req*+), the controller forces the master into a {00} or {11} spacer depending on the value of the toggle. If the value is 1, the spacer for the master is all-zeros and the spacer for the slave is all-ones, and vice-versa. The controller is used to generate the spacers for the master and slave, and to propagate and store the data inside the latch. After the master is forced into the spacer, the code-word is latched into the master and the slave is forced into a {11} or {00} spacer depending on the value of the toggle. The spacer is propagated to the environment, and then the *req* signal is withdrawn (*req*-). After that, the code-word is propagated to the slave. Since the spacers are dependent upon the value of the toggle, the toggle should be updated in each cycle.

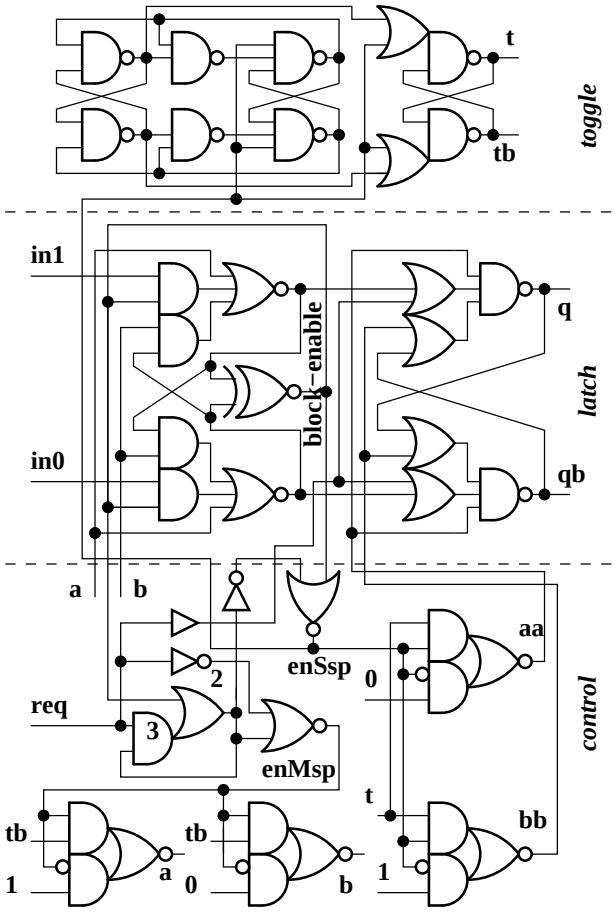

Figure 10: A dual-rail and dual spacer security latch.

One possible implementation of the dual-rail alternating spacer security latch (RDDMSL) is shown in Fig. 10, which was designed manually. In Fig. 10, the top is the toggle, the bottom is the controller and the middle is the main latch. In this figure, 0, 1, 2 and 3 stand for logic 0, logic 1, inverter and AO21 respectively. When a 1 is written into the latch, it consumes 2.59703e-12 Joule; when a 0 is written, 2.59732e-12 Joule. The difference is 0.0029e-13 Joule. Compared to the normal latch and the single return-to-spacer latch, the latch has a high security value.

The design works as follows: when the latch stores data the *block-enable* signal is clear (*block-enable*=0), which blocks new inputs; when the *req* signal is asserted (*req*+), it is propagated through gate 2 (inverter), the *enMsp* signal is set (*enMsp*=1), and the master is forced into a spacer state. If the toggle is 1, {a,b} equals {1,1}, which causes the all-zeros spacer to be generated; otherwise, the all-ones spacer is generated. After the master is forced into a spacer state, the block-enable is set (block-enable=1). At this point, new data cannot be latched until *emMsp* is withdrawn, and {a,b} equals {0,1}. After new data is latched into the master, the block-enable is set again, and then the *enSsp* signal is generated (enSsp=1). After that, the slave is forced into a spacer state, using the same mechanism as the master. The choice of all-zeros or all-ones spacer is decided by the value of the toggle. If the toggle is 1, {aa,bb} is set to {0,0}, and the all-ones spacer is generated; otherwise, the all-zeros spacer is generated. After the slave is forced into a spacer state, the spacer is fed back to the inputs, the input data is returned to a spacer, and *req* is withdrawn. After *req*-, the data is propagated and stored into the slave. This happens only after {aa,bb} is changed to {1,0}.

The following timing assumption is used:

Timing assumption 1: in order to keep *enMsp* low, gate 2 should be faster than gate 3 (AO21).

(1) $\Delta inverter \prec \Delta AO21$

This is clearly a reasonable assumption because the inverter is smaller and faster than AO21.

The toggle (shown at the top of the Fig. 10) is basically a T latch, and it is used only for generating spacers. It can be changed after data is latched in the slave. In order to improve its performance, it is also designed as a master-slave latch. After *enSsp*+, the master is updated and after *enSsp*-, the master value is propagated to the slave.

Two simple assumptions are used:

Timing assumption 2: before *enSsp*-, the master of the toggle should be updated.

(2) $\Delta(3NA2 + 2NA3) \prec \Delta(AN22 + ON221 + AO21 + NO2)$

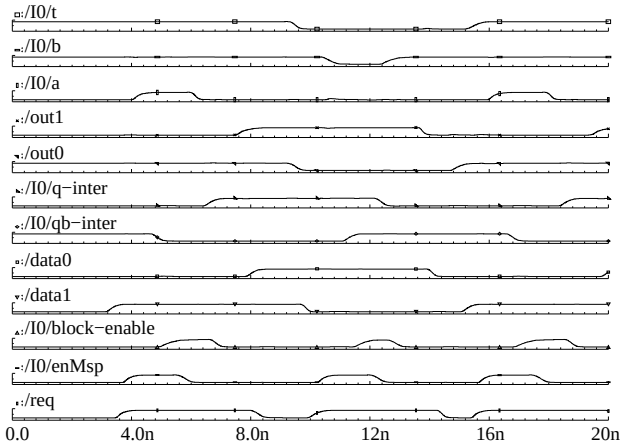

Figure 11: Simulation results (1).

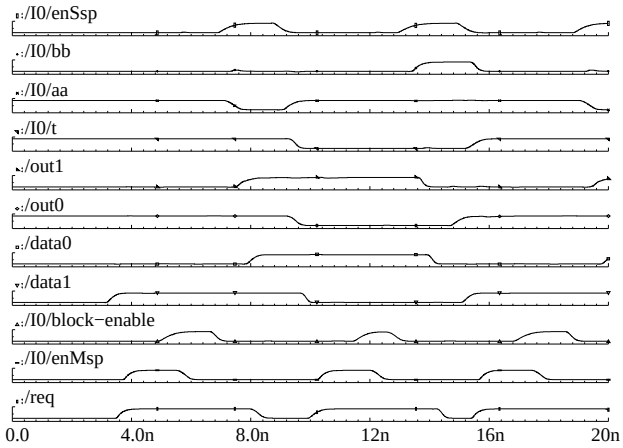

Figure 12: Simulation results (2).

Timing assumption 3: after *enSsp-*, the slave of the toggle is faster than the slave of the main latch.
(3) $\Delta(2ON21) \prec \Delta(2ON221)$

Both of these assumptions are reasonable, because, in both cases, the delay of several simple gates should be smaller than the delay of several complex gates.

4 Discussion

The latch is implemented in the AMS 0.35u CMOS technology using the Cadence toolkit. The analogue simulation results show that the latch works as expected. The waveforms are shown in Fig. 11 and Fig. 12.

The other latches mentioned in this paper are also implemented in the same technology under the Cadence toolkit. In the previous sections, some comparisons have been made in terms of power consumption and security. These are summarised in Table 1. The higher the number, the better the latch. In terms of power consumption, the standard dual-rail latch is the best one. However, in terms of the security property, the dual-rail alternating spacer latch is the best.

Table 1: Comparison results (1)

	4	3	2	1
Power	standard dual-rail latch	dual-rail single spacer latch	DDMSL	RDDMSL
Security	RDDMSL	DDMSL	dual-rail single spacer latch	standard dual-rail latch

We also compare the three security security latches in terms of their usages, speed, and size. The comparison results are shown in in Table 2. The clocked security latch (DDMSL) is the fastest one with the smallest size. The SI one is the slowest one with the biggest size. The new security latch (RDDMSL) is fast with reasonable size. A special feature is that the RDDMSL can share the controller to reduce the size of the circuit in which it is used. The RDDSML latch can be used in both synchronous and asynchronous circuits, compared to the DDMSL which can be used in synchronous circuits only, and is about 3 time faster than the DDMSL, although there is a 67% area penalty. In addition, its power balancing property is excellent.

Table 2: Comparison results (2)

	SYNC Env.	ASYNC Env.	Shared Controller	req(clk) to req(clk)	Size
DDMSL	yes	no	no	2n	1
RDDMSL	yes	yes	yes	7.5n	1.67
SI	yes	yes	no	18n	3.2

5 Conclusions and future work

We have designed a new dual-rail dual-spacer master-slave latch. The analogue simulation results show that the latch works as expected. We compared it with the latest DDMSL type latch, and found that it is faster, has better power balancing, and has shared controller advantages, although there is about a two-third area penalty.

In the future, we will design a security system using this latch, to check that the properties we reported work in practice. Furthermore, our group now focuses on security design, which includes the development of additional security components.

6 Acknowledgement

This work is supported by EPSRC project SCREEN at the University of Newcastle upon Tyne. Thanks go to Dr. Frank Burns and Dr. Fei Xia for useful discussions.

References

[1] McLoone, M., and McCanny, John V., 2003, “System-On-Chip Architectures and Implementations for Private-Key Data Encryption”, Kluwer Academic/Plenum Publishers, New York.

[2] Mangard, S., Aigner M., and Dominikus, S., “A Highly Regular and Scalable AES Hardware Architecture”, IEEE Transaction on Computers, Vol. 52, No. 4, April 2003.

[3] Hess, E., Janssen, N., Meyer, B., and Schutz, T., “Information Leakage Attacks Against Smart Card Implementations of Cryptographic Algorithms and Countermeasures: A Survey”, EUROSMART Security Conference, 2000.

[4] Kocher, P., Jaffe, J., and Jun, B., “Differential Power Analysis”, CRYPTO 1999, Santa Barbara, CA, USA, August 15-19, 1999, LNCS 1666, Springer.

[5] Mangard, S., Popp, T., and Gammel, Berndt M., “Side-Channel Leakage of Masked CMOS Gates”, CT-RSA 2005, The Cryptographers’ Track, San Francisco, CA, USA, February 14-18, 2005, LNCS 3376, Springer.

[6] Plana, L.A., Riocreux, P.A., Bainbridge, W.J., Bardsley, A., Garside, J.D., and temple, S., “SPA - A Synthesisable Amulet Core for Smartcard Applications”, ASYNC 2002, Manchester, UK, April 2002.

[7] Sparso, J., and Furber, S., 2001, “Principles of Asynchronous Circuit Design: A System Perspective”, Kluwer Academic Publishers, Boston.

[8] Sokolov, D., Murphy, J., Bystrov, A., and Yakovlev, A., “Design and Analysis of Dual-Rail Circuits for Security Applications”, IEEE Transaction on Computers, Vol. 54, No. 4, April 2005.

[9] Shang, D., Burns, F., et. al, 2004, “A Low and Balanced Power Implementation of the AES Security Mechanism Using Self-Timed Circuits”, PATMOS 2004, LNCS 3254, Springer.

[10] Sokolov, D., Murphy, J., Bystrov, A., and Yakovlev, A., 2004, “Improving the security of dual-rail circuits”, CHES 2004, Boston Marriott Cambridge, USA, August 11-13, 2004.

[11] Stevens, Ken, Ginosar, Ran, and Rotem, Shai, “Relative Timing”, ASYNC 1999, Barcelona, Spain, April 1999.

[12] Moors, S., Anderson, R., Cunningham, P., Mullins, R., and Taylor, G., “Improving Smart Crad Security Using Self-Timed Circuits”, ASYNC 2002, Manchester, UK, April 2002.

[13] Yu, Z., Furber, S., and Plana, L., “An Investigation into the Security of Self-Timed Circuits”, ASYNC 2003, Vancouver, BC, Canada, May 2003.