Balanced Self-Checking Asynchronous Logic for Smart Card Applications

Simon Moore, Ross Anderson, Robert Mullins, George Taylor
Computer Laboratory, University of Cambridge
simon.moore@cl.cam.ac.uk
Jacques J.A. Fournier
Gemplus Card International Card Security Group
jacques.fournier@gemplus.com

Abstract

Delay-insensitive or unordered codes may be used to construct both robust asynchronous circuits and self-checking systems. The redundant nature of the coding scheme also provides the possibility of a balanced implementation, where the power dissipated is independent of the input data. We demonstrate how these characteristics may be exploited to construct smart card functions that are resistant to both side-channel and fault injection attacks. We also describe how the removal of the clock secures a potential point of attack and enables additional fine-grain timing countermeasures to be introduced. Preliminary results are presented for a smart card test chip containing multiple implementations of a 16-bit micro-controller, a smart card UART and a Montgomery modular exponentiator.

Keywords: asynchronous logic, smart card, security, differential power analysis, electromagnetic emissions analysis

1. Introduction

Smart cards are increasingly prevalent for authentication and payment mechanisms (credit cards, pay-TV conditional access, public transport payment, medical records, personal identity, mobile phone SIMs, *etc.*). The security of such systems relies on the ability of the smart card to perform cryptographic

operations while keeping the key stored on the card secret. Of particular concern are attacks that provide the ability to remove secret key information relatively quickly, while leaving no evidence of tampering. Non-invasive or side-channel attacks extract keys by analysing information leaked in the form of power usage [1], electromagnetic emissions [2, 3] or timing [4]. In addition to monitoring the card, circuit activity may be externally influenced by modulating power or clock pins [5, 6]. A more targeted attempt to induce switching activity may also be performed using an EM pulse [7]. Such techniques are often used in an attempt to force the card to malfunction in a predictable and instructive manner.

In addition to non-invasive attacks it is often necessary to assume that design details may be acquired through reverse engineering. In this case it is only the granularity at which an attacker may detect and influence circuit activity that limits their ability to extract keys. A particular threat is the use of optical fault injection. An attacker can use a laser flash to cause a target transistor group to conduct at the time of their choice [8].

In this paper we describe techniques for improving smart card security by severely limiting the degree to which circuit activity may be monitored or influenced externally.

2. Power Analysis Countermeasures

Power dissipation in static CMOS circuits is dominated by switching activity. As a result, the power dissipated by a circuit is highly dependent on the switching activity produced by a change of input data. In the simple case of a bus, activity is observed as the Hamming weight of the state changes.

Power analysis attacks are prevented by removing or hiding information leaked by such switching activity. Information may be removed or at least reduced by attempting to dissipate constant power, regardless of input data. At the transistor level, a move from static logic to a form of constant current logic may produce large reductions in data-dependent power, although performance may suffer considerably [9].

Information may be hidden by randomising the order of execution [10] or randomising the particular hardware resource used in an operation [11]. Additional random operations may also be performed in order to further obfuscate power emissions, although care must be taken to avoid the possibility that such rescheduling or dummy operations could be detected and reversed during signal post-processing.

At the logic level, transformations may be applied to the computation to remove any correlation between input data and power dissipation [12]. These techniques exploit a random number source to encode each input bit in order to hide data dependent emissions. Each set of encoded bits are recombined after computation to produce the correct binary output. While such techniques potentially provide a high degree of security, they often require significant area overheads and do not naturally possess a

self-checking property.

The techniques presented in this paper rely on balancing the power consumption at the gate level. Each bit or group of bits is encoded using 1-of-`n` or one-hot encoding. While in general any `m`-of-`n` encoding could be used, 1-of-`n` is often most appropriate for an asynchronous implementation. The encoding guarantees that the same number of signal transitions are made regardless of the input data, hence balancing power consumption. The 1-of-2 or dual-rail case is illustrated in Figure 1. Another common encoding is 1-of-4 which represents groups of two bits. In addition to balancing power consumption the encoding allows the completion of an operation to be detected, permitting a robust clock-free implementation. The design of such logic is often simplified if a return-to-zero (RTZ) is forced after each computation in order to provide a *spacer* between different data items. The combination of this RTZ phase and the careful design of the logic itself means that correct operation is guaranteed regardless of gate delays [13] (such circuits are known as speed-independent).

A common building block for asynchronous circuits is the Muller C-element [13, 14]. The gate operates by setting its output when both inputs go high, but only resetting when both inputs go low. The element hence provides a conjunction of both rising and falling transitions. This is crucial when using delay-insensitive encodings to prevent RTZ being detected too early (the case if AND gates were employed). The C-element symbol is shown in Figure 2, we use an AND-gate symbol with an additional C character at its centre.

Unfortunately, 1-of-`n` encoding is not sufficient to guarantee a completely data independent power signature. One potential problem is that gate loads may differ due to differences in routing. The design of each dual-rail gate must also ensure equal input pin loads and balanced power usage. In practice this is easy to achieve at the gate level and may be reinforced with careful grouping of cells during placement. The implementation of a dual-rail OR gate is discussed in detail in Section 5.

A final concern is that while each individual functional unit may no longer produce a data-dependent power signature, activity at a higher level may still be data-dependent. An obvious solution is to maintain a high level of system activity at all time, even if superfluous operations must be performed in order to keep logic busy. In a synchronous system this is trivial as the clock provides a constant indication of when a new operation should begin (regardless of the data dependencies in the circuit). In contrast, an asynchronous system is data-driven where each functional unit has no concept of an idle cycle. In practice, this problem can often be tackled at the architectural level. A further observation is that forcing a high level of circuit activity will lead to a pseudo-synchronous mode of behaviour even in a

delay-insensitive asynchronous implementation.

3. Fault Injection Countermeasures

The possibility of performing clock glitch attacks is removed completely in the case of asynchronous circuits. Where a clock is required, for example in the clocked serial smart card interface, it is easy to arrange that data corruption should not result in sensitive data leakage.

Power glitch or localised fault induction techniques are tackled by exploiting the self-checking property of `m`-of-`n` encoded circuits. Such coding schemes are known as *immutable* codes, and hold the property that any valid codeword of length n contains exactly m 1's. Any errors that lead to an increase or decrease in the Hamming weight of the code are therefore easily detected. Any single bit or asymmetric errors fall into this category. Thus, for a dual-rail encoded circuit the invalid code (1,1) now represents an *error* or *alarm* state (see Figure 1) [15, 16, 17]. A fault leading to a (0,0) code will be prevent completion and may be detected by a timeout. To guarantee that we observe any illegal codewords on the output of the circuit we must only guarantee that each gate generates an illegal output encoding in response to any illegal input encoding. While many other self-checking codes and architectures have been proposed [18], it is not clear that any of them offer the possibility of easily balancing the power dissipation for different data inputs.

A1	**A0**	**meaning**
0	0	clear
0	1	logical 0
1	0	logical 1
1	1	alarm

Figure 1. Dual-rail encoding with alarm signal defined

Ideally all detectable faults would now produce one of the following two behaviours in the case of an asynchronous circuit:

1. A reduction in the number of 1's will prevent completion detection, forcing the circuit to stall indefinitely. Additionally, a time-out may be associated with each circuit to quickly cause an alarm code to be propagated.

2. An increase in the number of 1's will cause an error to be detected at the output. The alarm signal

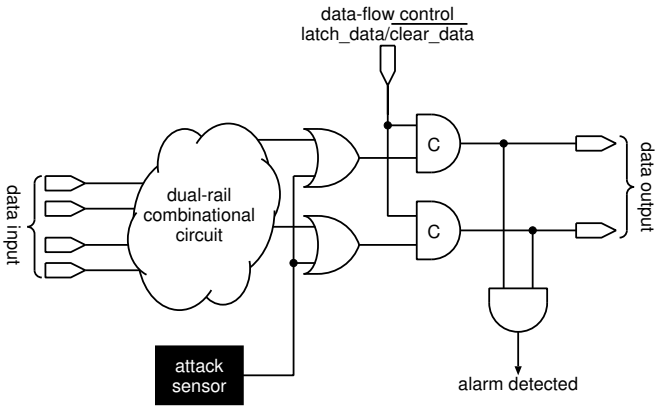

Figure 2. Alarm insertion and detection

can subsequently be distributed to other circuits to quickly force a system-wide deadlock.

Many different tamper sensors exist that are capable of detecting changes in supply voltage or attempts at depackaging (*e.g.* temperature, light level, package capacity/conductivity and metal mesh sensors). All emit a signal when the appropriate condition is violated. The *alarm* state, as indicated in the previous section, can be activated by ORing an *alarm* control signal from the tamper sensor with the dual-rail data path (see an example fragment of data-path in Figure 2). The *alarm* state is detected by an AND–OR tree on the data, and may injected into another part of the circuit to accelerate deadlock. The amount of circuitry is substantially reduced when it is known that setting the alarm for one logical bit will result in the alarm being propagated to other bits — see the worked example in Section 5.

4. Diffusing data dependent timing

Timing attacks on clocked processors are undertaken by counting cycles between known events in order to derive secret information [4]. An asynchronous processor is more susceptible to leaking data in the time domain. For example, an adder could be carefully designed to consume constant energy per operation, but the duration of the operation is dependent upon the propagation of the carry. In this instance, the adder must be designed to always achieve worst case performance. This does not mean that the adder must have bad performance. For example, we have designed a speed-independent carry select adder which operates quickly and with constant timing properties.

Variation in wire and gate loads can also lead to data dependent timing. Wire loads can be minimised by applying suitable constraints during place and route. If wire loads are to be made insignificant then

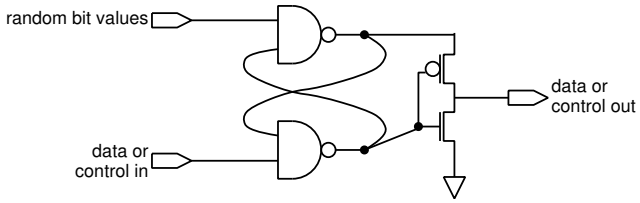

Figure 3. Random delay insertion

the design should be partitioned into a number of small blocks. Wire load balancing then only needs to be considered at interface boundaries.

Fan-out in SI circuits is typically symmetric for dual-rail wires, since as many conditions need to be tested if the value is a *logical-0* as they do when the value is a *logical-1*. For example, see the full adder in Section 5.

Where there is concern over data dependent timing, random delays may be inserted using the circuit in Figure 3 which is based upon a Seitz arbiter [19].

Data dependent software timing is also an issue. It is imperative that application software for a smart card works in collaboration with the hardware. One approach is to ensure that the same work is undertaken regardless of the data. However, this soon becomes difficult and results in very slow code. An alternative approach is to introduce non-determinism at the software level (see the example in Figure 4). It is important to note that this form of non-determinism only adds complexity where the side channel information is hard to analyse in the first place.

```
if(random_condition) {
            f(); g();
} else {
            g(); f();
}
```

where `f()` and `g()` are independent functions

Figure 4. Adding non-determinism to software

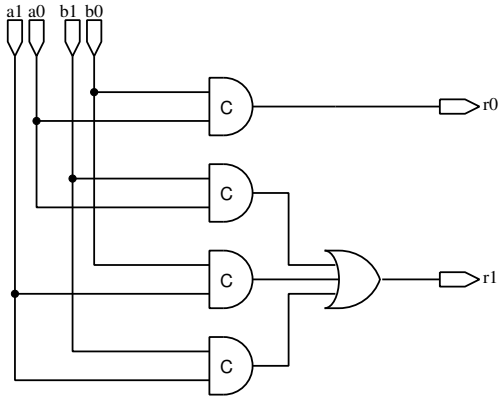

Figure 5. Dual-rail SI OR gate

5. Worked Example

To illustrate the techniques discussed in the previous section, we describe the design of a dual-rail OR gate and a dual-rail full-adder. To improve security, data-independent timing and power consumption is sought. We also wish to be able to guarantee propagation of *alarm* signals.

A simple dual-rail OR gate is shown in Figure 5. If an *alarm* signal is present on (a_1, a_0) or (b_1, b_0) inputs then the output r_1 will always be set (when anything but a null signal is present on the other input). However, the r_0 signal will not be set if only one of the dual-rail inputs is at *alarm*. Also, the OR gate takes a data dependent time to produce an output. These two issues are resolved in Figure 6 by adding additional logic to detect the alarm state, forcing r_0 high and at the same time making the delay data independent. The C-elements in the alarm propagation circuit could be replaced with AND gates if one was confident that the alarm signal coming in was stable. However, under a glitch attack we cannot be confident of this so the storage aspect of the C-elements is desirable. It also serves to balance the loads on each input pin.

The OR gate example illustrates how path delays can be balanced. However, this is not always necessary. For example, let us consider a dual-rail full-adder. The dual-rail inputs are (a_1, a_0), (b_1, b_0), (cin_1, cin_0) and the outputs are (sum_1, sum_0) and $(cout_1, cout_0)$. Since we require several functions of the inputs it is worth expanding the inputs into 1-of-8 code vis:

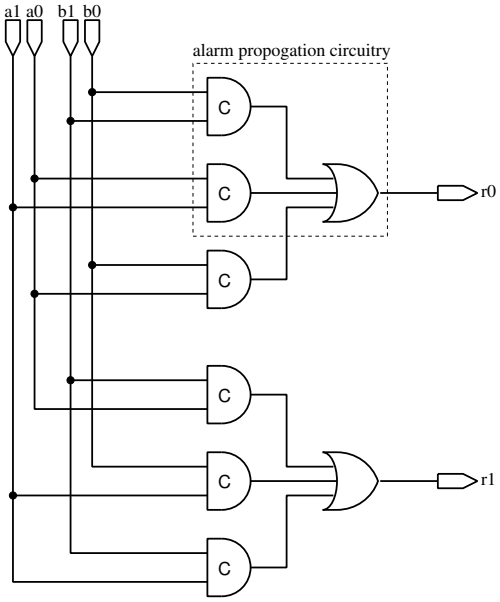

Figure 6. Dual-rail SI OR gate with alarm propagation

$$i000 = C(a_0, b_0, cin_0)$$
$$i001 = C(a_0, b_0, cin_1)$$
$$i010 = C(a_0, b_1, cin_0)$$
$$\vdots$$
$$i110 = C(a_1, b_1, cin_0)$$
$$i111 = C(a_1, b_1, cin_1)$$

where $C()$ represents a C-element function

Now we can define the outputs as an OR-plane of the 1-of-8 code:

$$sum_0 = i000 + i011 + i101 + i110$$
$$sum_1 = i001 + i010 + i100 + i111$$
$$cout_0 = i000 + i001 + i010 + i100$$
$$cout_1 = i011 + i101 + i110 + i111$$

This approach results in data independent timing but the *alarm* signal is not guaranteed to propagate. The *alarm* signal may be identified and incorporated into the sum and carry terms:

$$
\begin{aligned}
alarm &= RS(reset, a_0.a_1 + b_0.b_1 + cin_0.cin_1) \\
sum_0 &= i000 + i011 + i101 + i110 + alarm \\
sum_1 &= i001 + i010 + i100 + i111 + alarm \\
cout_0 &= i000 + i001 + i010 + i100 + alarm \\
cout_1 &= i011 + i101 + i110 + i111 + alarm
\end{aligned}
$$

where $RS()$ is an RS flip-flop (reset dominant)

The RS flip-flop is used to keep *alarm* high causing deadlock. Many dual-rail functions may be implemented in this SI form: first expand the inputs to a 1-of-n code and then have an OR plane to determine the outputs.

6. Case Study: Bus Cryptography

It is convenient to be able to use standard memory components when designing a secure system. Such parts typically have a conventional binary encoded interface. Consequently these devices, and the buses going to them, consume data dependent power. This may be ameliorated in two ways:

1. balance the power consumption
2. encrypt the data

Balancing the power may be achieved in a number of ways. One approach is to write the true and complement of the data at the same time. However, this doubles the amount of storage required. Power consumption during writes may be balanced using dummy circuitry, but this is not possible with reads since you do not know what is going to be read, and hence what data needs balancing, until the read operation has been performed.

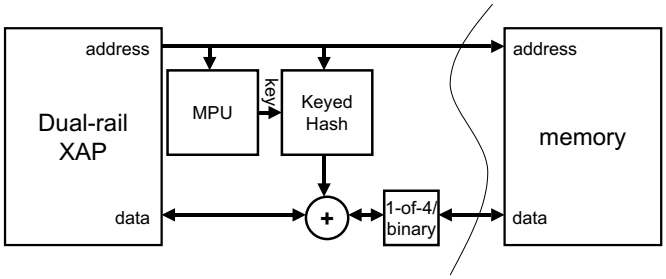

Figure 7. Bus cryptography architecture

An alternative strategy is to encrypt the data held in the memory. We have developed the scheme depicted in Figure 7. A dual-rail 16-bit processor (based on the commercial XAP[1]) has a memory protection unit which supplies a 64-bit key per protection region. The hash of the key and address is used to XOR with the data. On writes, the hash function is on the critical path, but this performance impact can be amortised by pipelining. On reads, the hash is computed in parallel with the memory access and results in no performance degradation.

In our prototype chip, the hash is computed in four rounds (in a production chip with a 32-bit bus, we would have six rounds for cryptographic security). Each round consists of an XOR of the input with part of the key followed by a set of 4-input to 4-output S-boxes taken from the Serpent block cipher [20]. There is then a further layer of XOR. In our system the hash is computed using 1-of-4 encoded data. We compute the XOR by extracting all 16 combinations of the two 1-of-4 inputs using 16 C-elements. The 1-of-4 result is then computed by an OR plane (in Figure 9 the C-element plane is inverting so a NAND plane is then used). Next the S-boxes and transposition have to be implemented. The S-boxes are a complex function of the 4 logical bits of input. The two pairs of 1-of-4 inputs are first converted into a 1-of-16 code. Each S-box and transposition is then just a transposition of the 1-of-16 code, giving a 1-of-16 output. Given that the final function required is a large XOR, having the data in a 1-of-16 form is advantageous. As a result, the S-boxes and transposition functions are reduced to wiring. Thus, the computation is dominated by the XOR functions. In our implementation, four rounds of the keyed hash function takes 3.3 ns which is comparable to the 3.5 ns access time for the low power SRAMs used for data storage (on a 0.18 μm CMOS process). The 1-of-4 XOR functions are perfectly balanced and propagate the *alarm* signal without further modification.

While designing this unit, we observed a vulnerability of some existing bus cryptography systems. This is that multiple writes of the same data to the same address yield the same enciphered data, as the key in such systems remains unchanged. This means that coincidences in data become apparent to an observer. With some algorithms this is highly undesirable. For example, if the chip software is implementing a common block cipher such as the Data Encryption Standard, the observation of repeated identical values of round keys may enable the cipher key in use to be deduced with high probability. The solution we implemented is to design the memory protection unit (with which the bus cryptography unit is integrated) so that it enables the operating system to treat segments of memory as write-once. Thus each round of a block cipher can use a different segment of memory, encrypted with a key that is cleared after use.

[1]The original XAP microcontroller is available from Cyan Technologies: http://www.cyantechnology.com/

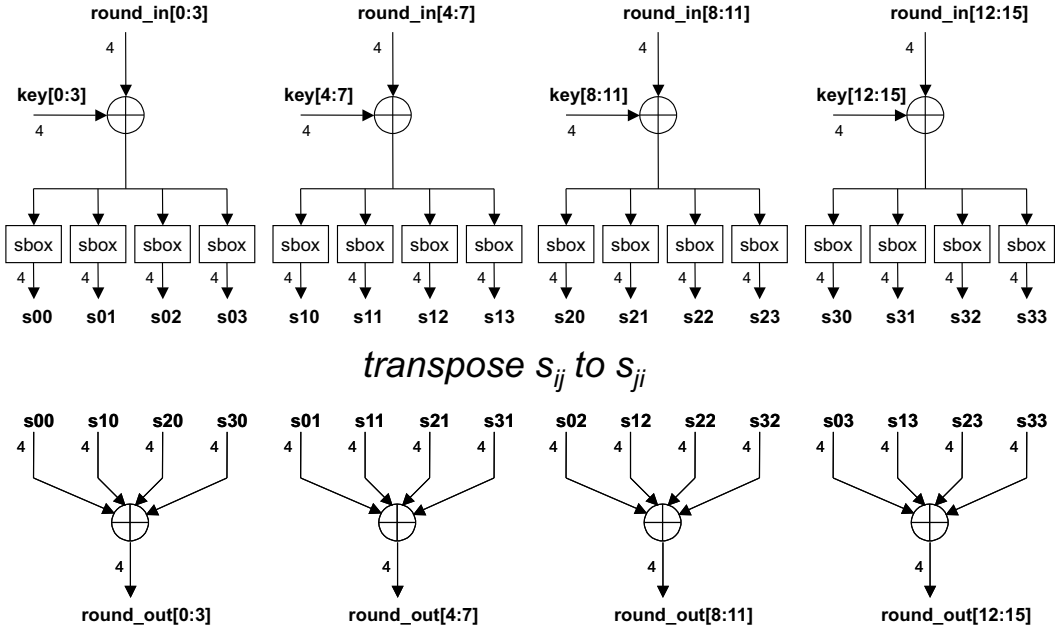

Figure 8. Overview of the keyed hash function

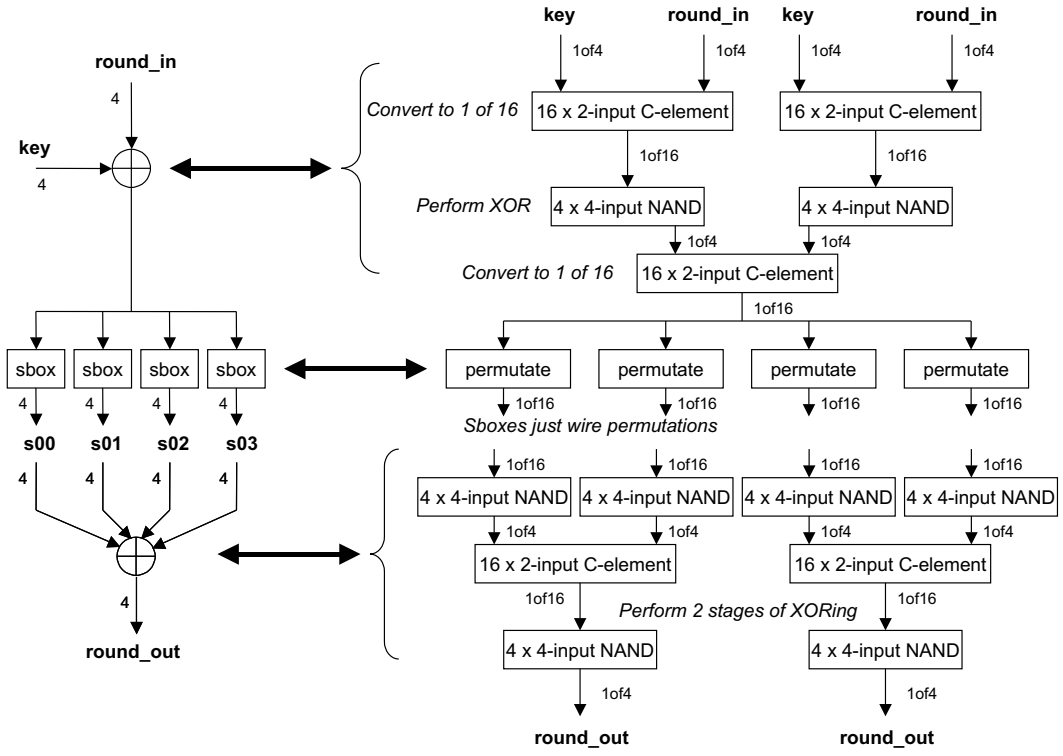

Figure 9. Implementation of the keyed hash function

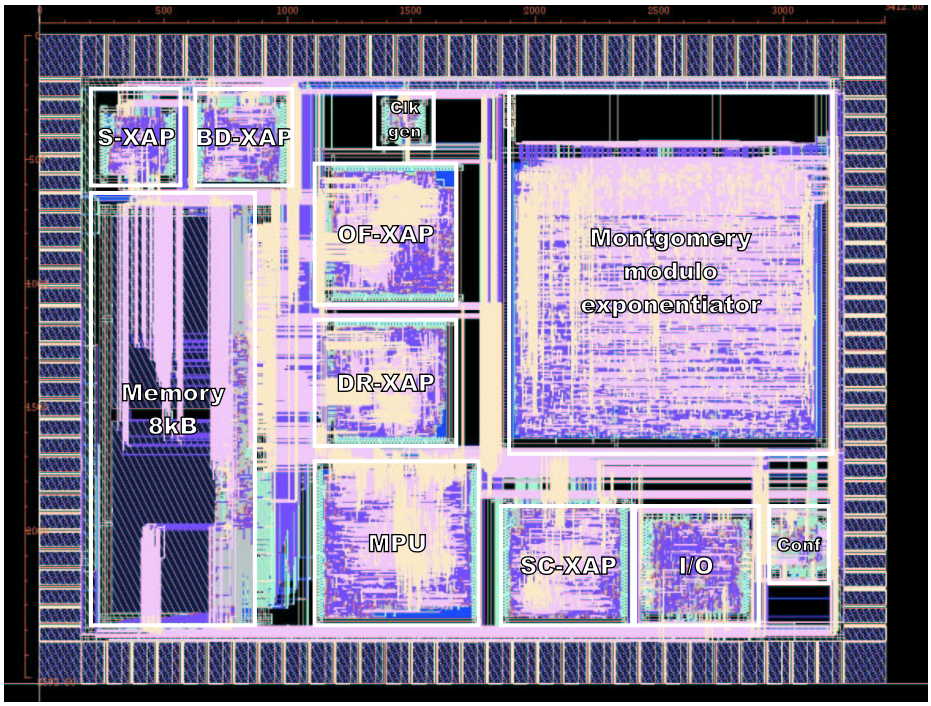

Figure 10. Test chip

7. Test Chip Results

We have recently fabricated a test chip in a 0.18μm five layer metal process. This chip contains five 16-bit XAP processors, a Montgomery modular exponentiator (for fast public key exchange using RSA with keys of up to 2048 bits), an I/O block which includes a smart card UART, and a distributed 1-of-4 interconnect (see Figure 10). One of the 16-bit processors (the SC-XAP) has been specifically designed with security in mind and has a memory protection unit with bus cryptography attached. The SC-XAP uses balanced dual-rail logic throughout. Standard on-chip single-rail SRAM blocks were used to store program and data.

In order to undertake a security analysis of the proposed asynchronous circuit techniques, we undertook *power analysis* [1] and *electromagnetic analysis* [2, 3] to see if data could be extracted. We also used *optical fault injection* [8] to determine if state could be surreptitiously changed.

7.1 Experiment set-up

The XAP based test chip was designed to demonstrate asynchronous circuit techniques rather than being efficient at executing cryptographic functions. Therefore, we concentrate our investigation on

basic operations like memory access, boolean operations and integer arithmetic. Moreover, the focus has been put on the Secure XAP (SC-XAP) and its performances compared to its synchronous alternative (S-XAP). Given that both processors lie on the same chip and have been designed using the same standard cell library, comparisons do not need to take into account any technology or foundry variations. We also designed the test chip to make measurements simpler by providing extra test points (e.g. separate power and ground pins to the core and pads) and additional I/O (e.g. a parallel input/output device to allow trigger signals to be received and generated, and an RS232 interface to communicate with a PC).

The experiment set-up is straightforward. We download a small program into the shared on-chip memory which receives data from the PC over the RS232 interface. For example, if testing the `XOR` instruction, we receive two 16-bit words of data which are then XORed together and written to memory. With this set-up, we are able to investigate the security of instructions and memory accesses.

7.2 Power Analysis

Power analysis of the Secure XAP's ALU revealed that small imbalances in the design of the dual-rail gates and loads due to interconnect allowed some data-dependent power usage to be observed. Although due to a low signal-to-noise ratio such results could only be obtained after averaging many (approx. 4000) individual experiments. This was in stark contrast to the synchronous case where a single experiment could yield useful information.

While higher level power consumption is not balanced within the secure XAP, the results show how power consumption may be balanced within a single block using dual-rail logic. Data dependent activity at the block level may then be tackled by randomised scheduling of operations or by the insertion of dummy operations to maintain a high level of block utilisation.

7.3 Timing Analysis

The Secure XAP is data driven and theoretically it might be more vulnerable to timing analysis, though we were careful to minimise data dependent timing. To assess this potential weakness we performed ADD operations with the first operand at 0x0000 or 0x0001 whilst the second operand varied between 0x0000 and 0xFFFF. One might expect the maximum delay introduced by the carry propagation for 0x0001+0xFFFF whilst 0x0000+0xFFFF does not propagate the carry so might complete more quickly. We did not see any significant difference between the sets of power curves, at least proving that this feature is not as worrisome as initially expected. Nonetheless, it would be prudent to make use of the random timing countermeasure.

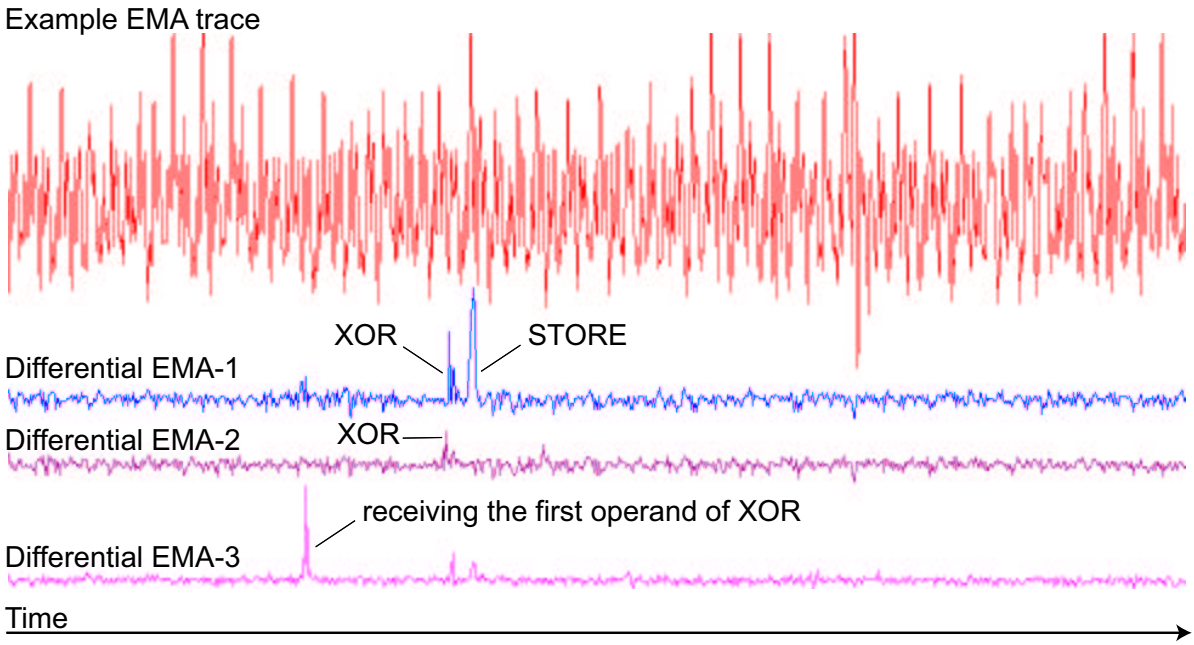

Figure 11. EMA on Secure XAP

7.4 Electromagnetic Analysis

The principle of EMA, as presented in [2, 3], measures the electromagnetic radiation emitted by the active processor. Statistical post-processing of the data is required to correlate samples from successive runs. To ease the evaluation, the provided chips had an aperture through which we had direct access to the Silicon structure.

The EM emissions of both the secure and synchronous processors were recorded using a small coil placed above the chip. Using DPA-like treatment of the data enabled signals of exploitable magnitudes to be recovered in both cases without signal averaging. In fact, due to the absence of clock signal noise, the signal collected from the asynchronous processor was stronger. However, data from the asynchronous processor required some realignment before performing the differential analysis.

In Figure 11, we illustrate the DPA curves obtained with EMA on the Secure XAP. The uppermost curve is one of the curves obtained through the measurement of the electromagnetic waves emitted by the processor. The next one is the DPA curve for the set of results, with significant peaks at the XOR execution and the STORE instruction. The third curve corresponds to DPA on the second operand of the XOR operation. We can note that a peak appears at the time of the XOR execution. And finally, the last curve illustrates what is obtained when working on the first operand: information is mainly leaked

when the data is loaded from memory.

7.5 Optical Fault Injection

Experiments were performed to determine the impact of exposing the secure processor to short bursts of focused laser light. If the dual-rail implementation had provided a completely fault-secure design all attempts at inducing faults would have resulted in deadlock. In many cases the processor did propagate an error signal resulting in deadlock. Unfortunately, two weaknesses were revealed in the current design by the experiments.

The first involved the injection of faults into the ALU design. By targeting two different regions within the ALU, two different fault behaviours were possible. The first was to disrupt the ALU operation to produce an incorrect result, the second forced the ALU to always return the result `0x0001`. These results are possible as some of the dual-rail gates within the ALU do not guarantee that the presence of the error state on their inputs (in this case a *logic-1* on both dual-rail wires) is propagated. This was a known and unfortunate concession due to time restrictions at design time.

Perhaps more interesting is the second failure behaviour. In this case it was possible to set the contents of the processor's registers. The exposure of a single register cell to laser light reliably resulted in setting its value to a *logic-1*. The dual-rail register design that was used in the Springbank chip is illustrated in Figure 12. The ability to set the cell was possible due its inability to store an error state (both states of the single flip-flop are valid). The precise mechanism by which the register was set first involved both the outputs of the `NOR` gates being pulled-low. This was as a result of the laser producing photocurrents in the junctions of the N-type transistors in both gates. When the laser was removed the flip-flop resolved into the *logic-1* state due to differences in the threshold values for each gate. N-type transistors in general produce much larger photocurrents due in part to the superior mobility of electrons when compared to holes [2]. It is important to note that the attack was successful even with a large spot size exposing many transistors (we estimate around 100).

The vulnerability of the ALU design may be countered by ensuring an error state on any gate input is always propagated to its output. An approach to providing a secure dual-rail register design is to increase the number of flip-flops to two to enable an error state to be stored. This also enables a register to be set into the error state preventing any surreptitious use when the register's value is not required (on reset and after its last use in a program).

[2] The electrons and holes in this case are the minority carriers on the larger side of the depletion region. The depletion region extends mostly into the region of least doping.

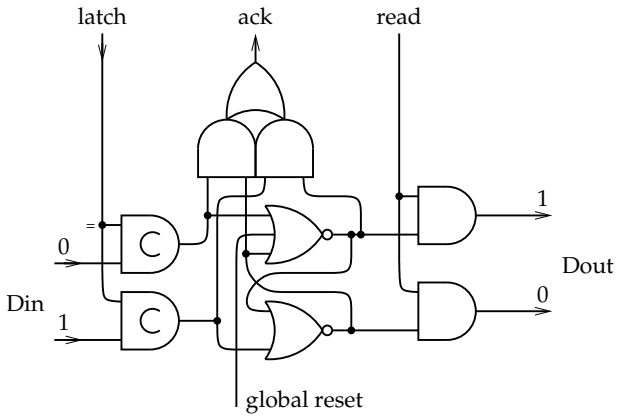

Figure 12. Single Flip-Flop Dual-Rail Register

Security may be improved further by including small optical tamper sensors within each standard cell. These sensors, constructed from one or two transistors, would normally play no part in normal circuit behaviour (only adding a small amount of capacitance). Their only function is to force the dual-rail outputs of the gate into an error state when illuminated. These ideas together with security-driven place-and-route would again increase the level of controllability required to perform a successful modification of register values.

8. Conclusions

Attack testing has revealed strengths and weaknesses in our test chip. Balanced dual-rail logic has been demonstrated to be substantially more robust against DPA which was the primary focus during the design of the test chip. The emergence of more sophisticated EMA and optical probing attacks have been demonstrated to be effective against our test chip. However, where redundancy was present, optical attacks were defeated. To improve security further we need to look at countermeasures like: a top level defence grid, optical tamper sensors, etc. These countermeasures are already available in state-of-the-art smart cards, but will need to be improved upon as the attack technology advances.

The cost of using dual-rail logic is a significant area penalty. Our secure processor requires nearly three times the area of the original synchronous one, though this could be reduced significantly by adding Muller C-elements to the standard cell library. One must also look at the whole system and question how software countermeasures are can be simplified if a more secure processor is used. Software countermeasures can increase the code size by a factor of 2 to 3, which has a substantial impact on the ROM/Flash memory size which in turn has a pronounced impact on chip area. We believe that a more

secure processor provides a better security/area trade-off.

From a performance perspective, the secure processor is slightly slower than the synchronous design at 100 MHz versus 120 MHz. Of course, the asynchronous design requires no external clock or clock multiplier to operate at this rate. Furthermore, the lack of a clock removes the possibility of any clock glitch attack and allows fine-grain timing noise to be introduced into all computations.

Fabrication of the test chip has allowed us to compare measured results with simulated results. We are now in the process of building an ECAD flow which will allow us to perform security analysis at design time.

References

[1] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Proc. 19th International Advances in Cryptology Conference – CRYPTO '99*, pp. 388–397, 1999.

[2] K. Gandolfi, C. Mourtel, and F. Olivier, "Electromagnetic analyis: Concrete results," in *Cryptographic Hardware and Embedded Systmes (CHES 2001)*, LNCS 2162, 2001.

[3] J.-J. Quisquater and D. Samyde, "ElectroMagnetic analysis EMA: Measures and coutermeasures for smart cards," in *Smart Card Programming and Security*, pp. 200–210, LNCS 2140, 2001.

[4] P. C. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems," in *Proc. 16th International Advances in Cryptology Conference – CRYPTO '96*, pp. 104–113, 1996.

[5] E. Sprunk, "Clock frequency modulation for secure microprocessors." US Patent 5404402, filed December 1993.

[6] R. Anderson and M. Kuhn, "Tamper resistance – a cautionary note," in *Second USENIX Workshop on Electronic Commerce*, (Oakland, California), pp. 1–11, USENIX, Nov. 1996.

[7] J.-J. Quisquater and D. Samyde, "Eddy current for magnetic analysis with active sensor," in *eSMART*, Nov. 2002.

[8] S. Skorobogatov and R. Anderson, "Optical fault induction attacks," in *Cryptographic Hardware and Embedded Systmes (CHES 2002) Workshop*, 2002.

[9] H. Ng and D. J. Allstot, "CMOS current steering logic for low-voltage mixed signal integrated circuits," *IEEE Transactions on VLSI Systems*, vol. 5, Sept. 1997.

[10] D. May, H. L. Muller, and N. P. Smart, "Non-deterministic processors," in *Information Security and Privacy, LNCS 2119* (V. Varadharajan and Y. Mu, eds.), pp. 115–129, Springer Verlag, July 2001.

[11] D. May, H. L. Muller, and N. P. Smart, "Random register renaming to foil DPA," in *Cryptographic Hardware and Embedded Systems CHES 2001, LNCS 2162* (C. K. Koc, D. Naccache, and C. Paar, eds.), pp. 28–38, Springer Verlag, May 2001.

[12] S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi, "Towards sound approaches to counteract power-analysis attacks," *Lecture Notes in Computer Science*, vol. 1666, pp. 398–412, 1999.

[13] J. Sparsø and S. Furber, eds., *Principles of Asynchronous Circuit Design: A Systems Perspective*. Kluwer Academic Publishers, 2001.

[14] D. E. Muller and W. S. Bartky, "A theory of asynchronous circuits," in *The Proceedings of an International Symposium on the Theory of Switching*, pp. 204–243, Harvard University Press, Apr. 1959.

[15] D. A. Rennels and H. Kim, "Concurrent error detection in self-timed VLSI," in *Twenty-fourth international symposium on fault-tolerant computing*, 1994.

[16] I. David, R. Ginosar, and M. Yoeli, "Self-timed is self-checking," *Journal of Electronic Testing: Theory and Applications*, vol. 6, pp. 219–228, Apr. 1995.

[17] S. J. Piestrak and T. Nanya, "Towards totally self-checking delay-insensitive systems," in *Twenty-fifth international symposium on fault-tolerant computing*, pp. 228–237, 1995.

[18] D. K. Pradhan, *Fault Tolerant Computing: Theory and Techniques*, vol. 1. Prentice-Hall, 1986.

[19] C. L. Seitz, *Introduction to VLSI Systems*, ch. System Timing. Addison Wesley, 1980.

[20] R. J. Anderson, E. Biham, and L. Knudsen, "Serpent: A proposal for the Advnaced Encryption Standard," 2001. `http://www.cl.cam.ac.uk/~rja14/serpent.html`.

Acknowledgements

The authors would like to thank Vincent Rijmen (at Cryptomathic) for his help with the bus cryptography scheme. Sergei Skorobogatov and Markus Kuhn (University of Cambridge) provided useful insights into smart card attack techniques. Thanks are also due to David Samyde (Université Catholique de

Louvain) for discussions about electromagnetic emissions analysis. Thanks are also due to Christophe Mourtel and Pascal Moitrel (GemPlus) for their help with the attack tests. We are also grateful to Scott Fairbanks (University of Cambridge) for his comments. The Europractice support team at IMEC were very supportive during the tape out phase of our test chip. We thank the European Commission for funding some of this research as part of the project: IST-1999-13515.