An Investigation into the Security of Self-timed Circuits

Z. C. Yu, S. B. Furber and L. A. Plana
Department of Computer Science, The University of Manchester,
Oxford Road, Manchester, M13 9PL, UK
{yuz, sfurber, plana}@cs.man.ac.uk

Abstract

Self-timed logic may have advantages for security-sensitive applications. The absence of a clock, as a reliable timing reference, makes conventional power analysis attacks more difficult. However, the variability of the timing of self-timed circuits is a weakness that could be exploited by alternative attack techniques.

This paper introduces a methodology for the differential power analysis of self-timed circuits which does not rely upon a clock signal. This methodology is used to investigate the security of a self-timed, ARM-compatible processor designed specifically to explore the benefits of self-timed design in secure applications. Timing analysis is also applied to the same design. The results from the analyses are presented and confirm that self-timed logic with dual-rail encoding and secure storage significantly improves resistance to non-invasive attacks.

1. Introduction

Cryptosystem designers usually focus on implementing rigorous algorithms to protect data from logical attacks. They pursue ways to process data securely, however, strong protocols are not necessarily secure [1]. A well-studied cryptosystem may fail for many reasons [2,3]. Figure 1 shows a typical encryption process. The device encrypts the plain text input using a key hidden in a secure on-chip memory and produces ciphered text. The algorithm design ensures that an attacker with access to the ciphered output cannot work out the plain text input without knowing the key, and vice versa. It also guarantees that an attacker with access to both the plain text input and the ciphered output cannot deduce the key.

This assurance is based on the assumption that the attacker has access only to the plain text input and the ciphered output. However, a CMOS VLSI circuit may leak information through side-channels such as timing, power consumption and electromagnetic emissions [4], as illustrated in Figure 1. Attacks that access and exploit such leaked information have been demonstrated [5-11]. Among

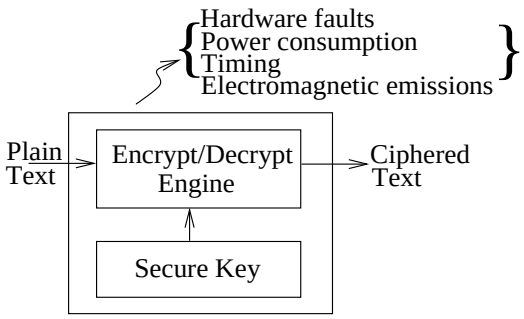

Figure 1: Information leakage in cryptosystems

these techniques, power analysis attacks have been most successful in cracking cryptography implementations.

A power analysis attack is based upon examinations of the power consumed by the system when processing data or executing instructions. By analysing the relationship between the variation in power consumption and the data or instruction, an attacker can discover the secure information being processed and the keys hidden in the system. The success of these attacks could result in an eroding of confidence in smart cards as the bulwark of secure applications. This kind of attack exploits the weaknesses of physical implementation aspects of cryptosystems [12] rather than algorithmic aspects, and is more difficult to defeat [10].

Self-timed or asynchronous circuits have been proposed as being able to reduce side-channel information leakage and thereby to improve security [13,14]. Unlike synchronous circuits, which synchronize all parts of the circuit by using a single global clock signal, asynchronous circuits employ many local synchronization signals. This allows individual circuits to work independently. Thus, there is no global timing signal for use as a reference and the analysis of power consumption is consequently expected to be more difficult.

Dual-rail encoding uses two wires to present a one bit logic value. A transition on one wire indicates the transmission of a logic “one” and a transition on the other wire a “zero”. There is therefore always exactly one transition to

transmit a bit of data, and hence the differential power signature can be significantly reduced.

SPA [15] is a synthesized self-timed ARM-compatible processor designed specifically for smart card applications, and the design aims to investigate the ability of self-timed circuits to resist differential power analysis attacks. The processor was designed using Balsa [16], an asynchronous hardware description language and synthesis system. With a specially-developed secure, dual-rail back-end, the Balsa family of tools was used to produce a gate-level netlist of a secure dual-rail processor, which was then laid out as a custom VLSI device using commercial CAD tools.

Designing circuits to minimize side-channel information leakage and devising attacks to exploit the weaknesses in those designs are complementary activities. This paper introduces techniques that can be used to analyze the power signature leaked by asynchronous circuits.

The paper briefly reviews the power dissipation characteristics of CMOS VLSI circuits and non-invasive attack techniques in the following section. The security features in the SPA processor are discussed in Section 3. The methodology for the power analysis of asynchronous circuits is presented in Section 4. Power and Timing analyses results from SPA simulations and measurements are presented in Section 5. The results and conclusion of the investigation are presented at the end of the paper.

2. Non-invasive attacks

The power consumption of a CMOS VLSI circuit consists of three parts: static dissipation, dynamic dissipation and short-circuit dissipation [17,18]. The dynamic dissipation, Pd, is normally the dominant component and is primarily responsible for information leakage. It is the result of the charging and discharging of the load capacitance, and is given by:

$$Pd = f \cdot C_l \cdot V_{dd}^2 \cdot A_c \qquad \textbf{(1)}$$

where A_c is the circuit activity (the proportion of the total capacitance switching), f is the frequency of switching, C_l is the total circuit capacitance and V_{dd} is the power supply voltage.

From this equation it can be seen that, for a given circuit where the load capacitance and supply voltage are constant, the dynamic power dissipation depends on the circuit activity, A_c. In other words, the more capacitance that is switched, the more power is dissipated.

For the attacker of a secure system, this is the characteristic of CMOS VLSI circuits that inspires power analysis attacks. For example, a multiplier usually does more work when the multiplier operand bit is one than when it is zero. Hence the multiplier dissipates more power processing a

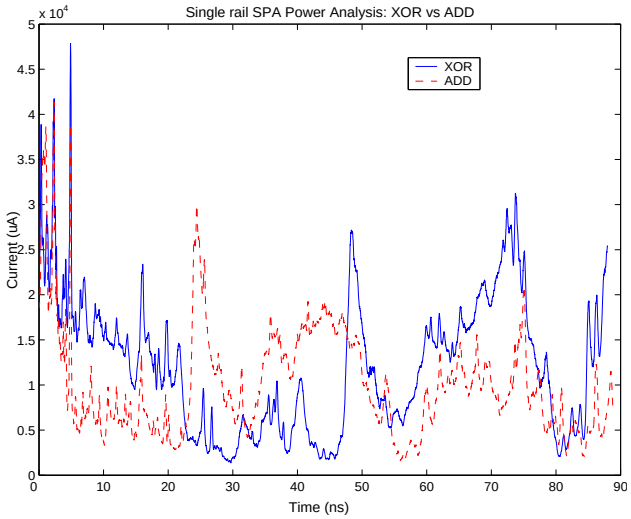

Figure 2: Power consumption of XOR and ADD

multiplier bit which is a one than when it is a zero. If the multiplier operand is a secret key, then power analysis on the multiplication operation yields vital information about the key.

Power analysis attacks [1,8] include Simple Power Analysis (SPA, not to be confused with the processor being investigated!) and Differential Power Analysis (DPA). A SPA attack observes the power consumption waveform directly and measures the obvious peaks which are known (through prior investigation) to relate to security information. SPA attacks are relatively easy to defend against [1].

DPA is much more powerful and difficult to defend against. It uses statistical analysis techniques to maximize the information extracted from power variations by subtracting the average power (which is not correlated to the data) and other extraneous signals from the waveforms. DPA aims to eliminate the power consumption and noise that is not relevant and to amplify the power consumption that is correlated with the secure information.

A timing analysis attack has a similar basis to power analysis attacks [7]. Instead of using differential power consumption, timing analysis uses the leakage of information through data- and instruction-dependent timing variations. If a secure implementation takes a different amount of time to execute different instructions or to process different data, then the secure information can be extracted by analysing the timing differences. For instance, many multiplier architectures employ a clever algorithm to speed up the operation, however the timing differences may leak the hamming weight of the multiplier operands.

Figure 2 shows the power and timing information revealed when executing XOR and ADD instructions on a single-rail SPA processor. (The single-rail SPA is synthesized from the same Balsa source as the secure, dual-rail

SPA but uses a different, less secure back-end.) The solid curve is the power waveform for XOR and the dashed curve the waveform for the ADD instruction. It is clear that the two waveforms are readily distinguishable. Different time is also taken to execute these two instructions - XOR takes less time to execute than ADD (about 88 ns for XOR and 89 ns for ADD). By measuring power and/or timing information attackers may be able to identify the instructions being executed. Similarly, attackers can extract information on the data being processed if there is data-dependent side-channel information leakage.

3. SPA security features

SPA was developed specifically for smart card applications. It was designed with the goal of being resistant to DPA and timing analysis. Although the removal of the clock makes statistical analysis more difficult in asynchronous designs, this alone is not sufficient. To achieve such a goal, SPA employs several additional design techniques to minimize side-channel information leakage. A summary of these features is presented below. Details can be found elsewhere [15].

- Dual-rail encoding is used in SPA to reduce the power consumption signature due to differing data Hamming weights.
- In addition to the balanced signalling inherent in dual-rail codes, a 'null' is inserted between consecutive data bits to ensure that the power consumption for a new data value is independent of the previous data value.
- Although conventional return-to-zero dual-rail encoding has this property for logic functions, SPA extends the property to include balanced secure storage elements for on-chip registers and state-holding storage. Information-bearing state is cleared by returning the stored data to a null state before new data is loaded. This again reduces the leakage of side-channel information.
- Self-timed circuits are often built to exhibit average-case performance. However, this desirable property may leak information. To avoid this problem, the functional units in SPA, such as the adder and multiplier, were designed with data-independent response time.

4. Power analysis of asynchronous circuits

Differential power analysis, reviewed in Section 2, has been well-studied and successfully applied to the attack of secure implementations. However, all the analysis is based upon conventional synchronous circuits where a global clock signal is available for use as a time reference, enabling sophisticated statistical analysis to be used. The global clock signal is not available on a self-timed circuit, introducing new problems for differential power analysis.

Firstly, in synchronous circuits the global clock signal results in the power consumption waveform being periodic. Using the clock as a time reference, the attack is made much easier as it is simple to apply statistical analysis techniques on the power waveforms and to compute the average waveform. Self-timed circuits, on the other hand, have many local handshake signals instead of one global signal. Activities in each subcircuit only depend on their local handshake signals and circuits work independently. Therefore the power consumption waveform does not have a global time reference. This creates a problem for statistical analysis as the waveforms are no longer periodic.

Secondly, because in synchronous circuits all the activity is triggered by the clock edge, there is always a large power consumption peak at a fixed time after each clock edge. In self-timed circuits, each subcircuit is independent from the others. The power consumption of each subcircuit does not contribute to the total power consumption with a consistent relative timing as it does in synchronous circuits. Thus, there may be not large peaks in the total power consumption waveform. Indeed, the power consumption of self-timed circuits will be much smoother since it is the aggregation of many small peaks appearing at arbitrary times according to their handshake signals.

Thirdly, power waveform shifts and overlaps are a problem raised by self-timed circuits for statistical analysis. Handshake signals dominate the behaviour of every part of the circuit. The behaviour of one stage may be affected by the previous stage and even possibly by many other stages. For the same instruction, the cycle time can vary in different periods. It is not necessary for waveforms to have exactly constant periodicity. The statistical analysis of such power waveforms is hard as it is difficult to line them up in order to compute the average value and to subtract this from each waveform to yield the difference.

The upper window in Figure 3 shows two SPA power traces for the same operation and the dashed curve shows their difference. The two traces are clearly shifted and produce a significant difference. However, this curve does not properly account for the real differences between the two waveforms. The temporal shift contributes a large amount of the difference. The lower window shows the corrected waveforms and the difference, which is now much smaller than before. This result indicates that the difference between waveforms contains not only the real difference which attackers want to extract but also the affects of temporal waveform shifting which can be considered as noise.

In conclusion, differential power analysis as applied to conventional synchronous circuits cannot be applied directly to attack self-timed circuits. A new approach is required, as described below.

The power analysis of asynchronous circuits is divided

into two steps: data collection and data analysis. The data collection step involves data sampling and data correction. A single power trace, obtained by monitoring the supply current of the system, must be split into individual instruction or cycle traces. This is relatively easy to do in synchronous circuits with the help of the clock signal. To split an asynchronous power waveform is more difficult because of the lack of a time reference. However, an asynchronous circuit does display some characteristic periodicity due to the cyclic nature of the handshake controls and the algorithms being executed and this periodicity can be used to extract waveforms that are approximately repetitive.

The data samples that are thus collected are, however, still not ready for analysis. The corresponding waveforms must be dilated or clipped to map them all onto the same basic periodic behaviour. A piece-wise, best cross-correlation shift correction function is also applied to temporally shifted waveforms. This must be a piece-wise process as different fragments of the waveform can be shifted by different amounts of time. The shift correction function is implemented based on a cross-correlation function provided in Matlab [19].

The analysis scheme assumes that suitably split waveforms can be obtained and enough control tests performed to obtain characteristic power waveforms for different data cases. Here we show how the analysis scheme is applied to evaluate the extent of information leakage from a circuit - to what extent does the power waveform reveal the data value being processed? We assume, for simplicity, that there are only two cases we need to differentiate, the data 0 case and the data 1 case.

Figure 4 shows the scheme used for the power analysis. Firstly, a partitioning function is used to split the waveforms, P(i,j), into two sets: $P_0(i,j)$ (the data 0 set) and $P_1(i,j)$

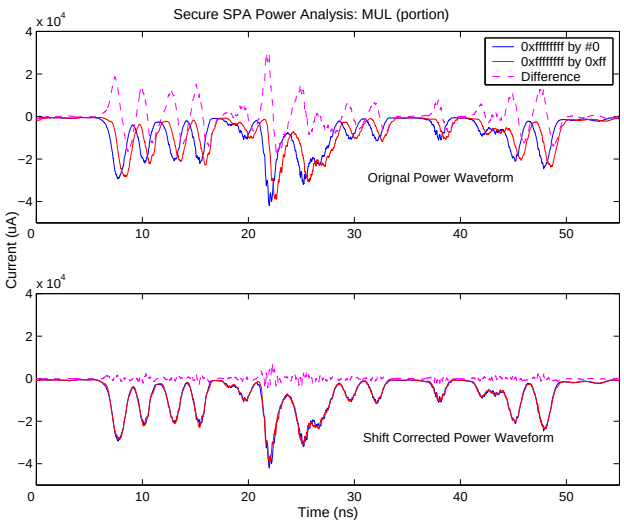

Figure 3: The problem of shifted waveforms

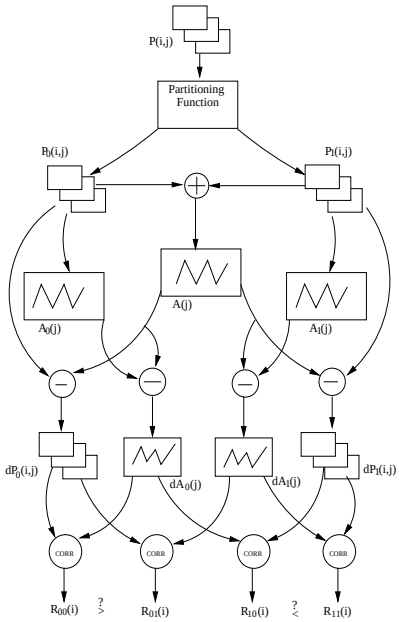

Figure 4: Power analysis methodology used for asynchronous circuits

(the data 1 set). The first index, i, is the waveform sequence number and the second, j, is the sample point in each waveform. Equation 2 is used to compute an average data 0 waveform, $A_0(j)$, from $P_0(i,j)$. Similarly, $A_1(j)$ is obtained from averaging $P_1(i,j)$.

$$A_0(j) = \frac{1}{N}\sum_{i=1}^{N} P_0(i,j) \qquad \textbf{(2)}$$

An overall average waveform, A(j), is obtained by averaging the whole data set comprising P(i,j). Subtracting A(j) from $A_0(j)$ yields the difference between the average waveform of the data 0 set and the overall average following Equation 3. Similarly, $dA_1(j)$ can be acquired. In a power waveform where the consumption for one data pattern is distinguishable from that for the other data pattern, $dA_0(j)$ should be distinguished from $dA_1(j)$.

$$dA_0(j) = A_0(j) - A(j) \qquad \textbf{(3)}$$

Then subtracting the overall average waveform, A(j), from each data set, P(i,j), yields the power difference waveforms, dP(i,j), as shown in Equation 4.

$$dP(i,j) = P(i,j) - A(j) \qquad \textbf{(4)}$$

If P(i,j) belongs to the data 0 set, dP(i,j) should show a strong correlation with $dA_0(j)$ and a weaker correlation with $dA_1(j)$. If it belongs to the data 1 set the opposite should be true.

The analysis technique yields a correct ‘deduction’ if the

above assertion is true; otherwise it yields a wrong deduction. The use of a correlation analysis is necessary because an asynchronous power waveform does not display the very large peaks that characterise synchronous power waveforms and reveal the data being processed. Instead, many small peaks can be found on the waveform. Subtracting the average makes the differences more relevant, but the waveforms are still noisy.

The percentage of correct deductions (PCD) is used to evaluate the success of the analysis. For an analysis of the experiments outlined above, where there are only two alternative data values, a PCD of 100% would indicate perfect information extraction whereas a PCD of 50% would indicate that the analysis was extracting no information as the results are essentially random.

The scheme described above illustrates how much information attackers may be able to extract from the side-channel. However, it does not explain how difficult the analysis would be. A 'signal-to-noise ratio' shows how large a side-channel information signal is compared to the mean power waveform that contains no relevant information, and from which it must be extracted. It gives some indication of the dynamic range required of the attacker's equipment and the likelihood that extraneous electrical noise might obscure the information-bearing signal.

The SNR is computed as in Equation 5:

$$SNR = 20log_{10}\left(\frac{S}{N}\right) \quad (5)$$

Here S and N are the root-mean-square of the (information-bearing) signal and the (average power waveform) noise respectively. From the equation it is understood that the smaller the SNR is the more difficult the analysis is.

5. SPA security analysis

The experimental chip designed to evaluate the resistance of asynchronous designs to non-invasive attacks contains both a single-rail SPA and a secure dual-rail SPA so that their security properties can be compared. To evaluate the maximum potential side-channel information leakage, an optimal experimental environment was set up for the attacker:

- A simple program that continuously executes the targeted instruction was run in each experiment and only two different data values were used, minimising the difficulty of the attack. For example, to analyse the ADD instruction the program can execute:

  ADD r3, r2, r0
  ADD r3, r1, r0

  where r3 is the destination register, r1 contains 0xff, r2 contains 0xffffffff and r0 contains 0x1. To eliminate interference between the two instructions, several no-op instructions are inserted before, between and after the two ADD ARM instructions [18]. The difference between the two instructions is in the different carry propagation distance. One has 8-bit and the other 32-bit carry propagation. This simple test sequence is used to measure whether there is information-bearing side-channel leakage resulting from the difference in carry propagation.
- The knowledge of which instruction is being executed and which data values are used was made readily available to the attacker, so that an optimal partitioning function can be applied.
- A signal (execCtrl) is artificially made available to the attacker to allow optimal synchronization of the power traces. The rising edge of the signal indicates the beginning of the execution stage of an instruction.

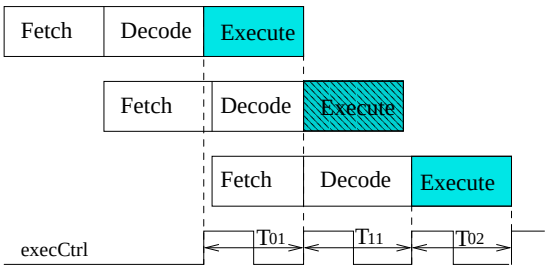

Figure 5: The SPA 3-stage pipeline

Figure 5 shows the SPA 3-stage pipeline signal along with the timing of execCtrl. In a self-timed processor every stage of the pipeline is no longer required to take exactly the same time; it can take different times to execute different instructions. There may be fine-grain timing noise [13], but this can be a source of side-channel information to attackers. The following sections discuss this further.

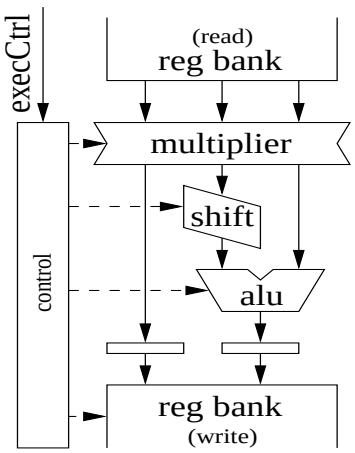

Figure 6: The SPA execution stage

In a real attack, the environment available to the attacker would presumably not be so helpful, therefore much more effort would be required to undertake such an analysis. The absence of the timing reference signal would particularly complicate the attacker's task. Consequently the results obtained here can be considered pessimistic in terms of what they say about the intrinsic security of the SPA system. Nevertheless, they provide a baseline measure of the relative security of the system.

5.1. Differential power analysis of SPA

In this section we employ the asynchronous power analysis methodology introduced in Section 4 to evaluate the effectiveness of the measures taken in the design of SPA to protect it against attacks.

The analysis of SPA is based upon power waveforms taken from simulations and actual measurements of SPA instruction execution. In the analysis of simulations, the traces were partitioned using the execCtrl signal introduced earlier. In order to understand these waveforms, the SPA execution unit is described in detail. It is illustrated in Figure 6 and includes three stages: pre-control, execution and write-back.

The pre-control stage determines whether an instruction is executed. Some instructions, for example conditional branches, might not need to be executed. Only if an instruction is executed is the execCtrl signal triggered, otherwise it remains inactive.

The second stage, the execution stage, can be broken down into sub-stages: register read, multiply, shift, ALU and return-to-zero. The register bank in SPA comprises 16 32-bit registers as it is designed for ARM compatibility. The multiplier is transparent if the instruction does not need a multiplication operation. The shifter in the SPA execution unit contains five shift stages with different shift distances: 1, 2, 4, 8 and 16 bits. The ALU consists of XOR, AND, ADD and OR basic functions. To maximise security an instruction passes through the ALU even if it does not require any ALU operation - not to do so would reveal information in the power and timing signatures.

The last stage, write-back, writes results back into the register bank.

For further partitioning of the power waveform, timing information for each of the stages is presented in Figure 7. About 21% of the time is used in the pre-control stage and about 8% of the time is for register write-back. The majority of the time, about 71%, is used to execute the instruc-

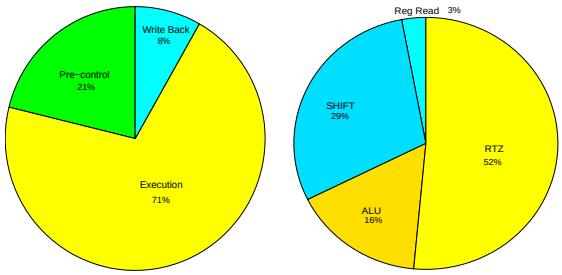

Figure 7: SPA execution time breakdown

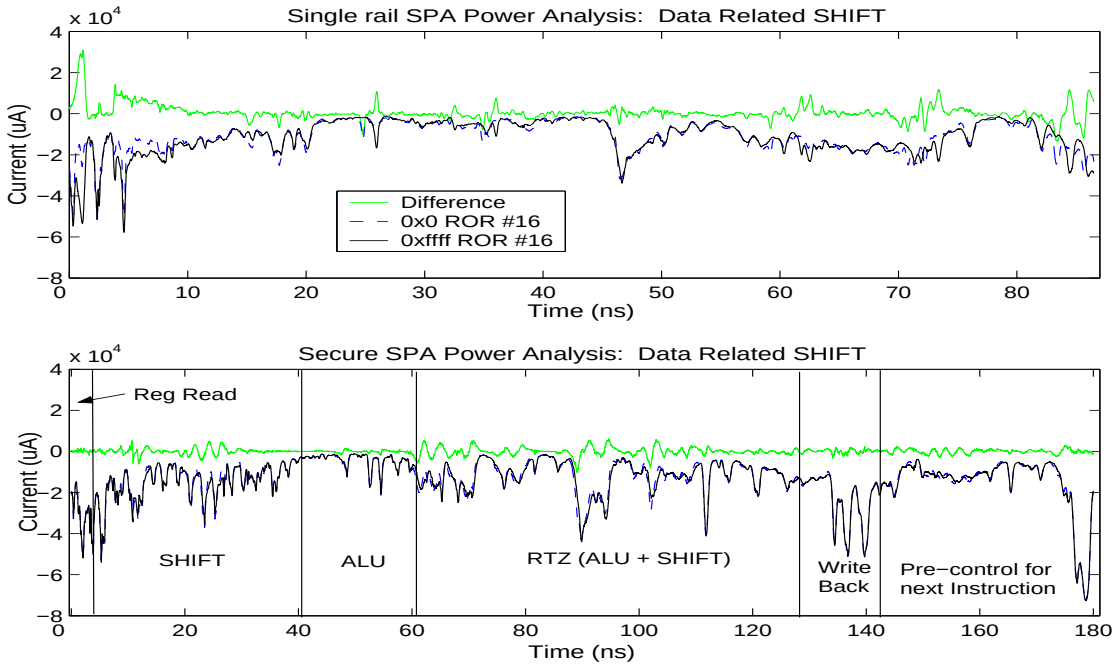

Figure 8: Differential power waveform of the Data-related SHIFT instruction

tion. The right chart is a further breakdown of the execution stage timing. About 3% of the execution time is spent in register read, 29% in the shift operation and 16% in the ALU, while almost half of the time is used for the execution unit to return to zero. The multiply operation is not included in this case and the execution unit takes a longer overall time if a multiplication is being performed.

The power analysis of SPA looks at six instructions: a shift with varying distance (distance-related shift), a shift with varying data (data-related shift), a branch, an xor, an add and a multiply. These were selected as they are the most likely to generate different power consumption. The data-related shift instruction is explained in detail below while the results from the other instructions follow.

The data-related shift instructions use the same shift distance of 16 bits, but shift different data values of 0x0 and 0xffff. We observe the differential power consumption caused by processing different data. To isolate the effects of the previous and following instructions, three no-op instructions are inserted between the instructions.

Figure 8 shows power consumption traces of SPA executing these shift instructions. The upper window shows the single-rail SPA and the lower window the secure, dual-rail SPA. There are three traces in each window. The dashed power trace is executing a right shift of 0x0 by 16 bits and the solid trace a right shift of 0xffff by 16 bits. The two curves are the average (mean) waveform of hundreds of simulation loops. The difference in power consumption is shown as the lighter solid trace. The secure SPA waveform is split into sections according to the timing breakdown described previously.

The first part of the waveform is the register read operation which accesses the register operands in the register bank. There is almost no evidence of a power difference in the secure SPA, confirming the safety of the dual-rail registers. An obvious difference appears on the single-rail SPA. This is also true for the register write-back period.

The shift operation generates different power spikes on both single-rail and secure SPA. Shifting 0xffff consumes more power than shifting 0x0 on the single-rail SPA as the power consumption difference is mainly positive. In contrast, the power consumption difference on the secure SPA is much smaller and oscillates around zero.

During the ALU phase the shift instruction does not require any operation in the ALU. The ALU is still active, however, and there is clearly power consumed during this period. The secure SPA does not exhibit any significant differential power signature, but again there is a very clear spike on the single-rail SPA.

The large region between the ALU and register write-back phases is for the execution unit to return to zero in order to clear the information stored by the present state. In this period, the differential power curves exhibit small

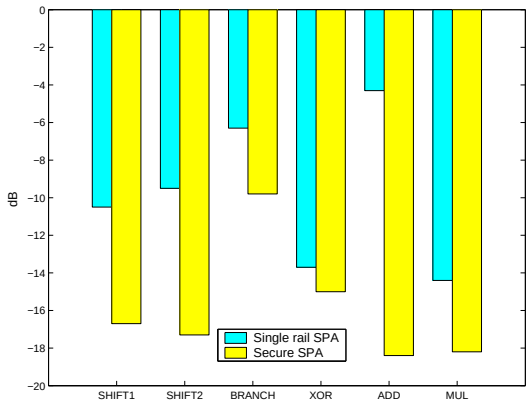

Figure 9: SPA power analysis: signal to noise ratio

spikes on both the single-rail and secure SPAs. Although they are small compared to the average waveform they may still leak information. Fortunately, during this period the power is less correlated with the data.

The last region on the waveform is the pre-control unit for the next instruction. Because the execCtrl signal is generated after the pre-control stage, the power waveforms partitioned using this signal begin with the execution stage and end with the pre-control stage for the next stage. In this experiment, the next instruction is a no-op which moves data from one register to the same register. Even then a spike is still shown on the single-rail SPA.

The signal-to-noise ratio of the data-related shift instruction is the group of bars named SHIFT2 in Figure 9 where the SNR of the single-rail SPA is the darker bar to the left of each pair and that of the secure SPA is the lighter bar to the right. The SNR of the single-rail SPA executing the data-related shift is about -10dB whilst that of the secure SPA is about -18dB. In a environment where external noise was a factor in determining the difficulty of extracting the information, the secure SPA could clearly be much harder to analyze than the single-rail SPA.

The other groups of bars show the SNRs of the remaining instructions being measured. Apart from the branch instruction, where the SNR is greater than -10dB for both the single-rail and the secure SPA, the secure SPA SNRs are around -18dB. In contrast, the SNRs are about -10 dB for the single-rail SPA.

Figure 10 shows PCD results indicating the extent of information leakage, derived from the application of the power analysis scheme discussed in Section 4 to actual power traces obtained from SPA. The left most bar of each group represents the single-rail SPA. In every experiment, single-rail SPA measurements yield a 100% correct deduction, disclosing poor resistance to differential power analysis. By contrast, the right most side bar represents the secure SPA which produces a much lower PCD in most cases, apart from the BRANCH and distance-related

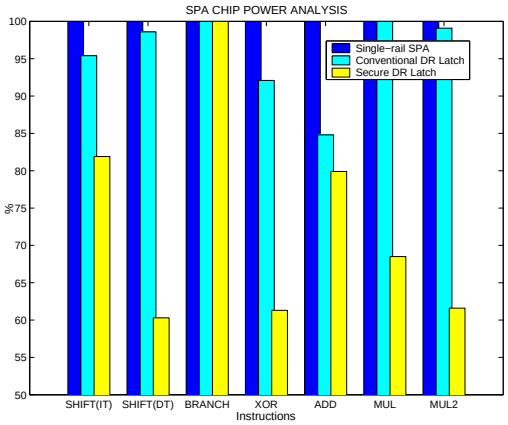

Figure 10: SPA Power Analysis: percentage of correct deduction (PCD)

SHIFT instructions with (nearly) 100% PCD. This means that such instructions are still vulnerable on the secure SPA.

To evaluate the impact of the different security-enhancing techniques [15], the secure SPA was built using configurable latches, i.e., latches that can be configured to operate in secure or conventional mode. The same experiments were repeated with the configurable latches set as conventional ones. The middle bars in Figure 10 show the PCD for the secure SPA with insecure conventional dual-rail latches. These results indicate that the secure SPA with the conventional latches improves security slightly over the single-rail SPA, suggesting that the use of dual-rail encoding and balanced logic without the secure latch is not likely to deliver adequate security. The results clearly show that the secure latches have a large impact on the PCD, indicating that their use can result in significantly enhanced security.

5.2. Timing Analysis

The theory for timing analysis is similar to that for power analysis. Power analysis uses information leakage though data- or instruction-dependent power consumption. Timing analysis measures data- or instruction-dependent timing differences.

Our timing analysis is based upon running one instruction with two different data values alternately N times. The timing information is obtained using the execCtrl signal, measuring the time from one rising edge to the next (that is, from the beginning of the execution of one instruction to the beginning of the execution the next instruction as shown in Figure 5). If the times for the first and second executions are T_{01} and T_{11} respectively, the timing information T_{ij} extracted from the signal can be split into two sets using a partitioning function:

$$T_0 = \{T_{ij}\}, i = 0 \quad \textbf{(6)}$$

$$T_1 = \{T_{ij}\}, i = 1 \quad \textbf{(7)}$$

Then, the average time for each set is computed:

$$TA_0 = \frac{1}{|T_0|} \sum_{T_{ij} \in T_0} T_{ij} \quad \textbf{(8)}$$

$$TA_1 = \frac{1}{|T_1|} \sum_{T_{ij} \in T_1} T_{ij} \quad \textbf{(9)}$$

Where $|T_0| + |T_1| = N$. Subtracting the two average times, a timing difference is obtained:

$$\Delta T = TA_0 - TA_1 \quad \textbf{(10)}$$

For a chosen partitioning function $f(I,D_i)$ corresponding to time consumption T_i, an ideal secure implementation would require the variation, ΔT, to be zero. However, if the time consumption is data-dependent, the chosen partitioning function will yield a non-zero variation. An attacker who can extract this variation, will probably be able to extract the data value. The larger the variation, the easier the data is extracted. A system that takes different lengths of time processing different data values potentially leaks timing information which an attacker will want to discover.

The timing analysis results of an ADD instruction executed by both SPAs are shown as Figure 11. The left hand chart is the result for the secure SPA and the right hand chart is for the single-rail SPA. The x axis displays the range of timings which each SPA takes to execute the instruction and the y axis indicates the number of cycles which fall into the respective bins. The darker bar in each chart is for executing an ADD instruction which causes an 8-bit carry propagation and the lighter bar a 32-bit carry propagation. From the figure, it is obvious that the two sets of bars are almost fully overlapped for the secure SPA. This means that using an ADD instruction to process two different data consumes almost exactly the same amount of time. For the secure SPA, the average time to execute an ADD instruction for both data causing 8- and 32-bit carry propagation is about 191.5 ns. In contrast, the two sets of bars are totally separated on the other chart for the single-rail SPA. The times fall into two groups, one around 89.3 ns and the other around 95.1 ns. The difference between the two average times is obvious and is about 6%. The larger variation clearly indicates that the single-rail SPA leaks information much more readily than does the secure SPA.

The other conclusion that can be drawn from the figure is that processing the same data with the same instruction does not always take the same amount of time. For instance, the time to execute the ADD instruction with an 8-bit carry

propagation on the secure SPA varies from 191ns to 192ns while most of the cycles reside at the centre at about 191.5ns. In synchronous circuits, timing variations are usually the result of noise. However, in asynchronous circuits they result not only from noise but also from the effects of handshake protocols. A handshake stage is affected by its neighbouring stages. Even though the same instruction and data are processed, there is still the possibility to take a slightly different time. This feature is also visible on the power consumption waveform. This confirms that the self-timing behaviour of asynchronous circuits contributes to making attacks more difficult.

Average times for the observed instructions are shown in Table 1. SHIFT(1) is the right rotate shift with different shift distances and the same data, while SHIFT(2) is the data-related shift instruction with the same shift distance. The BRANCH instruction measures the different timing information when the processor does or does not jump. XOR is executed with one constant operand (0xff) and one of two other values, 0x00 and 0xff, to produce outputs of either 0xff or 0x00. This measures whether the timing information is correlated with the Hamming weight of the result. The ADD instruction measures the relationship between carry propagation and timing as discussed previously. The last instruction measured is MUL, the multiply instruction. The multiply instruction used in SPA always performs 32 iterations regardless of the operand values.

In the single-rail SPA results it is obvious that the timing varies when different instructions are executed. For example, MUL takes three times as long as a shift instruction. For each individual instruction, the time varies when different data is processed. For example, ADD reveals a 6% time variation when processing the two different data values. In the secure SPA results, it is clear that timing differences still

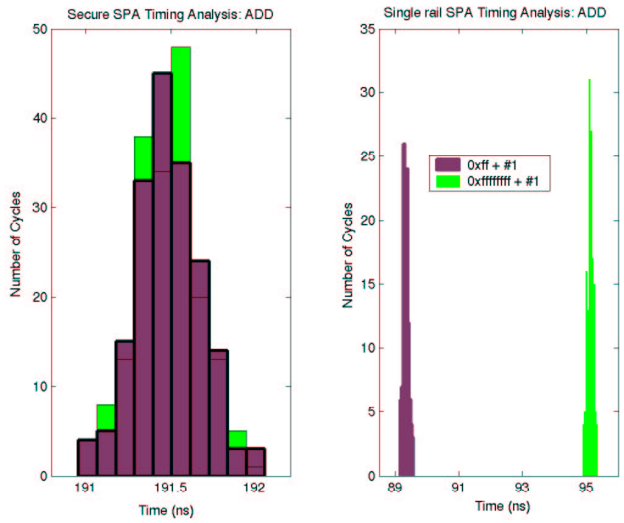

Figure 11: SPA ADD instruction timing analysis

Instructions	Average time(ns) (Single-rail SPA)		Average time(ns) (Secure SPA)	
	TA_0	TA_1	TA_0	TA_1
SHIFT(1)	81.64	82.49	175.22	180.02
SHIFT(2)	86.50	87.66	180.12	180.11
BRANCH	48.49	226.68	92.09	462.63
XOR	88.11	89.37	174.11	174.19
ADD	89.35	95.11	191.49	191.49
MUL	244.93	245.13	2192.49	2193.38

Table 1: Average instruction times

exist with some instructions, particularly for SHIFT(1) and BRANCH. For the other instructions the timing variation is almost eliminated. The BRANCH instruction displays very different timing as it is not executed if the branch condition is false. Clearly it will take much longer to execute when it does result in a jump operation. Thus, the BRANCH instruction will always reveal whether its condition is true or false.

	Single rail SPA(%)	Secure SPA(%)
SHIFT(1)	100	100
SHIFT(2)	100	51.9
BRANCH	100	100
XOR	100	60.0
ADD	100	51.0
MUL	91.8	76.1

Table 2: SPA timing analysis: probability of correct deduction

As for power analysis, PCD is used to determine how much information an attacker can extract from the different timing information using a partitioning function. In this experiment, as there are only two possible choices, the worst deduction would be 50% correct, which can be obtained by random choice. For the ADD instruction, the attacker managed to extract the data correctly (100%) by measuring the different time information when the single-rail SPA is analysed, which confirms its poor resistance to a timing attack. However, the same analysis of the secure SPA yields only 51% correct deduction, barely better than

random guessing.

Table 2 shows the timing analysis results. Apart from the MUL instruction, which gives 91.8% correct deduction, the attacker managed to extract 100% correct data from the single-rail SPA. However, the attacker only obtained 100% correct data with the secure SPA when executing distance-related shifts and branch instructions. The data-related shift, ADD and XOR instructions give data only slightly better than random guesses. However, the MUL instruction still leaks about 76.1% accurate information.

6. Conclusions

We have introduced a methodology for performing power and timing analysis on asynchronous power waveforms. The results from a security analysis of two variants of a self-timed, ARM-compatible processor have been presented. The results indicate that the secure SPA has significantly improved resistance to non-invasive attacks compared to the single-rail SPA. We have confirmed that self-timed logic has the potential to improve the resistance of CMOS VLSI devices to such attacks.

The removal of the clock creates difficulties for would-be attackers using statistical analysis techniques. These difficulties include the lack of a time reference, misaligned waveforms and random timing noise. Dual-rail encoding and balanced logic with secure latches dramatically reduces the differential power signature, though great care is still required to maintain balance and to avoid revealing information through unintentional timing variations, such as during layout, balanced routing of dual-rail signals and matched transistors in all inputs of standard cells used in design.

From these experiments we have also learnt that some instructions, such as BRANCH and distance-related SHIFT, leak significant information. This knowledge is useful in guiding the programmer of the present SPA and the designer of an enhanced future variant. In a secure application of the current design, conditional instructions should avoid depending on secure information such as a key, as they reveal the condition in their power and timing signatures. The processor could be redesigned in the future to ensure that it always does similar work whether or not a condition is passed, so that it has similar timing and power characteristics and leaks less information. For example, a conditional BRANCH instruction could always branch, either to the target or to the next sequential instruction, rather than skipping the branch execution as in the present SPA.

It is clearly possible to design a self-timed processor that is highly resistant to timing and power-analysis attacks. The present secure SPA design demonstrates both strengths and weaknesses in this respect, but it clearly points a way forward for this technology.

7. References

[1] Kocher, P.; Jaffe, J. and Jun, B., “Introduction to Differential Power Analysis and Related Attacks”, http:/www.cryptography.com/dpa/, 1998.

[2] Anderson, R., “Why Cryptosystems Fail”, Communications of ACM, 37(11), pp 32-40, 1994

[3] Anderson, R. and Kuhn, M., “Tamper resistance-a cautionary notice”, Proc. Second USENIX Workshop on Electronic Commerce, Oakland California, 1996

[4] Anderson, R., “Security Engineering-- a Guide to Building Dependable Distributed Systems”, Chap15 Emission Security, WILEY

[5] Anderson, R. and Kuhn, M., “Low Cost Attacks on Tamper Resistant Devices”, Proc. Security Protocols Workshop, Paris, 1997

[6] Biham, E. and Shamir, A., “Differential Fault Analysis of Secret Key Cryptosystem”, Proc. CRYPTO’97, pp 513-525

[7] Kocher, P., “Timing attacks on implementations of Diffie-Hellman, RSA,DSS and other system”, in Proceeding of CRYPTO’96, Santa Barbara, August 1996

[8] Kocher, P.; Jaffe, J. and Jun, B., “Differential Power Analysis”, Proc. CRYPTO’99, Santa Barbara, 1999.

[9] Kommerling, O. and Kuhn, M.G., “Design Principles for Tamper-Resistant Smartcard Processors”, USENIX Workshop on Smartcard Technology, Chicago, May 1999.

[10] Messerges, T.S.; Dabbish, E.A. and Sloan, R.H., “Investigation of Power Analysis Attacks on Smartcards”, USENIX Workshop on Smartcard Technology, Chicago, May 1999.

[11] Skorobogatov, S. and Anderson, R., “Optical Fault Induction Attacks”, Proc. Workshop on Cryptographic Hardware and Embedded Systems, San Francisco, 2002

[12] Weingart, S., “Physical Security Attacks and Defences for Computing Systems”, Secure System and Smart Cards, IBM T.J.Watson,

[13] Moore, S.; Anderson, R.; Cunningham, P.; Mullins, R. and Taylor, G., “Improving Smart Card Security using Self-timed Circuits”, Proc. Async2002, Manchester, 2002

[14] Yu, Z.C. and Furber, S.B., “Defeating Power Analysis Attacks”, Proc. of 9th UK Asynchronous Forum, Cambridge, December 2000.

[15] Plana, L.A.; Riocreux, P.A. and et al. “SPA - A Synthesizable Amulet Core for Smartcard Applications”, Proc. of Async’02, Manchester, April 2002

[16] Bardsley, A., “Implementing Balsa Handshake Circuits”, Ph.D thesis, University of Manchester, 2000

[17] Weste, N. and Eshraghian, K., “Principles of CMOS VLSI Design - A Systems Perspective SE”, pp 231-238.

[18] Furber, S.B., “ARM System-on-Chip Architecture”, ISBN 0-201-67519-6, Addison Wesley, 2000.

[19] The Mathworks Inc., “The student Edition of Matlab”, ISBN 0-13-855974-0, Prentice Hall.