An Asynchronous PLA with Improved Security Characteristics

Petros Oikonomakos, Simon Moore
University of Cambridge, Computer Laboratory
William Gates Building, 15 JJ Thomson Avenue
Cambridge CB3 0FD, United Kingdom
{po230,swm11}@cam.ac.uk

Abstract

Programmable logic arrays (PLAs) present an alternative to logic-gate based design. We propose the transistor level structure of a PLA for single-rail asynchronous applications. The geometrically regular layout together with the deployment of dynamic logic help us fine-tune the PLA to enhance its resistance to side-channel attacks, while parity prediction and checking is employed to protect against malicious fault injection. Finally, we demonstrate how our PLAs can be used as building blocks of large-scale systems with good security characteristics, when combined with special return-to-zero asynchronous latches.

1. Introduction

Programmable logic arrays can be used instead of custom logic in VLSI systems [10, 11]. The regularity of a PLA layout makes its timing predictable and controllable. This has prompted researchers to adopt PLA structures to achieve "Timing Closure by Design", thus reducing design time [5]. Previous asynchronous programmable logic work focused on architecture-level full-scale FPGA configurations [1]. More recently, dynamic logic has been employed within asynchronous field-programmable logic devices [9, 8]. We provide an alternative dynamic logic based PLA. The structure promises to offer quick timing closure for asynchronous designs, thus providing a powerful building block for the timing-critical parts of both microprocessors [5] and other large-scale systems, including architectures similar to [1, 9, 8]. Furthermore, the predictable nature of regular logic is relevant not only to timing but also to power consumption. Using a PLA one can more easily predict, control and *balance* power consumption, since parasitics are more likely to be "balanced by construction" in a regular structure rather than in a randomly placed and routed standard-cell based design. From the hardware security point of view, this is very interesting, since it can contribute to the production of large-scale systems with *predictably balanced* power consumption, thus providing defence against power analysis attacks "by design".

Two recent works [3, 6] focused on dual-rail logic for security purposes, since by nature it tends to offer balanced power consumption *and* fault detection capabilities. As an alternative, in this paper, we are seeking balanced consumption in a *single rail* configuration by exploiting the regularity of a PLA structure, while performing *parity prediction and checking* to protect against fault injection. More specifically, given *n* logic equations, we supplement them with an additional equation whose output maintains the parity of the output vector, and implement all equations using an *n*+1 output PLA. Subsequently, we apply an additional small single-output PLA that performs parity checking, while simultaneously we store the *n* useful results in *n* asynchronous latches. The latches themselves are modified for balanced power consumption.

Section 2 of this paper presents the PLA prototype from the synchronous domain and describes the modifications required for asynchronous operation, as well as power balancing refinements. Section 3 proposes an asynchronous fault indicating data processing stage, based on the presented PLA. Section 4 shows and discusses simulation results on a case study, while section 5 concludes the paper.

2. The proposed PLA

Figure 1 shows the structure of the proposed PLA. Like all similar configurations, it realises logic sums-of-products by using two NOR functions (AND and OR planes) and suitably following De Morgan's law. For example, $ab+cd$ is implemented as $\overline{\overline{(\overline{a}+\overline{b})}+\overline{(\overline{c}+\overline{d})}}$, requiring two AND and one OR plane elements. The synchronous version was first presented and analysed in [11]. The figure depicts the point at which the clock is applied in the synchronous version; in this work, the global clock is substituted with the "Request" signal from the previous asynchronous stage (shown in the

figure as Req_internal). The PLA is implemented using dynamic CMOS logic [12] and as such works in two phases, namely "precharge" and "evaluate". When Req_internal=0, the precharge phase is triggered, and points X1 and X4 are driven to Vdd. In the following evaluate phase, triggered by Req_internal=1, the pull-down network of NMOS transistors in the AND plane determines the logic value at X1. After two inverter delays, this value propagates to the OR plane through the interplane buffer composed of the NAND gate and inverter INV2. The OR plane pull-down network then determines the final PLA output. Capacitors C1 – C3 model parasitics, corresponding to long lines in the actual layout [11]. Capacitor C4 models the output load.

The C-elements implementing 4-phase single-rail handshaking are also shown, together with the required delay element (four inverters are shown for the illustration – more or less can be used as required). Evidently, in the asynchronous operation context the PLA is treated as combinational hardware handling bundled data coming from an asynchronous latch. The PLA output is also considered to feed the latch of the next logic stage.

While the PLA operation described above is typical of dynamic CMOS logic, the design of Figure 1 also includes some non-standard elements. Firstly, the first inverting element of the interplane buffer is not a pure inverter but a NAND gate. This ensures that the voltage at point X2 is the logic inverse of X1 only in the evaluation phase. During precharge, the voltage at X2 is kept high, therefore point X3 is kept low and the need for a ground switch in the OR plane is eliminated. This mechanism both speeds up the OR plane, and saves power, since it minimizes the switching activity in the interplane buffer. It was first proposed in [10]. The second non-standard technique is the charge sharing phenomenon exploited in the AND plane. Notice transistor MN1. In effect, it is the ground switch of the AND plane, but it has been moved between the precharge PMOS and the NMOS implementing the function. As soon as Req_internal goes high, capacitor C1 transfers some of its charge to C2 through MN1, regardless of the input pattern. If any of the MN2_i NMOS transistors are on, then the rest of the charge in C1 will be transfered to ground and X1 will be driven low. The charge sharing effect thus speeds up the discharge process and the overall PLA evaluation phase. If all MN2_i transistors are off, then C1 loses some charge to C2; this charge is replenished when transistor MP2 is turned on, since X2 is driven low. Thus, the design continues to operate correctly. In the next precharge phase, transistor MN3 turns on and discharges C2.

The addition of two inverter delays between the activation of the AND and OR planes in Figure 1 is our own modification to the original structure of [11]. Indeed, in [11] both planes were activated simultaneously by the system clock (equivalent to Req_internal). Simulation showed that this created unnecessary activity on the interplane buffer, consuming power needlessly.

2.1. Power consumption experiments

In order to study the PLA of the previous subsection better and, most importantly, evaluate its security characteristics, we conducted a number of HSPICE simulations, on an elementary asynchronous PLA cell, comprising three primary inputs, a single AND term and a single OR term. We targetted a UMC $0.18\mu\mathrm{m}$ CMOS technology. The top graph of Figure 2 shows the power consumption profile of the cell, for arbitrary values of the primary inputs. The two leftmost and the one rightmost power spikes correspond to power dissipated at the control logic of the structure and are therefore of minor interest in the security context, since they leak no data dependent information. The two important power consumption areas are the ones corresponding to the evaluate and precharge operations of the PLA. Each area comprises a number of distinguishable spikes, corresponding to power dissipation in the AND plane, the interplane buffer and the OR plane, separated in time due to the delayed activation of the OR plane (Figure 1). The bottom graph of Figure 2 shows the integral of the "evaluate" and "precharge" power dissipation areas, corresponding to the potentially data dependent *energy* consumed by the PLA.

Figure 3 shows the power and energy traces of the cell for the four possible combinations of the input signals, namely for all three inputs at 0 (experiment A), one input at 1 and the other two at 0 (exp. B), two inputs at 1 and one at 0 (exp. C), and finally all inputs at 1 (exp. D). The graphs show that the traces are practicaly identical whenever at least one of the inputs is at 1. Indeed, experiments B, C and D produce virtually non-distinguishable simulation results. The situation is different if all inputs are at 0. Indeed, the power trace corresponding to the evaluate phase of experiment A produces an additional spike not seen in any other experiment (pointed in Figure 3 by an arrow), while the subsequent precharge phase of experiment A produces a spike that can be measured to be 34.5% shorter than the corresponding spike in experiment B (the two spikes are also shown in the graph by arrows). There is also a small disparity in energy consumption, but this can be ignored as it is only of the order of 2%.

To explain these observations, refer back to Figure 1. Whenever one *or more* of the inputs to transistors MN2_i is at 1, point X1 is discharged in the evaluate phase, thus keeping the NAND gate and inverter INV2 at the precharged values. In contrast, if all inputs are 0, then the NAND gate will be fed by two 1s, thus consuming power by pulling down, forcing also INV2 to pull up and consume more power. In other words, when all inputs are at 0 there is switching activity on the interplane buffer during "evaluate", and this cre-

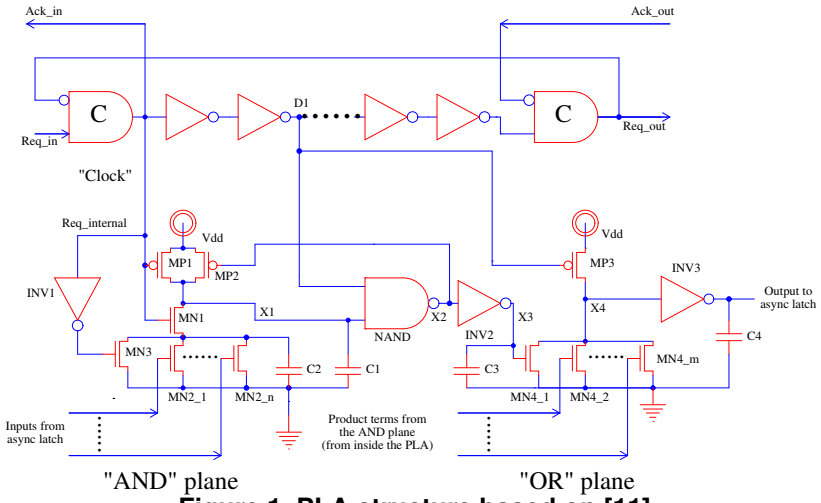


Figure 1. PLA structure based on [11]

ates the additional spike pointed at in the top graph of Figure 3. As regards the disparity of spikes in the precharge phase, notice that during that phase there always exists a short period of time during which the NAND gate has to output 0, since it is fed by two 1s (because X1 is precharged to 1, while the logic 0 value has not yet reached D1). If some of the previous primary input values are 1, this means switching activity, since the previous "evaluate" phase should have driven the NAND to 1. Simply put, during the *precharge* phase, input combinations B, C and D experience *the same* switching activity that combination A experiences during the *evaluate* phase. Hence, both cases consume practically equal energy, but at different time points, which leads to imbalanced power traces. This is a security hazard.

2.2. Balancing power signatures

The simulations of the previous subsection have shown that the considered PLA design has interesting security properties, because it offers naturally balanced power and energy consumption for all input combinations *but* the all 0s pattern. We now weaken this disparity, while retaining practically equal energy consumption.

The obvious way to balance out the differencies of Figure 3 is to increase the power consumption in the precharge phase of experiment A and in the evaluate phase of experiments B, C and D by appropriate amounts. Figure 4 shows our solution for this. In the figure, the original PLA cell is supplemented with two small power consuming networks, comprising three transistors each. Further, two inverters are added to the delay line of the PLA (between points D1 and D2); these inverters are deliberately faster than the rest of the inverters in the line. This keeps the overall delay at acceptable levels, while creating a data-independent power spike, as will be seen in Figure 5. For the time being, focus on networks N1 and N2. All transistors in N1 are open only when Req_internal=0, D1=1 and X2=0. The first two conditions only hold simultaneously for a brief period of time, in the beginning of the precharge phase, while the logic 0 in Req_internal has not yet propagated to D1. The last condition will be true only when point X2 has been driven to 0 during the previous evaluate phase, i.e. only when the previous primary inputs are all at 0. Therefore, this dummy network increases the power consumption at the beginning of the precharge phase of experiment A. The situation with network N2 is very similar. It only consumes power while D1=1, D2=0 and X1=0, that is, for a brief period of time in the evaluate phase of experiments B, C, D.

Figure 5 shows the results of simulation experiments A and B for the structure of Figure 4. Comparing the evaluate phase signatures of the two cases, we note that in exp. A the two previously distinguishable secondary spikes are now almost merged, while in exp. B the previously missing spike has appeared. The highest peak of exp. B is around 23% higher than its exp. A counterpart. This is a significant improvement compared to the situation of Figure 3, where there was no clear spike at all in exp. B. Besides, the percentage is probably not very meaningful, given the comparatively small heights of both spikes. Focusing on the precharge phase signatures, we observe the emergence of two sharp and tall spikes in both experiments. This is a result of the introduced fast inverters, and it is a data independent phenomenon. It can also be noticed that the disparity between the power spikes pointed in Figure 3 has been brought down to around 23%.

In summary, the picture of Figure 5 is more secure than that of Figure 3, both because of the weakening of disparities and because of the emergence of a new spike (in the "precharge" phase) that attracts attention from the disparities. The energy consumption is still acceptably balanced,

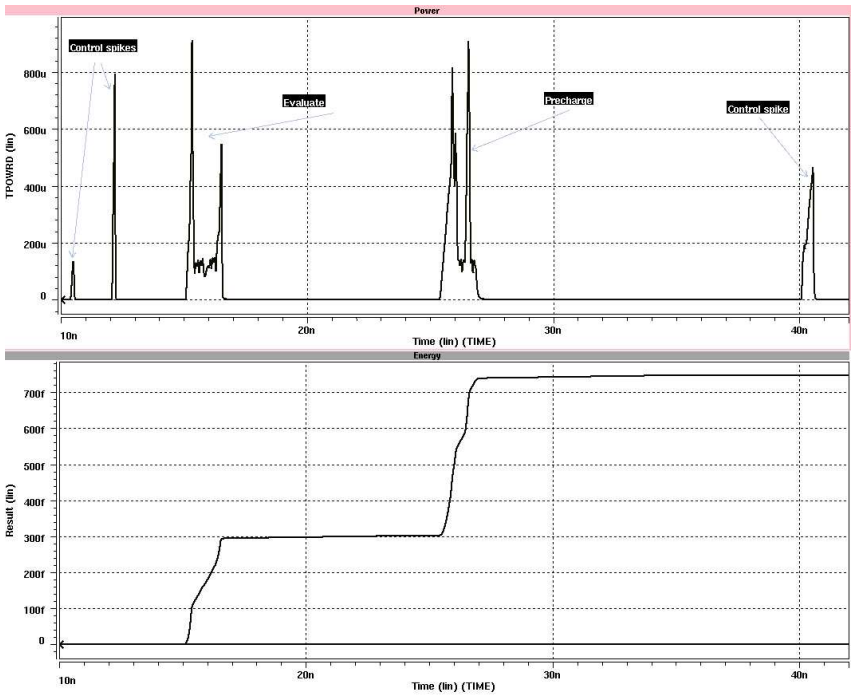


Figure 2. Power and energy profiles of a simple PLA cell

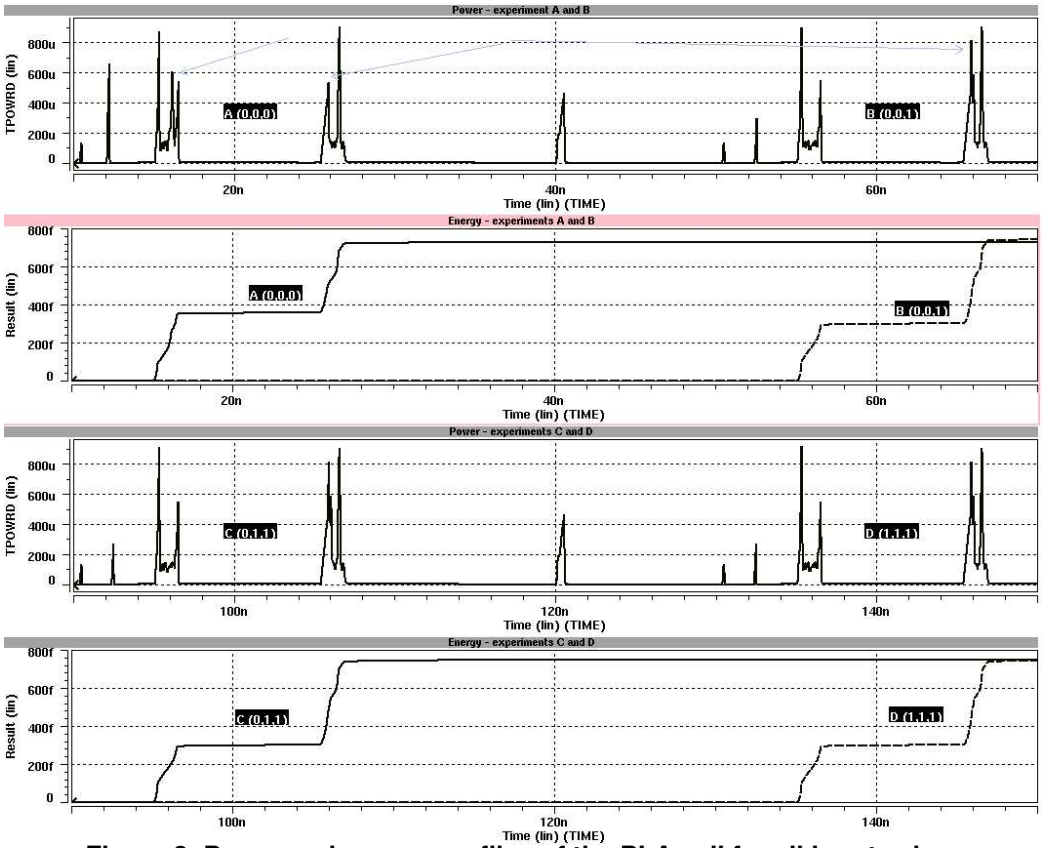


Figure 3. Power and energy profiles of the PLA cell for all input values

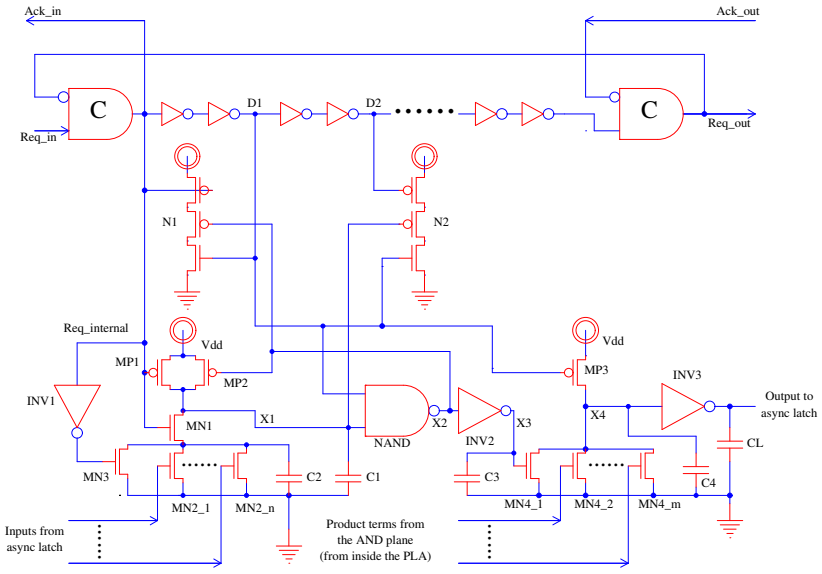


Figure 4. The PLA cell with "dummy" power consuming elements

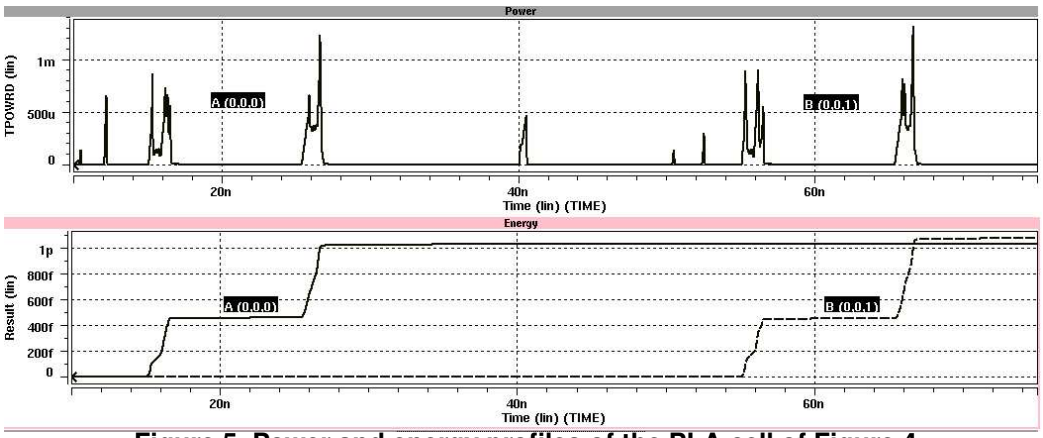


Figure 5. Power and energy profiles of the PLA cell of Figure 4

with a variation of around 4.9% (bottom graph of Figure 5). The transistor count of the cell has increased by 6. In the PLA layout, this will only affect the width of the interplane buffers. Finally, the overall increase in the energy consumption of the system is measured (bottom graphs of Figures 3 and 5) at 34.9%. This is a very encouraging observation, considering that the usual method of balancing power (dual-rail [3, 6]) is at times reported to require an increase of over 100%.

3. System Design from PLAs

Figure 6 depicts how an asynchronous data-processing stage can be configured using the PLA structure presented in the previous section. Clearly the configuration follows the standard single-rail asynchronous data path prototype [7]. The normally n-output PLA implements n+1 functions, the additional one being the *parity predictor*, maintaining even or odd parity on the output bus at all times. This provides a degree of defence against fault injection attacks; the problem at this point is the design of secure single-rail latches, as well as a secure solution for the parity checker. These are the tasks of the next two subsections.

3.1. Secure single-rail latches

Figure 7a illustrates the standard Non Return To Zero (NRTZ) asynchronous transparent latch design. This design is particularly vulnerable to side-channel attacks; indeed, although the first logic level (AND gates) always returns to zero when Req_out=0, whether or not the second level (NOR gates) will experience switching activity is highly data-dependent. The simple modification of Fig-

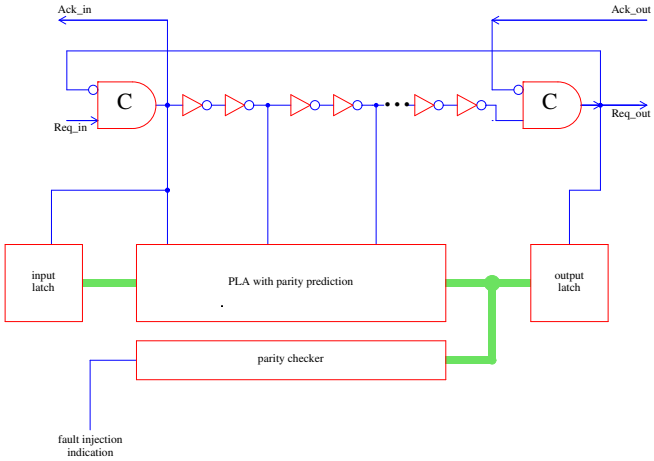


Figure 6. The overall programmable logic asynchronous configuration

ure 7b produces a Return To Zero (RTZ) latch that exhibits data-independent consumption. Indeed, as soon as the latch does not need to hold valid data (i.e. as soon as Req_out=Ack_out=0), the NOR gates are reset to 0. Therefore, exactly one of them will switch when Req_out assumes 1 again. The design of Figure 7b can be used in any single-rail data path (for example in the latches feeding the PLA in Figure 6). It cannot, however, be used for the latches that are being fed by the PLA. The reason is that when a latch is fed by a dynamic CMOS PLA as shown in this paper, it should only be able to register data during the evaluate phase of the PLA. In the nomenclature of Figure 1, the latch should only be allowed to register when *both* Req_internal and Req_out are 1. The modification of Figure 7c implements this, while maintaining data-independant consumption.

Notice that all latch designs receive complementary inputs. This does not mean a dual-rail data path; indeed, an inversion of the single-rail input is enough for the latches to be utilisable. In the particular context of this work, the PLA by nature produces both the functional output and its inverse; these are the output and input of inverter INV3 of Figure 4 respectively. Clearly this does not mean that the PLA is dual-rail.

3.2. Parity checking

Parity checking in the context of self-checking design is typically done using two XOR trees, thus satisfying the robust self-checking theory [4]. Our point of view in this work is considerably different: we do not expect and do not target faults that develop permanently on our system. We are only concerned with maliciously injected, short-lived transients. In this context, it is acceptable to use a single-output checker in the circuit of Figure 6. It is further sensible to use a small, single output, balanced power consumption PLA as designed in Section 2 to provide the parity checking functionality, particularly given that there is no obvious way to balance the power consumption of XOR trees. This way, the configuration of Figure 6 in a sense resembles a *Whirlpool PLA* [2], although the last two stages are minimal in size and are used purely for parity checking.

Overall, with the addition of secure latches and parity checkers, the architecture of Figure 6 provides a PLA-based secure asynchronous data path stage, providing a constant indication of the health of the chip.

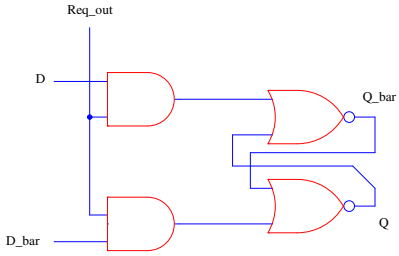
4. Case Study

This section presents power simulations on a small example PLA designed according to Figure 6, and thus demonstrates the power balancing of Section 2 on a somewhat bigger scale design. We implemented the following three logic functions on the PLA:

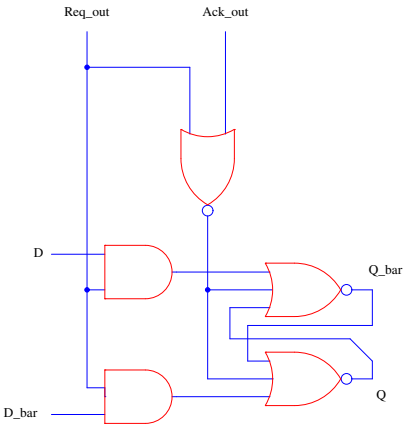
$$\begin{aligned} s &= \overline{a}\overline{b}c + \overline{a}b\overline{c} + a\overline{b}\overline{c} + abc \\ d &= ab + ac + bc \\ p &= \overline{a}c + b\overline{c} + a\overline{b} \end{aligned}$$

Function p is effectively the even parity prediction function of s and d. That is, the overall 3-bit output vector of the PLA will always maintain even parity. Thus, any fault injection attempt corrupting any one - or all three - of the PLA output lines will be detectable at the PLA output, because it will reverse the overall parity. Overall, the considered small PLA has three symmetrical inputs, ten product terms and three outputs ($3 \times 10 \times 3$). Three secure latches (Figure 7b) feed the PLA, while two special latches as of Figure 7c store the useful PLA outputs s and d. In addition, all of s, d, p are fed to the secondary parity-checking PLA that implements the parity checking function:

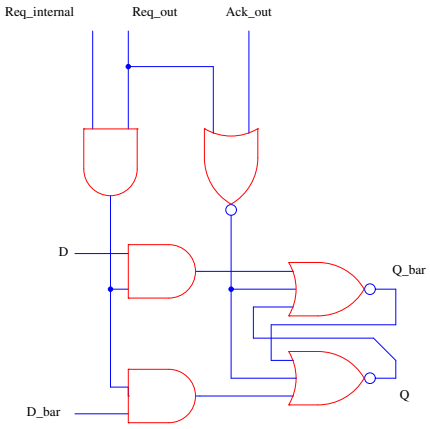
$$check = \overline{s}\overline{d}p + \overline{s}d\overline{p} + s\overline{d}\overline{p} + sdp$$



(a) Basic single-rail NRTZ latch



(b) Secure single-rail RTZ latch



(c) Secure single-rail RTZ latch fed by a PLA

Figure 7. Standard and secure latches

Function *check* produces a 1 to signify fault detection and a 0 to confirm fault-free operation. The parity checker is therefore a $3 \times 4 \times 1$ PLA.

We simulate for four different input vectors, namely *abc* = {111,011,001,000}. Since the inputs are symmetrical, these combinations are enough to depict the power consumption variations of the PLA. The power and energy graphs are shown in Figure 8.

A simple inspection of the waveforms reveals that the power traces for the evaluate and precharge phases in all four cases assume very similar forms, with the same number of spikes at the same time points. This is a very desirable situation, since it makes it harder for an attacker to guess the input pattern. As regards the heights of the power spikes, in the evaluate phase input vectors 000, 100, 110, 111 respectively demonstrate highest spikes at 3.53, 3.73, 3.74 and 3.97mW, corresponding to a maximum variation of 12.5% which is not disappointing. In the subsequent precharge phase, the respective values are 5.70, 6.25, 5.72, 5.99mW, thus experiencing a maximum variation of 9.6%. Notice that in the precharge phase the values are *not* monotonically increasing. This factor is very favourable for defence purposes. As regards the overall energy consumption, the variation between experiments is as low as 3.8%. Finally, the design exhibits strictly data independent timing, since timing only depends on the delay line. This is another source of immunity against side-channel attacks.

The simulation results of this section confirm the security potential of asynchronous programmable logic. In the synchronous domain, programmable logic takes more area than random gate-based design. While this will probably also be true in the asynchronous world, in the context of security applications this area penalty can be compensated for through area savings due to using single, as opposed to dual-rail [3]. Using single-rail also gives significant power consumption savings, as confirmed in section 2. Furthermore, good timing characteristics and easy timing closure have always been advantages of PLA-based logic design, and clearly these are inherent properties of asynchronous PLA systems as well. Finally, it has to be noted that the power balancing performed in this paper clearly refers to nominal values of device characteristics. Fabrication variations or temperature changes may change the power consumption behaviour. However, sensibly assuming that such variations will be random, this will introduce a degree of randomness which is always favourable for security.

5. Conclusion

In this paper, we have presented an asynchronous PLA prototype. It enables the asynchronous designer to enjoy the benefits of regularity, predictability and easy timing closure associated with PLAs. Further, we have demonstrated

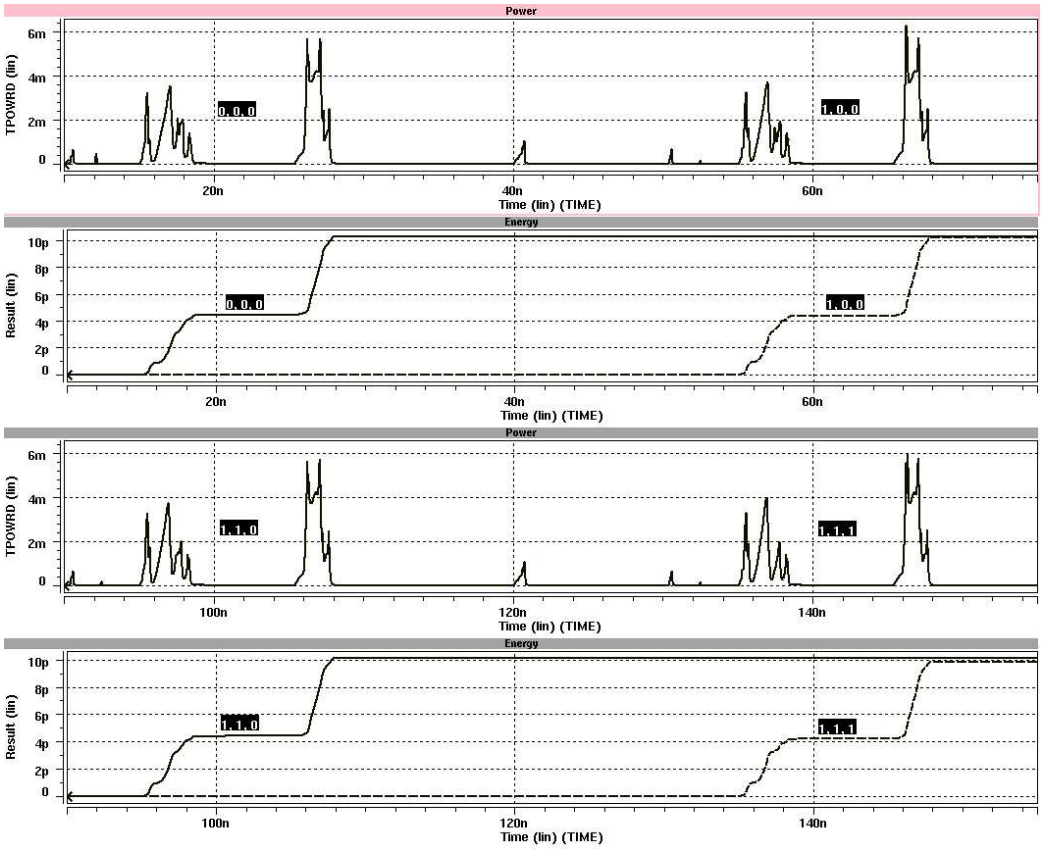


Figure 8. Power simulation results

the potential of using dynamic CMOS PLAs in security applications. Finally, we have given a full configuration from which side-channel attack defiant and fault-detecting asynchronous logic can be built.

References

[1] S. Hauck, S. Burns, G. Borriello, and C. Eberling. An fpga for implementing asynchronous circuits. *IEEE Design & Test of Computers*, 11(3):60–69, Fall 1994.

[2] F. Mo and R. K. Brayton. Whirlpool plas: a regular logic structure and their synthesis. In *Proc. IEEE/ACM International Conference on Computer-Aided Design*, pages 543–550, 2002.

[3] S. Moore, R. Andersson, P. Cunningham, R. Mullins, and G. Taylor. Improving smart card security using self-timed circuits. In *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 211–218, 2002.

[4] M. Nicolaidis and Y. Zorian. On-line testing for vlsi – a compendium of approaches. *Journal of Electronic Testing – A Compendium of Approaches*, 12(1/2):7–20, February/April 1998.

[5] S. Posluszny, N. Aoki, D. Boerstler, P. Coulman, S. Dhong, B. Flachs, P. Hofstee, N. Kojima, O. Kwon, K. Lee, D. Meltzer, K. Nowka, J. Park, J. Peter, J. Silberman, O. Takahashi, and P. Villarrubia. "timing closure by design", a high frequency microprocessor design methodology. In *Proc. Design Automation Conference*, pages 712–717, 2000.

[6] D. Sokolov, J. Murphy, A. Bystrov, and A. Yakovlev. Improving the security of dual-rail circuits. Technical Report NCL-EECE-MSD-TR-2004-101, University of Newcastle-upon-Tyne, April 2004.

[7] J. Sparsø and S. Furber. *Principles of Asynchronous Circuit Design – A Systems Perspective*. Kluwer, 2001.

[8] J. Teifel and R. Manohar. Programmable asynchronous pipeline arrays. In *Proc. International Conference on Field Programmable Logic and Applications*, pages 345–354, 2003.

[9] J. Teifel and R. Manohar. Highly pipelined asynchronous fpgas. In *Proc. ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pages 133–142, 2004.

[10] C. C. Wang, C. F. Wu, R. T. Hwang, and C. H. Kao. A low-power and high-speed dynamic pla circuit configuration for single-clock cmos. *IEEE Transactions on Circuits and Systems I*, 46(7):857–861, July 1999.

[11] J. S. Wang, C. R. Chang, and C. Yeh. Analysis and design of high-speed and low-power cmos plas. *IEEE Journal of Solid-State Circuits*, 36(8):1250–1262, August 2001.

[12] N. H. E. Weste and K. Eshraghian. *Principles of CMOS VLSI Design – A Systems Perspective*. Addison-Wesley, 2nd edition, 1993.