System Clock and Power Supply Cross-Checking for Glitch Detection

Pei Luo, Chao Luo, and Yunsi Fei

Department of Electrical and Computer Engineering
Northeastern University, Boston, MA 02115

Abstract. Cryptographic systems are vulnerable to different kinds of fault injection attacks. System clock glitch is one of the most widely used fault injection methods used in different attacks. In this paper, we propose a method to detect glitches in system clock to fight against clock glitch based fault attacks. We implement the proposed scheme in Virtex-5 FPGA and inject clock glitches into FPGA, results show that the proposed scheme can be easily implemented in both ASICs and FPGAs with very small overhead. Detection results show that the proposed scheme can detect very high frequency clock glitches with very high detection rate.

Keywords: Fault injection, Clock glitch, Hardware security

1 Introduction

Cryptographic systems are vulnerable to different kinds of fault injection attacks. Faults in crypto system may cause errors in the output and the faulty output can be used by attackers to recover the secret key information. Previous works on Differential Fault Analysis (DFA) of different crypto algorithms [1,2,3] show that attackers can inject faults into crypto systems to recover the whole key using only several pairs of correct and faulty ciphertexts. Another attack method, Fault Sensitivity Analysis (FSA), shows that attackers even do not need the ciphertext for attacks, only the correctness of the computation result is enough for attacks [4,5,6,7]. Thus different kinds of fault injection attacks are serious threat to the security of cryptographic systems.

To protect cryptographic systems against different kinds of fault injection attacks, some protection methods have been proposed to detect faulty results. Countermeasures based on faulty result detection make use of techniques such as error detection codes to detect the errors in the output [8,9]. Such methods can detect errors in the output and avoid the attackers to make use of faulty ciphertext for analysis, but they are not resistant to FSA attacks which only require the information that whether the output is correct [5]. Thus, other methods to detect disturbance used for fault injection has been devised [10,11,12,13]. Commonly used system disturbance for fault injection are system clock glitch [14,15], power supply disturbance [16], and electromagnetic (EM) [17], etc.. In this paper, we focus on the detection of glitches in system clock.

In this paper, we devise a method to detect clock glitches using ring oscillator (RO) in FPGA, and RO and voltage-controlled oscillator (VCO) in ASICs. We implement Advanced Encryption Standard (AES) protected with the proposed scheme in a Virtex-5 FPGA, we inject glitches into the system clock and measure the detection rate of the proposed scheme. Results show that the proposed scheme can detect very high frequency clock glitches with a high detection rate. Meanwhile, it can be easily implemented in both FPGAs and ASICs with very small resource overhead.

The rest of this paper is organized as follows. Preliminaries of fault injection using clock glitches are given in Section 2, followed by introduction of previous works. In Section 3, we introduce the proposed method used to detect clock glitch, and analysis about the proposed scheme will be given. In Section 4, we present the clock glitch detection results of the proposed scheme, and the implementation overhead results. Finally, we conclude this paper in Section 5.

2 Preliminaries

2.1 Fault Injection Using Clock Glitch

Clock glitches are widely used to inject faults into crypto hardware systems. In crypto systems, each round of the crypto algorithm needs some amount of time to generate the stable results, and we denote this time as $\mathsf{t_{setup}}$. If the clock cycle is smaller than $\mathsf{t_{setup}}$, the computation result will be passed to next round before it's stable, thus errors will happen in next round input. Shown in Figure 1, attackers can inject glitch with cycle $\mathsf{\Delta_t}$ ($\mathsf{\Delta_t} < \mathsf{t_{setup}}$) into system clock to generate incorrect computation results.

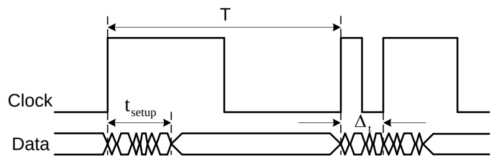

Fig. 1. Fault injection using clock glitch

Comparing with other fault injection methods, fault injection based on clock glitch is easy to implement and it's easy to control the width of glitch Δ_t. Thus besides DFA attacks [15,18], clock glitch based fault injection has also been widely used in FSA attacks which require precise control of the strength (width) of the disturbance (clock glitch) [4,6,7].

2.2 Related Works

In this paper, we focus on the detection of clock glitches. In [11,12], the authors propose to insert a delay buffer chain into an ASIC. The delay chain consists of a series of buffers followed by an inverter, and is inserted between two registers, such that the source register receives the complement of its current value every clock cycle. The latency of this delay buffer chain should be a little greater than the delay of the crypto module, such that if the clock cycle is smaller than the critical path delay, the complement relationship will be violated and thus the clock glitch will be detected.

In [10], a time-to-delay converter is implemented in a 45 nm microprocessor to monitor the system variations. Its fast time-to-digital converter enables quick characterization of transients and therefore the method can detect even nanosecond-scale glitches. Previous papers also find that the frequency of ring oscillators change with the supply power voltage, and this has been used to find the inserted resistors used for power acquisition [19]. However, this property has not been discussed for clock glitch detection.

Not much work has done on the detection of clock glitch in crypto systems. Meanwhile, as FPGAs have been widely used in crypto applications, protection methods which require analog modules are not suitable for FPGA devices. Good protection scheme should make use of existing resource in FPGAs, and should be easy to implement in both FPGAs and ASICs. In this paper, we design efficient clock glitch detection module which can be easily implemented in both FPGAs and ASICs.

3 Clock Glitch and Power Disturbance Detection

In this paper, we propose to monitor the system clock signal clock using a higher frequency clock signal clk. We propose to generate this high frequency signal clk inside cryptograohic systems using availbale resources in FPGA and ACIS. In FPGA, we generate clk using RO which requires only inverters and buffers, and we propose to generate clk using either RO or VCO in ASIC. We will present the glitch detection method assuming high frequency signal clk is available in this section, detailed implementation methods will be given in next section.

3.1 System Clock Glitch Checking

We assume that the generated clock signal clk has higher frequency than clock, and we propose to measure the width of clock using clk for cross-checking.

Firstly, we consider system clock signal with unequal high and low pulse, which means that the duty cycle is not 50%. Correspondingly, we have two separate width counters for the high pulse and low pulse, respectively. For two continuous cycles, the counter results should be the same. The width results of first cycle (n_0^L, n_0^H) and result in the second cycle (n_1^L, n_1^H) are shown in Fig. 2.

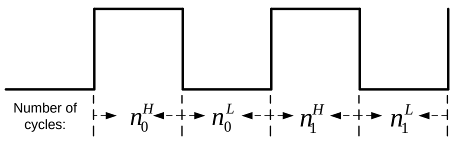

Fig. 2. Clock width count results

If clock and clk are both glitch free, the width of the previous pulse should be equal with the current pulse, which means:

$$\begin{cases} n_0^L = n_1^L \\ n_0^H = n_1^H \end{cases}. \tag{1}$$

If the clock is not constant, the width of logic '0' or logic '1' of clock is varying. Such difference between two consecutive cycles can be used to detect glitches in the system clock. If a glitch happens in system clock clock, the width of clock in terms of number of clk will change suddenly, and then (1) will not hold. We assume that there is a glitch in system clock as Fig. 3.

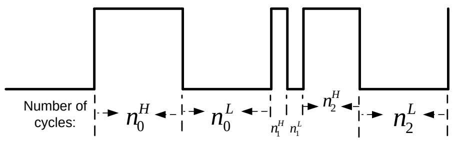

Fig. 3. Clock width count results when glitch injected

Then the width results will be different and the glitch will be detected by the comparators and the Alarm signal is triggered:

$$\begin{cases} n_0^L \neq n_1^L,\ n_1^L \neq n_2^L \\ n_0^H \neq n_1^H,\ n_1^H \neq n_2^H \end{cases}, \tag{2}$$

By comparing the measured width values continuously, glitches in clock signal should be detected. We note here that for clock with 50% duty cycle, n_0^L, n_0^H, n_1^L and n_1^H should be all equal with each other, and they can be compared with each other for system monitoring. Based on the previous discussions, we propose a detection module with the structure shown in Figure 4.

The proposed structure has two width counters, one is used to measure the width of the low pulse while the other one is used to measure the width of the high pulse. The width counter result of current cycle is compared with previous cycle for glitch detection. Alarm will be triggered if glitch in clock is detected, and this will stop the output of faulty cryptographic results.

In the above discussions, we assume that the frequency of generated clock signal clk is higher than the frequency of clock, and use clk to measure the width of clock. Actually, in some crypto systems, clock can have very high frequency, and in such situation, we can use ROs and DCMs to generate clk with much lower frequency and use clock to measure the width of clk for system monitoring.

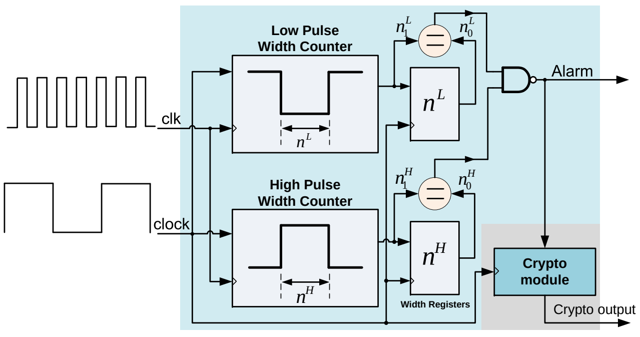

Fig. 4. Clock width monitoring system

3.2 Relaxation of the Design for Robustness

In previous section, we assume that both clk and clock are stable if no glitch exists in system clock. Actually, electronic parts are sensitive to environment variations like temperature. The output frequency of RO, VCO also changes with environmental variations, because the buffer delay is affected by temperature significantly. This is also a problem for the previously proposed scheme [12]. Meanwhile, the power supply may have small inherent variations due to noise or temperature variations, and system clock may have small jitters. Such variations should be differentiated from intentionally inserted glitches. In this section, we present methods to improve robustness of the proposed system to cancel the effect of such variations.

As environmental factors like temperature change slowly, the output frequency variations of VCO and RO caused by the environmental factors are very slow. While glitches in clock always change the frequency of clock abruptly, we can set a comparison threshold to improve the robustness of the proposed scheme. Previous works show that although ring oscillator output frequency changes with the temperature, the change is only several megahertz [20] when temperature changes 100 °C, which is very small comparing with the changes caused by clock glitches. The frequency change of clk caused by environmental variations is much smaller than the change caused by glitches.

Assume that clock or clk have slight variations and thus the width of clock is not constant, we set the comparator to tolerant difference between n_0^L and n_1^L (also the difference between n_0^H and n_1^H) below λ, i.e., variation that is not caused by glitches in clock or power supply is acceptable:

$$\begin{cases} |n_0^L - n_1^L| < \lambda \\ |n_0^H - n_1^H| < \lambda \end{cases}. \tag{3}$$

In our design, the width of current clock cycle is compared with previous clock cycle instead of a constant value, slow changes in the system caused by environmental variations will not trigger the alarm. Thus environmental variations have little effect on the proposed scheme.

3.3 Upper bound of the Propsoed Scheme

For clock glitch detection module, the most important thing is the detection range of glitches. Good detection scheme should have high detection rate for even high frequency injected glitches. We denote the frequency of clk as f_{clk}, and the frequency of glitch as f_g, then the proposed scheme can detect all injected glitches with f_g no higher than $\frac{1}{2}f_{clk}$.

In Figure 5(a), we show that if $f_g > \frac{1}{2}f_{clk}$, the high pulse (and low pulse) of clock glitch may be between two consecutive rising edges of clk, and then the glitch will not be detected in this situation. Meanwhile, in Figure 5(b), $f_g < \frac{1}{2}f_{clk}$, which means the cycle of clock glitch should be greater than two cycles of clk. Then the high pulse (low pulse) of the glitch will always be detected by at least one cycle of clk.

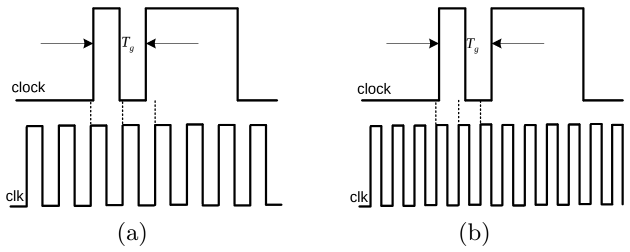

Fig. 5. Upper bound of the detection

Table 1. FPGA resource consumption comparison

	FPGA			ASIC		
	Slice Regs	Slice LUTs	LUT FFs	Area (um^2)	Power (mW)	Timing (ns)
Unprotected	748	2,206	2,481	27072.04	38.39	1.184
One RO	807	2,362	2,594	28325.07	38.76	1.179
Two ROs	867	2,458	2,747	29135.08	38.65	1.389
Three ROs	910	2,560	2,822	30202.27	38.61	1.395

4 Glitch Detection and Implementation Results

We implement unprotected AES and AES protected with the proposed scheme on a Virtex-5 LX30 FPGA, on a SASEBO-GII board. We use a Rigol DG4162 arbitrary waveform generator as system clock source to generate high frequency glitch signal. In this section, we present the detection results of injected clock glitches, and also the resource overhead of the proposed scheme.

4.1 Internal Clock Signal Generation

Different methods can be used to generate clocks in FPGA and ASIC implementations. For FPGA implementations, RO can be used to generate clock signals with different frequency. RO has been widely used for physical uncolonable function (PUF) and random number generator design in FPGA. In ASIC, VCO can be used to generate stable clock signals besides RO.

First of all, we measure the critical path delay of the AES module in Virtex-5 implementation. Results show that the critical path delay is about 129 MHz, which means that errors start to happen when the frequency of glitch is higher than 129 MHz. Meanwhile, all ciphertext bytes will be faulty if the clock cycle is higher than about 144.5 MHz when we inject clock glitch in the last round. Thus clock glitch used by attacker should be between this range (129 MHz to 144.5 MHz). According to the conclusion in Section 3.3, `clk` with frequency higher than 289 MHz will be able to detect all the injected glitches.

We implement five ROs in a Virtex-5 FPGA, they have three, five, seven, nine and eleven stages respectively. With default synthesis and mapping constraints, their frequency are 400 MHz, 305 MHz, 276 MHz, 167 MHz and 108.5 MHz respectively. So the first two ROs can be used for system clock glitch detection and they should be able to detect all the injected glitches for DFA and FSA attacks.

Meanwhile, Digital Clock Manager (DCM) and phase-locked loop (PLL) modules in FPGA can be used to generate stable signals with different frequency using existing clock signals. Thus RO and DCM (or PLL) can be used together to generate clock signals with different frequency. For example, DCM module can generate up to 450 MHz clock in Virtex-5 FPGAs. Thus ROs which have frequency not high enough can be used together with DCM and PLL in FPGA to generate high frequency clock signal to improve the glitch detection coverage.

4.2 Clock Glitch Detection Result

In the experiment, we use Rigol DG4162 to generate 18 MHz to 50 MHz clock signal, and use this signal to generate 0.5 time frequency clock for AES computation (9 MHz to 25 MHz). Meanwhile, we use a DCM module in the Spartan-3A FPGA on SASEBO-GII board to generate 5 times frequency glitch signal (90 MHz to 250 MHz). For each clock frequency, we run glitch injection at the 9^{th} round 10,000 times, and summarize the clock glitch detection rate as in Fig. 6.

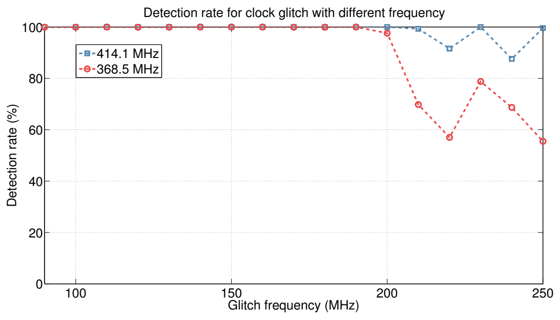

Fig. 6. Clock glitch detection rate at different glitch frequency ($\lambda = 2$)

Fig. 6 shows the clock glitch detection rate for two different ROs, RO_1 (414.1 MHz) and RO_2 (368.5 MHz). It shows that:

- RO_1 can detect up to about 207 MHz glitch with almost 100% rate, while RO_2 can detect up to about 185 MHz glitch frequency with 100% rate. This is the same as the conclusion in Section 3.3.
- Higher frequency clk has overall higher detection rate for injected clock glitches. For example, RO_1 has higher error detection than RO_2 based scheme.

To improve the clock glitch detection rate, we propose to combine the use of more than one detection modules, each with independent clk signal. Thus the system will miss the clock glitch only if all detection modules miss the clock glitch. Assume for one detection module, the detection rate is p, then for n detection module, the detection rate should be $1-(1-p)^n$ instead, which is much higher than p. Which means that the combination of multiple detection module will significantly improve the detection rate of the proposed scheme. We combine two detection modules for glitch detection and present the results in Fig. 7.

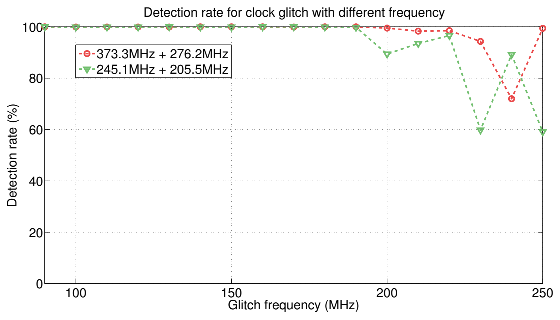

Fig. 7. Clock glitch detection rate with two ROs ($\lambda = 2$)

Fig. 7 shows the detection rate for two designs each with two detection modules, working with two different ROs respectively. Results show that the detection rates for these two designs improve significantly comparing with the single detection module design in Fig. 6. For example, the design with 373.3 MHz plus 276.2 MHz ROs have much higher detection rate than the design with only 368.5 MHz RO. Thus more detection module can be combined together to further improve the detection rate. We will show that although this scheme can significantly improve the detection rate of clock glitches, the resource overhead will be still very small in Section 4.3.

4.3 Overhead Result of the Proposed Scheme

We summarize the resource overhead results of the proposed scheme in Table 1. Table 1 has four rows, which are for unprotected AES, AES protected with one detection module, two detection modules and three detection modules respectively. For FPGA resource evaluation, we use Xilinx ISE 14.6 with default settings for all the implementations. For integrated circuit resource evaluation, the implementations (with and without protections) are modeled in VHDL and synthesized in Synopsys Design Compiler using a 45-nm standard-cell library (FreePDK45). The power and area overhead of the protection schemes are estimated under typical operation conditions assuming a supply voltage of 1.2V and a temperature of 25 Celsius degree. The resource utilization results for both FPGA and ASIC implementations are shown in Table 1.

Table 1 shows that the proposed scheme has very small resource overhead. For example, for AES protected with one detection module, the design needs 7.9% more slice registers, 7.1% more slice lookup tables (LUTs), and 4.6% more LUT flip-flops (FFs) than the original design. For design with multiple detection modules, the overhead increase slightly. For example, the design with two detection modules needs 15.9%, 11.4% and 10.7% more slice registers, slice LUTs and LUT FFs overhead respectively. For ASIC implementations, the protected implementation with one protection module consumes 4.63% more area resource than the original design, and designs with two and three protection modules need 7.62% and 11.56% more area resource overhead respectively.

Results in Section 4.2 and Section 4.3 show that the proposed scheme has a high glitch detection rate for high frequency clock signals. Meanwhile, the proposed scheme can be eaisy implemented in both FPGA and ASCI with small resource overhead. The proposed scheme has high robustness against environmental variations, and more than one modules can be used together to further improve the clock glitch detection rate.

5 Conclusion

We present a clock glitch detection method in this paper which can be easily implemented in both FPGAs and ASICs with very small resource overhead. Glitch injection experiment results on FPGA show that the proposed scheme can effectively detect high frequency clock glitches. Future work will be countermeasures against other kind of fault injection methods, and detection methods for higher frequency clock glitches.

References

1. E. Biham and A. Shamir, “Differential fault analysis of secret key cryptosystems,” in *Advances in Cryptology – CRYPTO ’97*, 1997, vol. 1294, pp. 513–525.
2. P. Dusart, G. Letourneux, and O. Vivolo, “Differential fault analysis on AES,” in *Applied Cryptography and Network Security*, 2003, vol. 2846, pp. 293–306.
3. L. Hemme, “A differential fault attack against early rounds of (Triple-)DES,” in *Cryptographic Hardware and Embedded Systems - CHES 2004*, 2004, vol. 3156, pp. 254–267.
4. Y. Li, K. Sakiyama, S. Gomisawa, T. Fukunaga, J. Takahashi, and K. Ohta, “Fault sensitivity analysis,” in *Cryptographic Hardware and Embedded Systems, CHES 2010*, 2010, vol. 6225, pp. 320–334.
5. O. Mischke, A. Moradi, and T. Guneysu, “Fault sensitivity analysis meets zero-value attack,” in *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2014 Workshop on*, Sept 2014, pp. 59–67.
6. A. Moradi, O. Mischke, and T. Eisenbarth, “Correlation-enhanced power analysis collision attack,” in *Cryptographic Hardware and Embedded Systems, CHES 2010*, 2010, vol. 6225, pp. 125–139.
7. A. Moradi, O. Mischke, C. Paar, Y. Li, K. Ohta, and K. Sakiyama, “On the power of fault sensitivity analysis and collision side-channel attacks in a combined setting,” in *Cryptographic Hardware and Embedded Systems – CHES 2011*, 2011, vol. 6917, pp. 292–311.
8. X. Guo and R. Karri, “Recomputing with permuted operands: A concurrent error detection approach,” *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 32, no. 10, pp. 1595–1608, Oct 2013.
9. P. Luo, Y. Fei, L. Zhang, and A. Ding, “Side-channel power analysis of different protection schemes against fault attacks on AES,” in *ReConFigurable Computing and FPGAs (ReConFig), 2014 International Conference on*, Dec 2014, pp. 1–6.
10. K. Bowman, C. Tokunaga, J. Tschanz, A. Raychowdhury, M. Khellah, B. Geuskens, S. Lu, P. Aseron, T. Karnik, and V. De, “All-digital circuit-level dynamic variation monitor for silicon debug and adaptive clock control,” *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 58, no. 9, pp. 2017–2025, Sept 2011.
11. T. Gneysu and A. Moradi, “Generic side-channel countermeasures for reconfigurable devices,” in *Cryptographic Hardware and Embedded Systems – CHES 2011*, 2011, vol. 6917, pp. 33–48.
12. N. Selmane, “Security evaluation of application-specific integrated circuits and field programmable gate arrays against setup time violation attacks,” *IET Information Security*, vol. 5, pp. 181–190(9), December 2011.
13. K. M. Zick, M. Srivastav, W. Zhang, and M. French, “Sensing nanosecond-scale voltage attacks and natural transients in FPGAs,” in *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, ser. FPGA ’13, 2013, pp. 101–104.
14. J. Balasch, B. Gierlichs, and I. Verbauwhede, “An in-depth and black-box characterization of the effects of clock glitches on 8-bit MCUs,” in *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2011 Workshop on*, Sept 2011, pp. 105–114.
15. T. Fukunaga and J. Takahashi, “Practical fault attack on a cryptographic LSI with ISO/IEC 18033-3 block ciphers,” in *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2009 Workshop on*, Sept 2009, pp. 84–92.
16. M.-C. Hsueh, T. Tsai, and R. Iyer, “Fault injection techniques and tools,” *Computer*, vol. 30, no. 4, pp. 75–82, Apr 1997.
17. A. Dehbaoui, J.-M. Dutertre, B. Robisson, and A. Tria, “Electromagnetic transient faults injection on a hardware and a software implementations of AES,” in *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2012 Workshop on*, Sept 2012, pp. 7–15.
18. H. Bar-El, H. Choukri, D. Naccache, M. Tunstall, and C. Whelan, “The sorcerer’s apprentice guide to fault attacks,” *Proceedings of the IEEE*, vol. 94, no. 2, pp. 370–382, Feb 2006.
19. A. Masle and W. Luk, “Detecting power attacks on reconfigurable hardware,” in *Field Programmable Logic and Applications (FPL), 2012 22nd International Conference on*, Aug 2012, pp. 14–19.
20. E. I. Boemo and S. López-Buedo, “Thermal monitoring on FPGAs using ring-oscillators,” in *Proceedings of the 7th International Workshop on Field-Programmable Logic and Applications*, ser. FPL ’97, 1997, pp. 69–78.