Fpga Chip Identificaton Generator Using Digital Clock Manager

S.Rexlin Leveena*

*(M.E – VLSI Design, Srinivasan Engineering College, Perambalur,
Email: rexlins188@gmail.com)

ABSTRACT

Physically Unclonable Functions (PUF) are commonly used in applications such as hardware security and property protection. Various PUF implementation techniques are used to translate the chip-specific variations into a unique binary string. It is difficult to maintain the repeatability of FPGA chip ID generation, especially used over a wide range of operating conditions. To overcome this problem, configurable ring oscillator, orthogonal re-initialization scheme to improve repeatability. We propose a Digital Clock Manager (DCM). It is used to generate the digital clock signals.

Keywords - **Field-Programmable Gate Array (FPGA), physically unclonable functions, digital clock manager, configurable ring oscillator.**

I. INTRODUCTION

Field-Programmable Gate Arrays (FPGAs) are a mainstream of hardware implementation platform, and need to be equipped with chip identification capabilities.
Today's FPGAs already contain such features. For example, in Xilinx Vertex devices, a bit-stream can be encrypted using a secret key.
When the bit-stream is downloaded, a hardware decryption core decrypts the bit-stream. The bit-stream only operates correctly if the device was programmed with the same key. This key is stored in RAM and it is not possible to read back the value [1]. Unfortunately, the chip Identifier (ID) used for bit-stream decoding is not available for other applications since this value cannot be read.
Xilinx also provides "Device DNA" in Spartan-3A series FPGAs to protect designs from cloning, unauthorized overbuilding and reverse engineering. This feature is a unique factory set FPGA ID hardwired into the device which can be used to implement designs which only operate with a particular ID.
Chip IDs generated in this way should be unique and repeatable. Uniqueness is required to avoid ID collisions between devices, while repeatability is necessary to ensure that a given device returns the same value every time. We use the term unstable to describe a chip ID with low repeatability.
Ring Oscillators (ROs) are often used to generate PUF IDs. One common method is to use a cell consisting of two or more ROs. Due to transistor delay variations, a random output for cell can be obtained from the difference in period of ROs with the same layout but different locations.
By using configurable ring oscillators and a run-time re-initialization scheme, the near-threshold residue values are eliminated. After threshold process the resulting IDs have very good statistical properties over a wide range of conditions, the reliability of chip ID generation is significantly improved. A power-up initialization and dynamic re-initialization process which selects and stores paths with the largest.

II. BACKGROUND

PUFs have drawn considerable attention from the hardware security research community since they were proposed in 2001 [1], [2]. Various PUF implementation techniques are used. They are given in the following subsections.

1. PUF on ASICs

It used an array of addressable NMOS transistors loaded with a common resistive load. Drain current mismatch caused the voltage across the load to be different for different transistors in the array. By addressing the transistors in the array sequentially, a sequence of voltages was generated and successive values converted to a binary sequence via an auto-zeroing comparator to form an ID [3]. An improved circuit which used cross-coupled logic gates to simultaneously generate amplifies and digitize transistor mismatch. This circuit was able to produce a 128-bit, 96% stable ID using only 1.6 pJ/bit [4].
Helinski et al. proposed another PUF design based on measured equivalent resistance variations in the power distribution system of an IC [5].

2. PUF on FPGAs

FPGA-based PUF implementations can be categorized into the following types.

2.1) Memory-Based PUF:

Guajardo et al. utilized the initialization state of static RAM cells in an FPGA and showed that they had suitable statistical properties for producing an ID [6], [7].

2.2) Logic-Based PUF:

The count variation-dependent glitches on the output of a combinational multiplier to generate unique identification [8]. Anderson used an FPGA's carry chain to implement a PUF [9].

2.3) Arbiter-Based PUF:

Figure. 1 shows an arbiter PUF, comprising two parallel n-stage multiplexer chains feeding a flip-flop. A transition is input to the arbiter it travels through a 0 series of 2-input/2-output switches. Each switch is configured to be either a cross or a straight connection based on its selection bit. The arbiter compares the arrival times of its two inputs and generates a response bit. The path segments are designed to have the same nominal delays but their actual delays differ due to process variation.
The difference between the top and bottom path delays on the segment is denoted by in Figure. 1. The PUF challenges are the selector bits of the switches. The output of the arbiter is a function of the challenge bits and different for different chips.

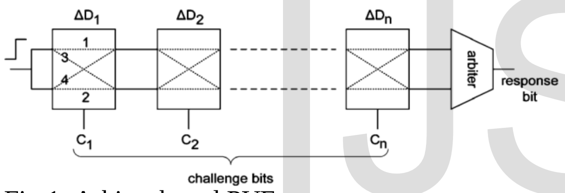

Fig.1. Arbiter-based PUF.

Suh and Devadas [10] generated binary outputs from a difference in path pair delays. Majzoobi et al. [11] proposed an improved arbiter-based PUF which utilized multiple delay lines for each response bit, transformations and combinations of the challenge bits and combination of the outputs from multiple delay lines. This scheme achieved lower predictability and higher resilience against circuit faults then reverse engineering and other security attacks.

2.4) RO-Based PUF:

A RO-based PUF uses differences in period between similar ROs. The RO is typically encapsulated in a hard macro with fixed layout, and arranged in different spatial locations on the FPGA. Since the logic cells and routing are identical, the same nominal value of loop delay is achieved.
Suh and Devadas [10] compared Arbiter and RO based PUFs and found the latter achieved better performance. Ring oscillators with vastly different periods were used to improve the robustness of the generated ID. For each of pairs, the pair with maximum distance was chosen, and a bit vector of these selections is saved so that the same pairs can be used to regenerate the output.

Maiti and Schaumont [12] proposed a configurable ring oscillator to achieve a higher reliability in an RO-based PUF. This scheme used in [10], this approach was more efficient in terms of hardware cost and ROs are required to generate bits.
Merli et al. [13] showed that RO frequencies strongly depend on the surrounding logic. Based on these findings, they proposed a strategy for improving the quality of RO PUF designs by placing and comparing ROs in a chain structure.
Morozov et al. [14] argued that symmetry requirements for Arbiter and Butterfly PUF architectures cannot be satisfied using available FPGA routing schemes despite the apparent routing flexibility of FPGA devices, and suggest that RO-based schemes are preferable.

3. Digital Clock Manager

Digital Clock Manager (DCM) that gives flexible, complete control over the clock frequency. The DCM is simulated in VHDL language using Xilinx Project Navigator software. It does not rely on phase-locked loop (PLL), delay-locked loop (DLL) or any feedback loop, but exhibits most of the functionalities of a DCM like digital frequency synthesis (multiplier/divider), duty-cycle correction, programmable phase shifter, programmable duty-cycle synthesizer and coarse phase shifter.

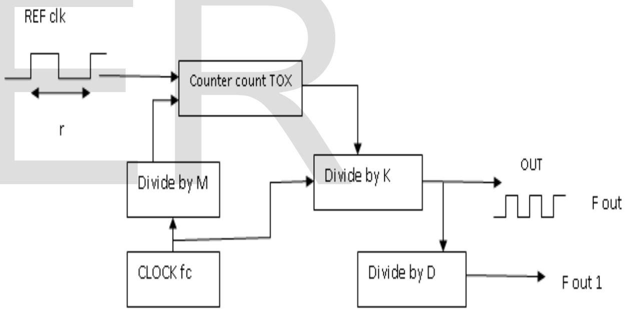

Figure2: Multiplier/divider module

III. PRINCIPLE OF OPERATION

1. One-Bit Generation

Our proposed bit generation is achieved via a 2X2 RO array. The four ROs are placed in a common centered layout, as show in Fig. 3(a). Such an arrangement is called a “cell” in this work and generates a single bit. We adopt an overlapped cell composition rather than the disjoint one used in our previous work.
This serves to improve the resource efficiency of the design by a factor of four. As an example, to generate a 64-bit ID, the new scheme requires a 9X9 RO array compared to 16X16, and the randomness of the generated bits is not compromised.
A timer driven by a 10 MHz system clock, f_{clk} is used to measure the number of rising edges of the RO, N_{RO}, over a period of N_{timer} cycles. The frequency of ring oscillator is given by,

$$N_{RO} = (f_{RO}/f_{clk}) \times N_{timer} \qquad (1)$$

For example, assuming, the value of ranges from 170 to 190 MHz at room temperature. This resulted in an in the range 34 000 to 38 000.If is positive, the bit generated by this cell is 0, otherwise, it is 1. We use the term “polarity” to denote this characteristic.

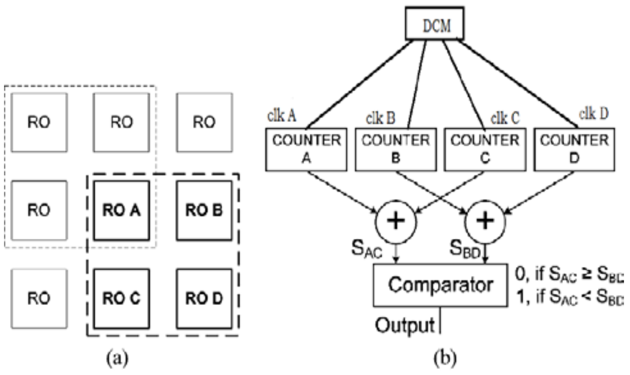

Fig.3. One-bit ID generation. (a) Block diagram of a cell. (b) 1-bit ID computation.

2. Sources of Instability

For the static RO design in the values across all cells were observed to have a Gaussian distribution as shown in Figure 3. This was confirmed by an Anderson Darling test [15]. Since the mean is zero, the most frequently occurring residues are close to this value, making them likely to become unstable. We propose to replace the static RO with a configurable RO to amplify the residue.

3. Digital Clock Generation

The complete DCM module is divided into five sub-modules: digital frequency synthesizer, duty-cycle corrector, programmable phase shifter, programmable duty cycle synthesizer and coarse phase shifter. The digital frequency synthesizer (DFS) performs the function of clock multiplication/division.
The multiplication of a clock frequency by a programmable number is often necessary in high-performance signal synthesizers as well as in FPGAs. Most of the systems utilize PLL or DLL for clock multiplication. These systems result in high phase accuracy with higher programmability but at the expense of high complexity and high power consumption. The output clock frequency of the DFS is given by:

$$f_{DFS} = f_{clk}\,(\text{multiplier/divider}) \qquad (2)$$

The output clock from the DFS is phase symmetric with 50% duty cycle. The duty-cycle correction module corrects the duty cycle of the input clock to 50%. If the input clock has 30% duty cycle, the output-clock duty cycle is corrected to exactly 50% with no change in frequency. The programmable phase shifter (PPS) provides the programmable phase shifted versions of the input clock. It has integer values ranging from –255 to 255 to which phase-shift attribute is assigned. The programmable duty-cycle synthesizer varies the duty cycle of the input clock as per the programmable input. This helps to improve the metastabilty factor due to set-up/hold time violations.
This module assigns 0 to 100% duty cycle to the output clock. The coarse phase shifter provides output clocks that are 90°, 180° and 270° phase-shifted versions of the in-put clock and require no programmability. The master clock used to sample the input reference clock has a frequency of 100MHz and is generated using the ring oscillator at the gate level with delay elements implemented as inverting buffer.
Since DLL or PLL is not the basis of this design, the maximum allowed input reference clock is of 1 MHz. All the sub-modules of the DCM are more or less based on the same basic concept. In VHDL description, an inverter in the ring oscillator is defined as,

$$Q \Leftarrow \text{not(A)after invdel} \qquad (3)$$

Where ‘invdel’ is estimated delay for the inverter. This delay is estimated on the basis of the device used for implementation and the values of delay at the gate and flip-flop level. As per the delay values provided in the datasheet of XILINX Spartan-3 FPGA XC3S1500, seven inverters are placed as delay element. The reset signal is provided to the ring oscillator as well as to all the counters and is an active- high signal.

4. Error analysis in DCM

There are mainly two types of errors to be analyzed in this implementation of the DCM. One of the errors occurs because the period of the reference clock may not be an integral multiple of the master clock period. This error results in jitter. The second error occurs because the accumulated value ‘K’ in the counter may not be exactly divisible by any of the divide-by-‘N’ counters. This error results in phase asymmetry. Both these errors can be minimized by having master clock at very high frequency so that the counter counts up to very large values. But since we cannot employ ring oscillators with very high frequencies (due to the need for the circuit to process the OSC signal typically, a few nanoseconds), this limits the maximum frequency that can be controlled by the DCM. In this case, based on the current technology of FPGA used, the output frequency is below 20 MHz.

IV. IMPLEMENTATION

1. Architecture

Figure 4 illustrates the architecture of our chip ID generator design. It includes a 9 x 9 RO array providing 8x8 cells.
This can generate 64 separate bits. The address generator together with the two decoders select a single RO to

operate over a given time interval. A 4-bit global RO configuration signal, detailed in the next subsection, is also sent to each RO. At any given time only one RO can be activated and hence the configuration only affects the operating RO. Two levels of multiplexors are used to route the output of the selected RO to the counter.

2. Configurable RO

The circuit implementation of the configurable RO is shown in Figure 5. In this work, a Xilinx Spartan-3e was used to demonstrate the technique. The design could be easily ported to different FPGA families. A four-stage RO is used where three of the stages are non-inverting and the final one is inverting. Each occupies two Xilinx logic elements (LEs) within a slice and a multiplexer is used to choose the signal path. The entire RO occupies a single Xilinx configurable logic block (CLB). By selecting different values of $S_0 - S_3$, 16 different configurations can be chosen. Logic and interconnect delay mismatch in the paths of the different configurations change the frequency of the RO.

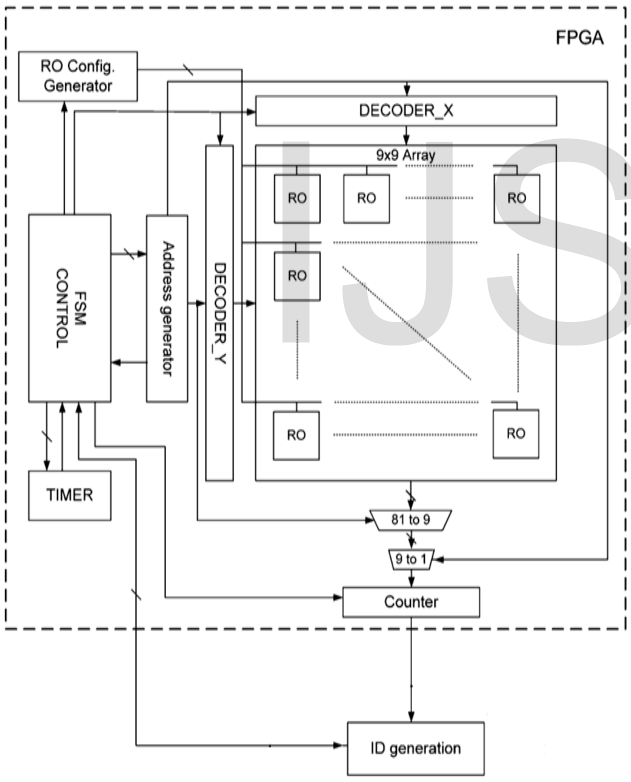

Fig. 4. Block diagram of chip ID generator architecture.

2. Configurable RO

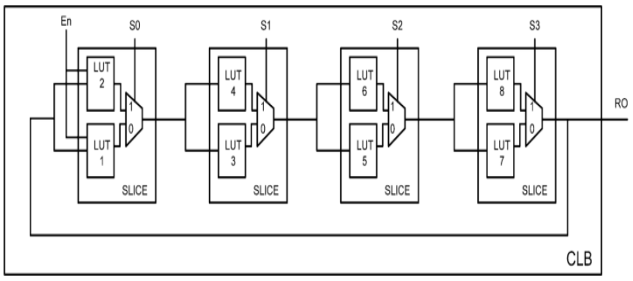

Fig.5. Circuit for the configurable RO.

The circuit implementation of the configurable RO is shown in Figure 5. In this work, a Xilinx Spartan-3e was used to demonstrate the technique. The design could be easily ported to different FPGA families. A four-stage RO is used where three of the stages are non-inverting and the final one is inverting. Each occupies two Xilinx logic elements (LEs) within a slice and a multiplexer is used to choose the signal path. The entire RO occupies a single Xilinx configurable logic block (CLB). By selecting different values of $S_0 - S_3$, 16 different configurations can be chosen. Logic and interconnect delay mismatch in the paths of the different configurations change the frequency of the RO.
Due to these expected systematic variations, generating using different configurations leads to correlated outputs. Instead, we use the same configuration for all four ROs in a cell, and choose the one with the largest. This technique employs one configurable RO to achieve a similar result to choosing from 16 normal ROs as done in Suh and Devadas's work [10], resulting in a reduction in area.

3. Configuration Initialization

3.1) Power-Up Initialization

Our proposed method requires a set of configurations to generate stable IDs so a scheme is required to initialize them upon power-up. One straight forward approach is to determine configurations when the FPGA is powered up the first time, and store them in non-volatile memory or on an authorized server. Such an approach would also need to carefully consider the possibility of information leakage and susceptibility to modeling attacks [20].
When the chip is subsequently powered up, configurations are transferred to the chip for ID generation. Unfortunately, for this scenario, a communication channel is required. Even though the configuration does not reveal relative speeds of the ROs, it leaks information. For instance, if the same configuration is used for overlapping cells, an adversary may be able to infer a dependency between the ROs involved. Moreover, if an adversary can see both the configuration and the resulting ID, this provides additional information to could aid modeling attacks [20].

3.2) Run-Time Re-Initialization

We propose a dynamic re-initialization technique to further improve reliability. Re-initialization could be

triggered by periodically measuring the frequency of a particular RO (or embedded sensor) to track temperature or voltage variations. It serves to find a configuration with larger than the previous one while maintaining polarity if possible. If the current configuration still remains the best one, no modification is made. Otherwise, a new configuration is stored.
An RO counter value can be monitored to detect environmental changes. We studied how the counter value changes with temperature and supply voltage. This threshold can be obviously be changed for different requirements.

4. Flow of Chip ID Generation

After power-up initialization, the overall process of the proposed chip ID generation can be represented as shown in Figure 6. It is divided into two phases, generation and re-initialization. During re-initialization, the configurations of all cells are swept to determine which one generates the largest of the same polarity as the previous best configuration. The information is stored for chip ID generation.

V. RESULTS

1. Summary of Hardware Resource Consumption

We implemented the system on a custom board with a Xilinx Spartan-3e FPGA (xc3s250e-4pq208). Xilinx ISE Design Suite 12.1 and Vision v3.62c are respectively used for FPGA design.

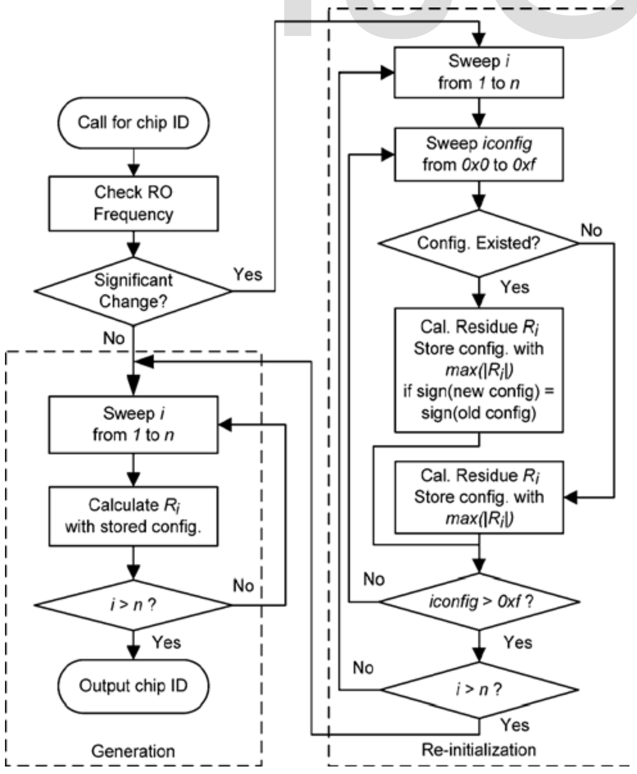

Figure6. Flowchart showing chip ID generation process.

2. Statistical Analysis

2.1) Cell Configurations:

The distribution of selected configurations over all cells. Although the distribution is not uniform, strong biases towards some particular configurations were not evident.

2.2) Hamming Distance:

The Hamming distances between all pairs of chip IDs. The average value is 30, which is 47% of the bit width. This is very close to the ideal of 50% for independent IDs.

2.3) Correlation Analysis:

As each is fully correlated with itself, the diagonal values are equal to 1. On average, the correlation between different s is and 90% of the correlations are in the range 0.2 to 0.2 with a maximum absolute value of 0.44. From this analysis we conclude that there was no evident correlation between bits in the ID generation process.

VI. CONCLUSION

The chip ID generation method using digital clock module, configurable RO, power-up initialization and adaptive re-initialization can considerably improve its repeatability. Results show that a very stable ID generation can be achieved over a wide range of operating conditions. Since this design was completely implemented using standard digital circuits. The DCM is simulated in VHDL language using XILINX Project Navigator software.
As future work, develop more parallel generation schemes to speed up chip ID generation.

REFERENCES

Proceedings Papers:

[1] R. Pappu, “”Physical One-way Functions” Ph.D. dissertation, Program in Media Arts Sci., Sch. Arch. Planning, Massachusetts Inst. Technol., Cambridge, 2001. [Online]. Available: http://pubs.media.mit.edu/pubs/papers/01.03.pappuphd.powf.pdf

[2] R. Pappu, B. Recht, J. Taylor, and N. Gershenfeld, “Physical one-way functions,” Science vol. 297, no. 5589, pp. 2026–2030, 2002. [Online]. Available: http://www.sciencemag.org/content/297/5589/2026.abstract

[3] K. Lofstrom, W. R. Daasch, and D. Taylor, “IC identification circuit using device mismatch,” in Proc. Int. Solid-State Circuits Conf. (ISSCC), 2000, pp. 372–373.

[4] Y. Su, J. Holleman, and B. Otis, “A digital 1.6 pJ/bit chip identification circuit using process

variations," IEEE J. Solid-State Circuits, vol. 43, no. 1, pp. 69–77, Jan. 2008.

[5] [5] R. Helinski, D. Acharyya, and J. Plusquellic, "A physical unclonable function defined using power distribution system equivalent resistance variations," in Proc. 46th Annu. Design Autom. Conf. (DAC), 2009, pp. 676–681.

[6] J. Guajardo, S. S. Kumar, G.-J. Schrijen, and P. Tuyls, "FPGA intrinsic PUFs and their use for IP protection," in Proc. 9th Int. Workshop Cryptograph. Hardw. Embed. Syst. (CHES) , 2007, pp. 63–80.

[7] J. Guajardo, S.Kumar,G.-J. Schrijen, and P. Tuyls, "Physical unclonable functions and public-key crypto for FPGA IP protection," in Proc. Int. Conf. Field Program. Logic Appl. (FPL), 2007, pp. 189–195.

[8] H. Patel, Y. Kim, J. McDonald, and L. Starman, "Increasing stability and distinguishability of the digital fingerprint in FPGAs through input word analysis," in Proc. Int. Conf. Field Program. Logic Appl. (FPL), 2009, pp. 391–396.

[9] J. Anderson, "A PUF design for secure FPGA-based embedded systems," in Proc. 15th Asia South Pacific Design Autom. Conf. (ASPDAC) , 2010, pp. 1–6.

[10] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in Proc. 44th Annu. Design Autom. Conf. (DAC), 2007, pp. 9–14.

[11] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Lightweight secure PUFs," in Proc. IEEE/ACM Int. Conf. Comput.-Aided Design, 2008, pp. 670–673.

[12] A. Maiti and P. Schaumon "Improving the quality of a physical unclonable function using configurable ring oscillators," in Proc. Int. Conf. Field Program. Logic Appl. (FPL), 2009, pp. 703–707.

[13] D. Merli, F. Stumpf, and C. Eckert "Improving the quality of ring oscillator PUFs on FPGAs" in Proc. 5thWorkshop Embed. Syst. Security, 2010, pp. 9:1–9:9.

[14] S. Morozov, A. Maiti and P. Schaumont "An analysis of delay based PUF implementations on FPGA" in Reconfigurable Computing: Architectures Tools and Applications, P. Sirisuk, F. Morgan, T. El-Ghazawi and H. Amano, Eds. Berlin, Germany: Springer, 2010, vol. 5992, pp. 382–387, Lecture Notes in Computer Science.

[15] T. W. Anderson and D. A. Darling "Asymptotic theory of certain goodness of fit criteria based on stochastic processes," Annu. Math. Statist., vol. 23, pp. 193–212, 1952.