A PUF Design for Secure FPGA-Based Embedded Systems

Jason H. Anderson
Department of Electrical and Computer Engineering
University of Toronto
Toronto, Ontario, Canada
e-mail: janders@eecg.toronto.edu

Abstract— The concept of having an integrated circuit (IC) generate its own unique digital signature has broad application in areas such as embedded systems security, and IP/IC counter-piracy. Physically unclonable functions (PUFs) are circuits that compute a unique signature for a given IC based on the process variations inherent in the IC manufacturing process. This paper presents the first PUF design specifically targeted for field-programmable gate arrays (FPGAs). Our novel design makes use of the underlying FPGA architecture, and unlike prior published PUFs, the proposed PUF can be naturally embedded into a design's HDL, consuming very little area, and does not require the use of "hard macros" with fixed routing. Measured results on the Xilinx Virtex-5 65 nm FPGA demonstrate PUF signatures to be both unique and reliable under temperature variation.

I. Introduction

The need to ascribe a unique binary signature to an integrated circuit (IC) has applications in digital design and embedded systems, ranging from digital rights management, IP protection, cryptographic key generation, device authentication, and IC counterfeit detection/prevention. Counterfeit hardware is a major concern that rose to prominence in 2008, when the FBI announced that counterfeit Cisco networking products had unknowingly been purchased and used by the U.S. government [1]. More recently, over the past few months cloned cellphones and the industry that produces them have garnered considerable media attention [2]. In addition to copies of specific products, a legitimate concern for fabless semiconductor companies is that counterfeit versions of their chips can easily be made and sold by malicious individuals in the same fabs to which they outsource their fabrication. To counter such threats, recent work has considered methodologies for ending IC piracy that requires a fabricated chip to generate its own unique signature [3]. A physically unclonable function (PUF) is a new concept in hardware security, and a promising candidate for IC signature generation.

An artifact of state-of-the-art sub-100 nm IC manufacturing is that random variations in doping concentrations, line widths, or other properties cause unpredictable variations in transistor speed and interconnect. Most PUF designs compute unique signatures by exploiting such delay differences. At a high level, the approach taken in PUF design is to incorporate multiple identical copies of a particular combinational path into an IC design. The copies must be perfectly matched from the logic, placement and routing perspectives, and ideally, also from the perspective of their physical surroundings. Since the copies are identical, delay differences between the copies are due to random variations that are inherent to the manufacturing process, and cannot be controlled or cloned. PUF circuitry measures the delay differences between path copies to generate the unique PUF signature. PUF circuits bring a useful purpose to the same variations which have a deleterious effect on other IC metrics, notably timing and yield.

Field-programmable gate arrays (FPGAs) are programmable chips that can be configured by the end-user to implement *any* digital circuit. FPGAs are used in a broad range of computing and embedded applications because of their reconfigurability, improving physical qualities (i.e. speed, area and power), and their steadily decreasing cost. The economics of semiconductor scaling is such that as feature sizes shrink, the costs associated with building a custom ASIC escalate rapidly. This trend has made FPGAs the technology of choice for many applications today, even those with tens to hundreds of thousands of units in volume.

The pervasive use of FPGAs in embedded applications suggests that it is desirable for PUFs to be realized on FPGAs. Aside from the known applications of PUFs, there may exist FPGA-specific uses. For example, since the use of FPGA vendor tools is mandatory, vendor software for FPGA configuration bitstream generation could be made to function correctly only for chips with specific *pre-qualified* signatures corresponding to legitimate (non-counterfeit) chips. It is also worth noting that PUF implementations on FPGAs is one focus of a recently founded start-up company [4].

This paper contributes the first PUF specifically designed for FPGAs. The PUF offers improved ease-of-use over prior designs. FPGAs contain arrays of identical logic and routing circuitry. We leverage this underlying architectural regularity to realize matched copies of combinational paths whose delay differences stem from manufacturing variations. In our design, each individual bit of the PUF signature is computed within a single FPGA logic block. No matched routing external to the logic block is required, easing its incorporation into a design. The PUF is completely described in VHDL and unlike prior PUF designs, it does not necessitate the use of hard macros with fixed routing. The PUF is demonstrated with Xilinx Virtex-5 65 nm FPGAs [5], though it can be easily ported to other FPGA families. The remainder of this paper is organized as follows: Section II describes prior work on PUF circuits. The proposed PUF design is presented in Section III. An experimental evaluation is described in Section IV. Conclusions and suggestions for future work are offered in Section V.

II. Related Work

We first review prior work on PUFs and then provide background on the FPGA architecture relevant to our PUF design.

A. Physically Unclonable Functions

Four main classes of PUF have appeared in recent literature: 1) Ring Oscillator (RO)-based PUFs [6], 2) arbiter PUFs [7, 8, 9, 10], 3) memory-based PUFs [11, 12], and 4) glitch count-based PUFs [13]. Of these, the RO-based and arbiter PUF are the most well-studied. Fig. 1(a) shows a simple RO-based PUF with two oscillators. The oscillators (shown in the dashed boxes) must have an identical layout on the IC, such that any frequency differences between the two are owing strictly to process variations. The two oscillators drive counters, and following a period of oscillation, the counter values are compared, producing a single bit of the PUF signature. RO-based PUFs contain many ROs and the pairs of ROs to compare may either be pre-selected, or a multiplexer may be added just before the comparators to allow user-selection of the ROs to compare. In this way, the PUF can function in a challenge/response paradigm wherein user-supplied input vectors are paired with PUF responses. The challenge input from the user indicates the ROs to compare; the response is the `0`/`1` indication produced by the comparator. The challenge/response notion is stronger from the security perspective than having one unique signature. In challenge/response, a unique signature response is produced for each unique user input challenge.

Fig. 1(b) shows an arbiter PUF, comprising two parallel n-stage multiplexer chains feeding a flip-flop. A step input is applied, along with challenge input bits $X(0:n)$. At stage i of the multiplexer chain, the step input signal is either fed forward along the same chain, or it is interchanged with the opposite chain, as controlled by $X(i)$. Depending on the challenge bits and the delay differences between the top and bottom MUX chains, the step input will either arrive at the flip-flop's D or clock input first, causing either a logic-`1` or a `0` to be latched, respectively. The latched value constitutes one bit of PUF signature. While Fig. 1 illustrates the basic design concepts, numerous improvements have been proposed to strengthen their resilience to attacks [7, 6, 14, 10]. The third type of PUF, a memory-based PUF, uses the random initial state of memory bits on device start-up as the PUF signature [11]. The fourth and final type of PUF counts variability-dependent glitches on the output of a combinational multiplier [13].

The PUF designs described above are not specifically targeted to FPGAs; they can equally be implemented within custom chips. Moreover, several challenges arise when they are implemented on FPGAs. First, both types of PUFs in Fig. 1 require that identical logic and routing be used along certain combinational paths, thereby guaranteeing that delay differences along such paths are due to process variations. In practice, identical implementation of paths requires the use of hard macros, which incorporate fixed placement and routing. The use of hard macros complicates the design flow and requires that a designer work at a lower-level of abstraction than RTL. In fact, it may be necessary to manually route the above PUF circuits to ensure that specific FPGA interconnect resources are used along certain paths. Manual routing is tedious and error-prone. Furthermore, the locked routing in the PUF presents an "obstacle" to other design signals in the routing stage of the flow, potentially increasing design congestion and reducing circuit performance. Finally, the prior PUF designs consume considerable silicon area per PUF bit. In contrast, ease-of-use within the FPGA CAD flow and low silicon area are key benefits offered by the proposed PUF design.

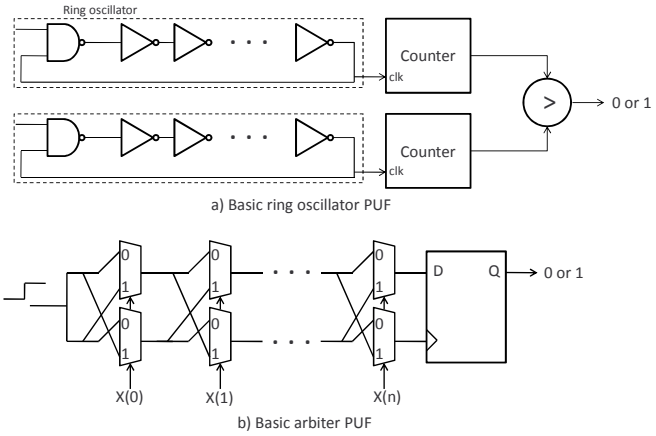

Fig. 1. Previous PUF designs.

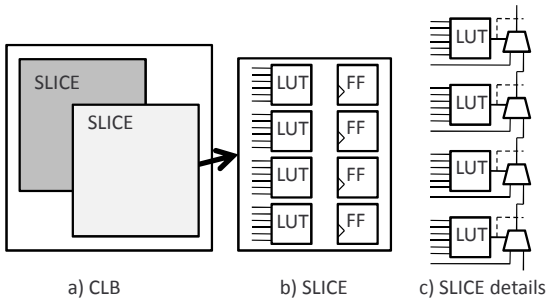

Fig. 2. Virtex-5 logic block architecture.

B. Xilinx Virtex-5 FPGAs

Fig. 2(a) depicts a Virtex-5 logic block, called a Configurable Logic Block (CLB). A CLB comprises two SLICEs, as shown in Fig. 2(b). A SLICE contains four 6-input look-up-tables (LUTs), four flip-flops, and other arithmetic circuitry (not shown in Fig. 2(b)). 6-input LUTs are small memories that are capable of implementing *any* logic function of up to 6 variables. In particular, each LUT contains static RAM cells that hold the truth table of the logic function implemented by the LUT. CLBs are arranged in a two-dimensional array on the FPGA and can be connected to one another through a programmable interconnection matrix.

Fig. 2(c) shows additional SLICE details that are used in our PUF design. Observe that each LUT output connects to the select input of a 2-to-1 multiplexer. Each multiplexer receives one of its data inputs from the multiplexer below it, and the second data input can be received from outside the SLICE. The dashed lines indicate that the LUT outputs also drive other circuitry, not shown in the figure. The vertical chain of multiplexers is called the *carry chain* and it is intended for implement-

ing fast arithmetic operations. We use the carry multiplexers in our PUF design implementation. Note that the carry chain is not unique to Virtex-5; similar structures are present in FPGAs from other vendors, such as Altera's Stratix-III [15].

C. LUTs as Memories and Shift Registers

In most applications, LUTs are used to implement combinational logic functions, in which case the LUT's SRAM cell contents is programmed during device configuration and remains unchanged thereafter. An alternative application of a LUT is to use it as a memory. Each 6-LUT in Virtex-5 contains 64 SRAM cells and can therefore be used as a 64x1 RAM. Moreover, the LUTs in a SLICE can be combined with one another to implement RAMs with different aspect ratios. Besides using LUTs as small RAMs, the Virtex-5 architecture allows the internal SRAM cells within a LUT to be chained together serially, thereby allowing the LUT to function as a shift register.

As we illustrate in the next section, our PUF design configures LUTs as shift registers. 25% of the LUTs in Virtex-5 can be used as memories/shift registers. Such LUTs reside in SLICEM blocks, and are spread at regular intervals throughout the array[1]. SLICEM blocks are variants of SLICE blocks, where the "M" indicates the LUTs can be used as memories. LUTs in Altera's Stratix-III FPGAs can also be used as memories [16].

III. Proposed Design

We propose an FPGA PUF circuit that, based on random process variations, will produce either a logic-`0` or logic-`1`. Multiple instances of our circuit will be instantiated to create a multi-bit PUF signature. Fig. 3 shows the core of our PUF design. Two LUTs, A and B, within a Virtex-5 SLICE are used in 16-bit shift register mode. The shift register contents are pre-initialized as follows:

- LUT A: `0101010101010101 (0x5555)`
- LUT B: `1010101010101010 (0xAAAA)`

Note that LUT A's initialization bitstring is the complement of LUT B's bitstring. The shift register implemented in LUT A will produce the sequence `0101...` and so on. Whereas, the shift register implemented in LUT B will generate the sequence `1010...`. The shift register inputs, IN, are assigned to allow the same sequences to continue beyond the initial 16 cycles. Importantly, the shift register OUT pins drive the select input pins on carry chain multiplexers. Both carry multiplexers have their "`0`" data input tied to logic-`0`. The bottom carry chain multiplexer has its "`1`" data input tied to logic-`1`. The output of the bottom multiplexer drives the "`1`" data input of the top multiplexer.

Consider the dynamic clocked behavior of the circuit in Fig. 3. Initially, the OUT pin of LUT A is at logic-`0`, and therefore signal $N2$ is at logic-`0`. The OUT pin of LUT B is logic-`1`, setting signal $N1$ to be logic-`1`. At the rising clock edge, the OUT pin of LUT A will transition from logic-`0` to logic-`1`, and

[1]Every second CLB in Virtex-5 contains one SLICEM and one regular SLICE.

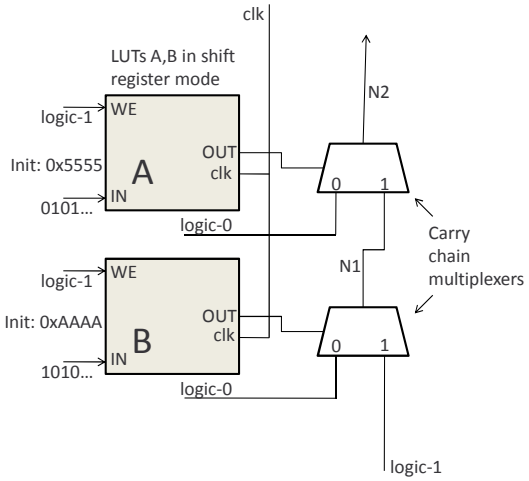

Fig. 3. PUF circuit.

the OUT pin of LUT B will transition from logic-`1` to logic-`0`. Although LUT A and the multiplexer it drives should be identical to LUT B and its multiplexer, the two pieces of circuitry in fact experience different delays due to random process variations. We exploit this property for PUF signature generation.

There are two cases worth highlighting. First, consider the case wherein LUT B and the multiplexer it drives are *faster* than LUT A and its multiplexer. In this case, when LUT B transitions from logic-`1` to logic-`0`, signal $N1$ also transitions from logic-`1` to logic-`0`. Following that, the *slower* LUT A transitions from logic-`0` to logic-`1`, and signal $N2$ is held constant at logic-`0` throughout the process. The second case is the opposite one where LUT A and its multiplexer are the faster ones. In this case, LUT A's OUT pin transitions from logic-`0` to logic-`1` and net $N1$ has not yet transitioned from logic-`1` to logic-`0`. A short positive spike (a glitch) will appear on $N2$ for the period before $N1$ transitions to logic-`0`. The presense or absense of a positive spike on $N2$, and the length of the spike pulse, are due to process variations that impact the relative delays of LUTs A and B and the carry chain multiplexers.

We use the presense/absense of a positive spike on $N2$ to determine a PUF signature bit. $N2$ is connected to the asynchronous preset input of a flip-flop, as shown in Fig. 4. The flip-flop is initialized to logic-`0` and has its output Q fed back to its D input. In the event that a glitch on signal $N2$ reaches the preset, the flip-flop output becomes logic-`1` and the PUF signature bit is logic-`1`. Otherwise, the PUF signature bit is logic-`0`. The flip-flop of Fig. 4 can be located in the same SLICE as the circuitry of Fig. 3 because they can use the same clock signal. The Virtex-5 SLICE has an architectural restriction that only a single clock signal may be used in any given SLICE. We used relative location (RLOC) constraints (a Xilinx physical packing constraint) in our VHDL to locate the flip-flop in the same SLICE as the circuitry of Fig. 3.

Although the delay differences due to process variations may trigger a short pulse on signal $N2$ as shown in Fig. 3, the pulse width may be so short that it is "filtered out" on its way along the routing path to the flip-flop preset input. FPGA routing contains buffered switches and metal wire segments. In essence, the resistive and capacitive loading on the routing path can be

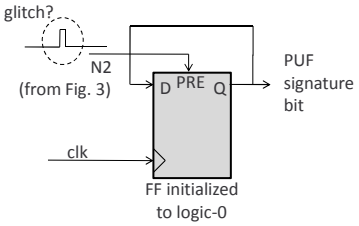

Fig. 4. PUF bit generation.

viewed as a low-pass filter, potentially damping out the high-frequency pulse, thereby causing the PUF bit to be logic-`0` with high probability. In fact, the RC nature of the routing is also impacted by process variations. Conversely, if the pulse is always generated and its width is too wide, it is more likely to reach the flip-flop preset input, making PUF bits logic-`1` with high probability. Hence, it is desirable if the pulse "resolution" can be tuned to maximize the PUF utility.

With the goal of managing pulse width in mind, we note that the position of LUT B in the SLICE can be tailored to create a meaningful PUF. As mentioned previously, each Virtex-5 SLICE contains four LUTs, shown in Fig 5. LUT A will reside in LUT slot position 0. LUT B can be placed in any of slots 1, 2 or 3 to modulate the pulse width. Increasing the slot location number of LUT B will tend to increase pulse width, as transitions from B's output will take longer to propagate to the top-most MUX in the carry chain. Furthermore, the bottom-most carry chain multiplexer in the SLICE can be driven by the top-most multiplexer in the SLICE immediately below. The purpose of this connectivity is to enable the creation of longer carry chains (i.e. longer than 4 bits). The carry connectivity between SLICEs permits longer pulse widths to be produced within the proposed PUF, with the PUF circuit spanning multiple SLICEs arranged vertically in the placement.

For our experimental study, we found that the configuration shown in Fig. 6 produced the best results in terms of achieving a roughly equal balance of PUF bits being logic-`0` and logic-`1`. LUT A is in the top-most position of a first SLICE, and LUT B is placed in the third position in a SLICE immediately below. Note that the SLICE-to-SLICE carry connection is made through a dedicated fast wire between SLICEs, and not through general purpose interconnect. The intermediate multiplexers along the carry chain between LUTs A and B have their select inputs tied to logic-`1`. Observe that the flip-flop receiving the potentially "glitchy" signal ($N2$) is placed within the top SLICE (utilizing an extra SLICE is not needed for this flip-flop).

The key benefit of our PUF design is that it is described completely in VHDL and can be automatically handled by synthesis, place and route tools, without manual intervention. It can be incorporated naturally within a "push-button" FPGA design flow. All of the matched circuitry lies within SLICEs and no external matched routing or hard macros are required. Indeed, the routing for signal $N2$ does not need to be matched with that of any other signal. $N2$ is considered as a normal unconstrained design signal to be handled by the router. An additional advantage of our design is its small size – each PUF bit is generated by two SLICEs.

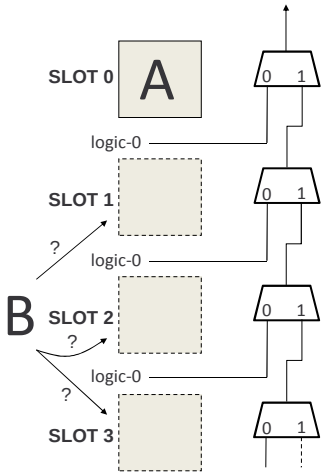

Fig. 5. Glitch pulse width tuning through SLICE LUT positioning.

While the focus of this paper is on the design of a PUF circuit to produce individual PUF bits, it should be apparent that the PUF design can be integrated and used within a challenge/response framework, as done in other research on PUFs (e.g. [8]). A simple approach is to have the challenge input word drive the select inputs on wide multiplexers, as shown in Fig. 7. The challenge selects two different PUF bits whose values are exclusive-OR'ed to produce an output bit depending on the challenge. Additional multiplexers can be added, each wired differently, to produce a multi-bit output.

IV. Experimental Validation

We instantiate 128 instances of the design described in Section III to generate a 128-bit signature. We evaluate the design using six Virtex-5 FPGAs on Xilinx XUPV5-LX110T development boards. The Xilinx LX110 Virtex-5 FPGA on each board has about 69,000 LUTs, of which about 18,000 may be used as RAMs. Our 128-bit PUF uses less than 2% of such RAM LUTs. The board has a serial RS-232 output which we use to communicate the PUF signature to a connected PC. We clock the PUF using the 27 MHz clock signal available on the board.

In addition to comparing PUF signatures across different FPGA chips, we can also implement a PUF multiple times on a single chip – each time in a different region of the chip. Naturally, we expect that any two FPGA chips should differ more (from the variations standpoint) than any two regions on a *single* chip. Consequently, if PUF signatures for different regions on a single chip are substantially unique, we have strong evidence that signatures between chips will be *at least* as unique. Following this reasoning, in addition to comparing PUF signatures across six different Virtex-5 chips, for each Virtex-5 chip, we investigated six PUF implementations, one implementation in each of the six regions shown in Fig. 8. Each region spans half the die horizontally and a third of the die vertically. PUF placement was constrained to regions using range constraints provided to the Xilinx synthesis tool. With six PUFs per chip and six chips in total, we have 36 PUF implementations. The

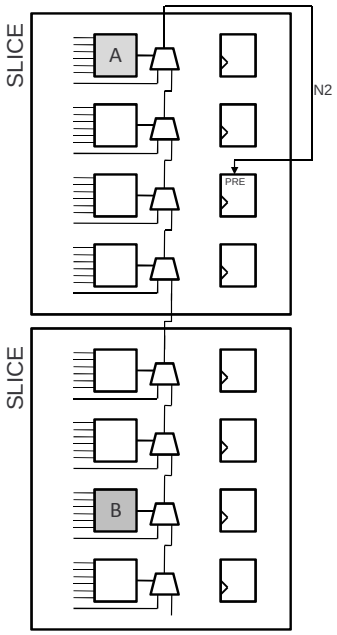

Fig. 6. Tuned PUF bit generator.

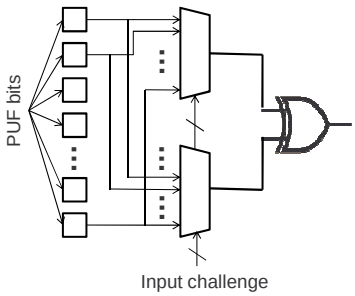

Fig. 7. PUF circuit in challenge/response framework.

methodology of using multiple unique implementations of the same circuit on a single chip can only be applied for reconfigurable platforms such as FPGAs. We believe the same methodology may prove useful in researching other aspects of process variations.

To analyze signature uniqueness, we consider the Hamming distance between all PUF pairs, producing $(36 \times 35)/2 = 630$ data points[2]. A probability histogram of such distances is shown in Fig. 9. If logic-`0` and logic-`1` were equally probable, one would expect the distribution to be clustered around an expected value of 64 bits. For the data in Fig. 9, the average distance between any two pairs was 61.8, which is relatively close to the expected 64. We found a bit bias towards logic-`1` across the signatures, which reduced average Hamming distance. The smallest distance between any two signatures was 43; the largest distance was 79. Surprisingly, the distribution in Fig. 9 qualitatively resembles a Gaussian distribution.

[2]The Hamming distance between a pair of 128-bit signatures is the number of bit positions in which the two signatures differ from one another.

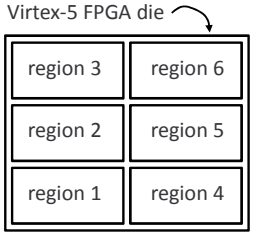

Fig. 8. FPGA regions corresponding to different PUF implementations.

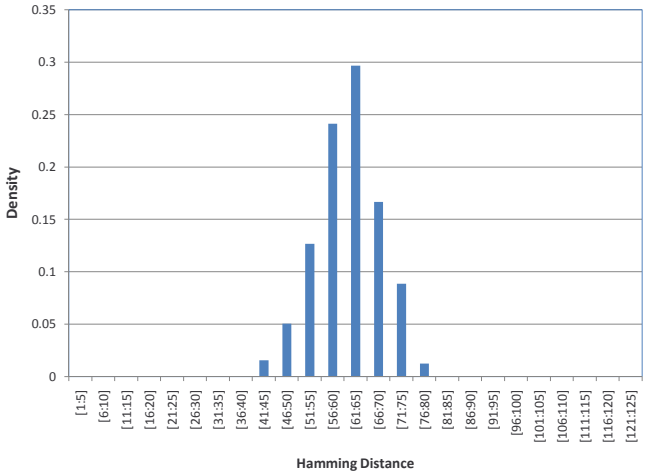

Fig. 9. Signature uniqueness: Hamming distance distribution (N = 630).

An interesting question is whether the 128-bit PUFs in the different regions on a single FPGA are "closer" to one another compared to PUFs on different chips. As there are six PUFs on each FPGA, we can compute $(6 \times 5)/2 = 15$ data points for each chip, yielding $6 \times 15 = 90$ data points in total. A probability histogram of the within-die Hamming distances is shown in Fig. 10. Observe that the histogram shape is quite similar to that of Fig. 9. On average, the distance between signatures was 59, which is again close to the expected value of 64. The average in this case may be influenced the smaller sample size. Nevertheless, the data suggests that for the proposed PUF design, the regions within a single die are roughly as dissimilar as regions across dies.

Circuit speed is influenced by die temperature and therefore, PUF signatures are sensitive to temperature. Ideally, signature variation with temperature would be small or non-existent. For the data given above, room temperature FPGAs were configured and a PUF signature measurement was taken immediately. We compared those "cool" signatures with "hot" signatures gathered at high temperature to gauge PUF reliability. The Virtex-5 FPGA has a built-in system monitor that can measure die temperature within 4°C [17]. Die temperature can be monitored graphically using the Xilinx ChipScope tool. We used a hair dryer to heat up the Virtex-5 FPGA chips to a die temperature of 70°C and then made "hot" PUF signature measurements.

For each 128-bit PUF in each region on each chip, we computed the Hamming distance between signatures at high and low temperature. The $6 \times 6 = 36$ Hamming distances are plotted as a histogram in Fig. 11. 72% of the signatures changed by five or fewer bits at high temperature and no signature experienced more than 10 bit flips. Comparing the histogram in

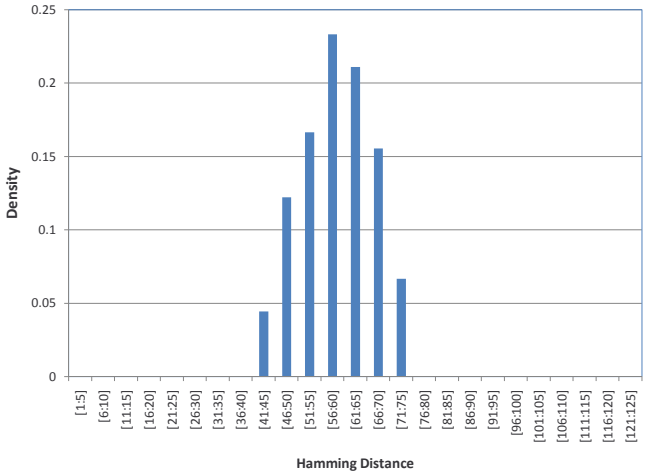

Fig. 10. Same-die signature uniqueness: Hamming distance distribution (N = 90).

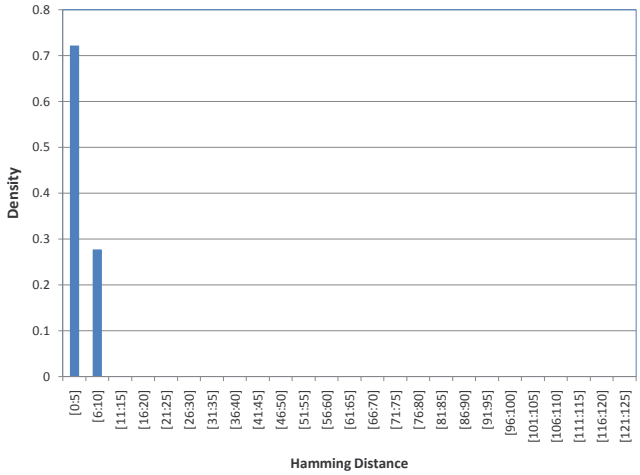

Fig. 11. Signature variability at high vs. low temperature (N = 36).

Fig. 9 with that in Fig. 11, one can see a large gap in the distributions roughly between 10-45 bits, which demonstrates that the proposed PUF design can be effective for device authentication and anti-counterfeiting. On average, 3.6% of signature bits flip under high temperature conditions, which is in line with other published PUF circuits in their "raw" form. As in other works, averaging and redundancy techniques can be applied to improve PUF reliability at the expense of area [8, 6]. For example, [8] suggests that BCH error correcting codes can be used to correct bit flips in PUF signatures.

V. Conclusion

Physically unclonable functions are circuits that leverage process variations to compute a unique signature for a fabricated IC. PUFs have varied applications, including anti-counterfeiting, hardware security, and cryptography. In this paper, we proposed the first FPGA-specific PUF design – one that takes advantage of the FPGA logic and routing architecture. Compared with prior work, our design consumes little area and is easy to implement and incorporate into a surrounding design. Measured results on 65 nm Virtex-5 FPGAs demonstrate the PUF signature uniqueness and its reliability at high temperature. Future work will involve the development of FPGA flows that employ PUF signatures for IP protection/licensing and anti-counterfeiting.

The VHDL for the proposed PUF is available for free download at the author's website.

Acknowledgements

The author thanks Xilinx for providing the Virtex-5 FPGA development boards. The author thanks Dr. Qiang Wang for his helpful comments on the manuscript.

References

[1] *Cisco Statement on Counterfeit Goods*, Cisco Corp., San Jose, CA, 2008.

[2] *In China, Knockoff Cellphones are a Hit*, The New York Times, New York, NY, April 2009.

[3] J. Roy, F. Koushanfar, and I. Markov, "EPIC: ending piracy of integrated circuits," in *IEEE/ACM Design Automation and Test in Europe*, 2008, pp. 1069–1074.

[4] *http://www.verayo.com/product/softpuf.html*, Verayo, Inc., San Jose, CA, 2009.

[5] *Virtex-5 FPGA Data Sheet*, Xilinx, Inc., San Jose, CA, 2007.

[6] H. Yu, P. Leong, H. Kinkelmann, L. Moller, and M. Glesner, "Towards a unique FPGA-based identification circuit using process variations," in *IEEE Int'l Conf. on Field Programmable Logic and Applications*, 2009, pp. 397–402.

[7] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Lightweight secure PUFs," in *IEEE Int'l Conf. on Computer Aided Design*, 2008, pp. 670–673.

[8] G. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *ACM/IEEE Design Automation Conf.*, 2007, pp. 9–14.

[9] S. Kumar, J. Guajardo, R. Maes, G.-J. Schrijen, and P. Tuyls, "The butterfly PUF: protecting IP on every FPGA," *IEEE Int'l Workshop on Hardware-Oriented Security and Trust*, pp. 67–70, 2008.

[10] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Techniques for design and implementation of secure reconfigurable PUFs," *ACM Trans. on Reconfigurable Technology and Systems*, vol. 2, no. 1, pp. 1–33, 2009.

[11] Y. Su, J. Holleman, and B. Otis, "A 1.6 pJ/bit 96% stable chip-ID generating circuit using process variations," in *IEEE Int'l Solid State Circuits Conf.*, 2007, pp. 15–17.

[12] J. Guajardo, S. Kumar, G. Schrijen, and P. Tuyls, "FPGA intrinsic PUFs and their use for IP protection," in *ACM Int'l Workshop on Cryptographic Hardware and Embedded Systems*, 2007, pp. 63–80.

[13] H. Patel, Y. Kim, J. McDonald, and L. Starman, "Increasing stability and distinguishability of the digital fingerprint in FPGAs through input word analysis," in *IEEE Int'l Conf. on Field Programmable Logic and Applications*, 2009, pp. 391–396.

[14] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Testing techniques for hardware security," in *IEEE Int'l Test Conf.*, 2008, pp. 1–10.

[15] *Stratix-III FPGA Family Data Sheet*, Altera, Corp., San Jose, CA, 2008.

[16] D. Lewis, E. Ahmed, D. Cashman, T. Vanderhoek, C. Lane, A. Lee, and P. Pan, "Architectural enhancements in Stratix-III and Stratix-IV," in *ACM/SIGDA Int'l Symp. on FPGAs*, 2009, pp. 33–42.

[17] *Virtex-5 FPGA System Monitor User Guide*, Xilinx, Inc., San Jose, CA, 2009.