Temperature-triggered Behavioral IPs HW Trojan Detection Method with FPGAs

Xiaotong Li and Benjamin Carrion Schafer[1]
The Hong Kong Polytechnic University
Department of Electronic and Information Engineering
(b.carrionschafer@polyu.edu.hk[1])

***Abstract*—This works targets the detection of temperature triggered HW Trojans, in particular for third party behavioral IPs (3PBIPs) given in ANSI-C. One of the biggest advantages of C-Based VLSI design is its ability to automatically generate architectures with different trade-offs by only setting different synthesis options. This work uses this property to detect temperature-triggered HW Trojan. A complete design flow is presented. It comprises two main phases: (1) In the first phase, a design space explorer generates micro-architectures with different area vs. power trade-offs automatically for the given behavioral IP. (2) The second phase, maps three of these micro-architectures with different power profiles onto a reconfigurable computing board to create a 3-way redundant system. This system combined with a majority voter scheme is used to detect if a HW Trojan is present in the behavioral IP. Having different power profiles implies that each micro-architecture has a different thermal behavior and thus will trigger the HW Trojan at different time intervals. The outputs of the three designs are compared for discrepancies at regular intervals, allowing our method to therefore exactly pinpoint the exact trigger temperature of the HW Trojan. A case study is presented showing the effectiveness of our method.**

I. Introduction

Design for Trust of Integrated Circuits (ICs) has emerged as an extremely important topic in IC design methodology. The globalization of IC design and manufacturing process poses serious concerns about their trustworthiness and security. It is nowadays virtually impossible to fully design and manufacture an IC in-house using in-house tools. With the increase time-to-market pressure, companies further rely on third parties for the development of ICs. The International Technology Roadmap for Semiconductors (ITRS) suggests that by 2020 a 10x productivity increase for designing complex System on Chips (SoCs) is needed. The main factor predicted to help achieving this goal is the use of Intellectual Properties IPs (third party or re-used from previous projects). ITRS estimates that at around 90% of the SoCs will be composed of IP components [1].

Having relinquished so much control for economic reasons and needing to rely on so many different third parties, opens the debate to how trustworthy the HW systems actually are. In this work, we consider the malicious modifications of behavioral IP's behavior. These malicious modifications are called Hardware Trojans and are intentional modification of an electronic circuit or design, which results in the incorrect behavior of an electronic device when in operation. Practical methods to ensure trustworthiness are needed at the pre and post-silicon level, which do not disrupt the current VLSI design flow. These methods and design techniques should seamlessly integrate with current design flows. We address the issue of HW trustworthiness for third party behavioral-level intellectual property protection (3PIPP) to detect HW Trojan which trigger at particular temperatures. These HW Trojan are extremely difficult to detect as no functional verification methods can capture their trigger condition.

Temperature has become in the last years a critical design parameter. Therefore, most complex IC nowadays include thermal sensors in order to perform active thermal management. E.g. Intel's Core i3/5/7 processors include thermal sensors and active thermal management support to control the power consumption and maximum temperature by dynamically adjusting the operating frequency and voltage of the processor, based on the thermal sensors' reading. State of the art FPGAs also contain thermal sensors that allow the thermal monitoring of the device. These thermal sensors are often instantiated as sensor IPs into the FPGA design. The main problem is that the FPGA vendors cannot provide simulation models of these sensor, which makes these systems vulnerable to HW Trojans, as it is not possible detect these during normal functional verifications.

This work deals with the detection of HW Trojan built into behavioral IPs which make use of the temperature readings from these thermal sensors by triggering the malicious behavior at a certain temperature. The detection method proposed in this work maps different version of the same IP with different micro-architectures and thus power profiles onto an FPGA to detect the temperature at which the HW trojan is triggered. Although in theory the HW trojan used in this work could be detected by performing a functional simulation for differen temperature readings, this is impractical as state of the art thermal sensors can measure temperatures between -40°Cto 125°C, which would mean that a simulation for each temperature value would be needed, leading to 40+125=165 full functional verification simulations, as the HW trojan proposed here is triggered at a very unique temperature. Moreover, more advanced HW trojan which make use of the fact that gate delay increases with temperature could also be captured by our method and which cannot be detected with any functional verification method as these delay increases cannot be simulated.

II. Previous Work

HW Trojans can be broadly classified based on either the insertion phase (e.g. specification, design, fabrication), the level of abstraction (e.g. system-level, RT-level, Gate-level, physical level), its activation mechanism (e.g. always on, internally triggered, externally triggered) its effects (e.g. downgrade performance, leak information, change of functionality) and/or its location (e.g. processor, memory, I/O)[2].

Hardware Trojan are typically only activated under very rare conditions, which makes them very difficult to detect during test time. Therefore, so far, statistical approaches have been proposed for logic testing [3]. Another popular detection technique is based on side-channel analysis [4] . However large process variations, noise, and environmental factors can mask the effect of the Trojan, especially for smaller ones. A survey on the most relevant protection techniques can be found at [5].

The main problem with these previous approaches is that they all require a golden Trojan-free IC or functional model.
In the case of detecting HW Trojan in third party IPs (3PIP), this is not the case. IP providers only provide a single version of the IP and there is no reference model against which the IP received can be verified.
To address the detection of 3PIPs HW Trojan Rajendran et al. [6] proposed to source the IP from different vendors, and consider that it is unlikely that both IPs are *infected*. The system can detect malicious outputs by duplicating the 3PIPs and comparing their outputs. Cui et al. [7] extended this work by proposing a run-time recovery system which rebinds at runtime the IPs from different vendors in case that a malicious output is detected. The main drawbacks of these methods are that they involve large overheads, because all 3PIPs need to be duplicated and also require that all of the IPs are available from different vendors. Moreover the runtime recovery method proposed in [7] does not clarify how the rebinding of IPs is done at runtime. This previous work mainly targeted RTL 3PIPs.
Another approach that does not require a golden model at the RT-level makes use of the fact that Trojans are not activated during normal operation [8]. The authors present the term *Unused Circuits Identification (UCI)* to identify potential Trojan, for circuit parts not sensitized during the verification stage. Because their method might remove legitimate circuits, it inserts logic to detect if the removed circuits would have been activated, and triggers an exception if the hardware encounters this condition at runtime.
This work presents a framework to detect temperature-triggered HW Trojan without any golden model. Our proposed method is based on the design of fault tolerant systems based on redundant systems combined with a majority voter scheme. By instantiating three different implementations of the same design onto an FPGA, the temperature triggered HW Trojan will be triggered at different time intervals, and thus can be exposed, without the need of a golden reference design, required by most previous work.

III. Proposed Detection Method

Our proposed method consists of two phases. In the first phase, different micro-architectures (designs) with unique power profiles are generated using a HLS design space explorer. Three of these micro-architectures are in turned instantiated on a reconfigurable computing platform. Because they have different power profiles, they will also have different thermal behaviors. The same input data are applied to the three designs and their internal temperatures measured using a ring oscillator instantiated within each design in order to accurately capture their temperature. The output values are then compared and the system checks for any discrepancies. A HW Trojan exists if one of the three designs generates different outputs at a specific temperature. This is reported through the serial port to the computer connected to the FPGA board. Our proposed method is not meant to be in the final design released in the field, but only as a detection platform. If the IP passes the check, then it can be deployed safely in the field.

HLS Design Space Explorer : Fig. 1 shows an overview of the first phase of our method. Our proposed method takes as inputs the behavioral description given in ANSI-C or SystemC (behavioral IP) to be verified for temperature triggered HW Trojans, a pragma library and the general HLS synthesis options (target frequency and technology libraries). Commercial HLS tools make heavy use of synthesis directives in the form of pragmas (comments) to guide the synthesizer. Some of the most common pragmas include how to synthesizes arrays (RAM or registers), unroll or pipeline loops and inline or not functions. By setting these pragmas the synthesizer will generate micro-architectures with unique area, performance and power trade-offs. Out of all these micro-architectures, we are normally only interested in the Pareto-optimal designs. These are the designs on the trade-off curve shown in Fig. 1. The usual exploration criteria is area vs. performance, where performance is normally specified as the latency or throughput of the design, depending on the application. In this case, our method explores area vs. power. Area is important because the designs will be mapped onto an FPGA board with limited resources, while power is critical for our work as it will determine the temperature of the design. Power is the most important factor when it comes to thermal-build up in any digital circuit. Other factors like the sequence of execution of tasks [10] and the placement of components [11] have also been shown to play a major role. In order to avoid the effect of the placement on the temperature of each design, these will be placed symmetrically on the FPGA as shown in the next section.

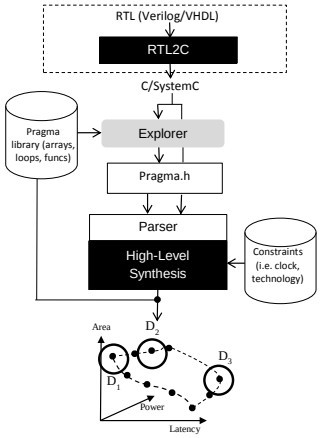

Fig. 1. Design space exploration flow starting from the behavioral description (ANSI-C/SystemC) or synthesizable RTL (Verilog/VHDL).

Because most of the IPs are currently provided at the RT-level, our method can also take synthesizable RTL as its input as shown in Fig. 1. Many commercial HLS tools provide RTL parsers in order to co-simulated newly generated C/SystemC code with legacy RTL code. These parsers read synthesizable RTL code and convert it to the tools' intermediate representation so that a simulation model of the entire design can be generated. This intermediate representation can in turn be converted to synthesizable C/SystemC. Our method can thus take as input any behavioral C/SystemC description or any synthesizable RTL code for exploration.
Out method then continues by generating a new unique set of synthesis directives. This is done by creating a new header file (pragma.h) with the synthesis directives. This source code is in turn parsed and synthesized. The HLS tool reports the area, latency and average power for the generated micro-architecture. In our case only the area and power are extracted

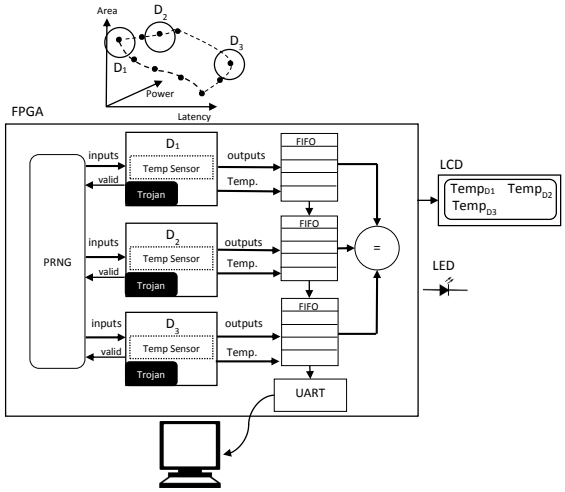

Fig. 2. Temperature-triggered HW Trojan Detection System Overview

as we are interested in finding 3 designs which fit on the FPGA board, with the largest possible average power difference. Because the search space is extremely large a genetic algorithm meta-heuristic is used to generate new pragma configurations, as genetic algorithms have shown to obtain good results for multi-objective optimization problems.

FPGA Implementation: Once the explorer finishes, the three designs with largest average power differences, which fit in the selected FPGA, are chosen and mapped onto a reconfigurable computing board as shown in Fig. 2. The three designs (D_1, D_2 and D_3) correspond to the ones with the largest average power differences. These normally also correspond to the fastest, slowest and one with intermediate performance. It should be noted that because the three designs are generated from the same behavioral description, they all contain the HW Trojan. They are placed at the FPGA at different corners so that the temperature build up of one instance does not affect the others as temperature also spreads laterally.

Because the FPGA that we target in this work (Virtex 4) does not contain any thermal sensors, these are replicated building ring oscillators (RO). A RO is placed inside each design, in order to accurately measure their temperature. It should be noted that each design requires its own thermal sensor, because if the same temperature reading is used for all three designs, then all three HW trojan would be triggered at exactly the same time.

ROs have been shown to be a very effective way of measuring internal die temperatures in FPGAs [12]. Each RO is composed of an odd number of inverters. As temperature increases, the mobility and threshold voltage (V_{th})decrease. This leads to longer propagation delays. Using different configurations, it was found that for 51 inverters lead to a good compromise between area and accuracy.

The fact that each of the three designs have a different latency poses the following challenges: First, in order to verify that they produce the exactly same outputs, each of them has to be passed the exact same inputs, but because the designs have different latencies, this means that one design will always *lead* of the others. Moreover, because our method wants to detect temperature-triggered HW Trojan, the execution cannot be *stopped* as this would allow the design to cool down and hence the HW Trojan would never be triggered. To overcome these issues, in order to apply the same inputs to the three designs, a pseudo random number generator (PRNG) is created for each of the designs, generating the exact same sequence for each of the designs.

The problem now is when to assign the new inputs to each of the designs. For this, our system makes use of a useful feature that most commercial HLS tool include to re-use untimed input and output test vectors used during the pure SW verification for verifying the synthesized, timed circuits. This verification method is called Dynamic Equivalence Checking (DEC), because it is allows to fully re-use the untimed input and output test vectors to compare the functionality of the initial untimed behavioral description and the synthesized timed circuit (either cycle-accurate model or RTL). During timed simulations, the timing of when data needs to be read and when a valid output is generated, needs to be made visible to the testbench. To enable DEC, commercial HLS environments generate for each input and output port an output valid signal. These valid signals are raised each time an input needs to be either read or a valid output has been written to the output port. HLS tools can very easily create these signals because they know in which particular state the inputs are read and in which states the outputs are generated. This feature is used by our method to apply test vectors from the PRNG, only when the valid signal of one of the three designs requests it. It is very important that all of the three implementations receive the same inputs in order to verify if the outputs match. Thus our system is independent of the actual latencies of the three designs making it extremely flexible for verifying any IP.

The next problem arises when deciding when to compare the output values. As each design has a different latency, the fastest design will generate outputs at a faster rate compared to the other designs. In order to solve this problem, the outputs are only stored into the output FIFO of each process when the a thermal threshold is reached. This is currently set to 1°Cin this work, but it is parameterizable. This means that whenever the temperature of any design increases by 1°C, the output value is stored in the FIFO. The problem now is that the three designs will not reach the same temperature at the same time. Therefore the system also counts the output valid signal pulses of each of the three designs and stores the counter value with the actual output values at each temperature interval at the FIFO. The output of the slower and cooler designs will in turn be compared when the valid signal pulses match the counted value of the hottest unit. Once the valid counter values match the outputs are compared. Any mismatch of the hottest unit indicates a possible HW Trojan and an LED starts flashing. This method also allows to dimension the FIFOs in such a way that it can be guaranteed that even if one of the units becomes hotter at a much faster rate than the other units, the system will still work. The size of the FIFO (S_{FIFO}) is set to $S_{FIFO} = bw_{outputs} \times (T_{max} - Tamb)$, where $bw_{outputs}$ is the bitwidth of the outputs of the design under verification, T_{max} is the maximum expected temperature,by default set to 100 and T_{amb} is the ambient temperature (in this work set to 20°C). The system is also connected through a UART to the PC to display the values stored in the FIFOs. The LCD on the FPGA board also constantly displays the temperature at each of the three designs.

IV. ADPCM Case Study

The adaptive differential pulse code modulation (ADPCM) design from the SystemC Synthesizable Benchmark suite (S2CBench) [13] is used as a behavioral IP to test the proposed detection flow. The IP was modified to include a thermal management unit, which turns the clock off if a certain temperature is reached (100°C). The output of the thermal sensor is therefore connected to the adpcm for temperature monitoring. The HW Trojan built into the design was programmed to when $T_{Trigger}$ =45°C, is reached. In this case the output of the adpcm is shifted by one to emulate the faulty behaviors of the circuit due to the HW Trojan. The HLS tool used is CyberWorkBench v.5.4 [14] from NEC targeting a Virtex 4 XCVSX35-10 FPGA, which is the same as the FPGA on the ML402 reconfigurable computing board on which the system is implemented. The target synthesis frequency is set to 50MHz. Table I shows an overview of the three designs obtained by the HLS explorer. D_1 has a latency of 4 cycles, requiring an area of 1,349 4-input LUTs, while consuming on average 17.2 mW. D_2 has a latency of 9 cycles, requiring an area of 689 4-input LUTs, while consuming on average 12.2 mW and D_3 has a latency of 19 cycles, requiring an area of 609 4-input LUTs, while consuming on average 9.1mW.

TABLE I. ADPCM configurations

ADPCM Version	4-input LUTs	Latency	$Power_{avg}$ [mW]
D_1	1,349	4	17.2
D_2	689	9	12.4
D_3	604	19	9.1

Table II shows the area in terms of 4-input LUTs for all the components in the system the contribution of each component to the total area in %. It can be observed that the components which require most of the area are the three designs under test and the compare and FIFO modules. The entire system occupies 14% of the entire FPGA (4,1731/30,720 LUTs). The proposed system, without the three designs only occupy 37% of the entire system, which comprises only 5% of the entire FPGA used in this work. This shows that our method could be used with larger behavioral IPs as the overhead of the circuit apart from the three designs under test is very small.

TABLE II. Total System Area

Component	LUTs	Number	Total Area	% Area
PRNG	8	3	24	0.01
ADPCM (D1)	1,349	1	1,349	0.32
ADPCM (D2)	689	1	689	0.17
ADPCM (D2)	604	1	604	0.14
Ring oscillator	51	3	153	0.04
Copmare+FIFOs	1,218	1	1,218	0.29
Top module misc	136	1	136	0.03
Total Area	4,055		4,173	1.0

Fig. 3 shows the measured results. It can be observed, as expected, that the temperature in D_1 grows faster than the temperature in D_2 and D_3, while the temperature build-up in D_2 is faster than in D_3, due to their different power consumptions. This leads to D_1 reaching the trigger temperature faster than the other two designs. In particular our system detected that the HW Trojan was triggered after 318, 352 and 390 seconds for D_1, D_2 and D_3 respectively. The temperature in each of the designs continued growing as show in Fig. 3 asymptotically towards 65°C. This means that for this particular design our method could only be used for trigger temperatures up to this temperature, unless the ambient temperature is increased (e.g. by placing the FPGA in an oven).

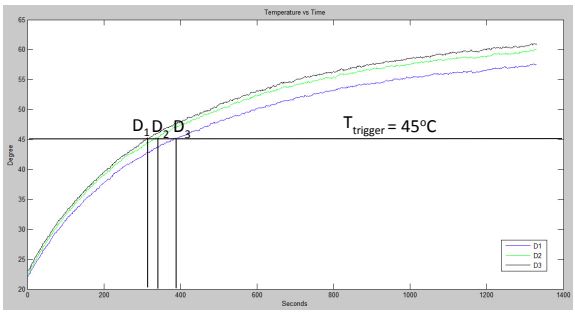

Fig. 3. Three ADPCM designs thermal-behavior vs. time

The results obtained from this case study, show that our proposed design flow works well for detecting temperature-triggered HW Trojan. The method can easily be adapted to any behavioral IP as the entire process has been fully automated.

V. Summary and Conclusions

In this work, we have presented a method to detect temperature-triggered HW Trojans in behavioral IPs. Our method makes use of one of the biggest advantages of behavioral VLSI design vs. conventional RT-level design: The ability of generating micro-architectures with unique characteristics without having to modify the original description. Our method hence explores the design space of the behavioral IP to be verified using not only area vs. performance as a cost function, but also power. Three designs with unique power profile are in turn automatically mapped onto an FPGA and a ring oscillator inserted inside in order to accurately measure their temperature. Our flow has been fully prototyped and tested on a Xilinx ML402 board and a case study of an APCM presented.

References

[1] International Technology Roadmap for Semiconductors, "www.public.itrs.net/Links/2013ITRS/2013Chapters," 2013.

[2] R. Karri, J. Rajendran, K. Rosenfeld and M. Tehranipoor, "Trustworthy Hardware: Identifying and Classifying Hardware Trojans," *IEEE Computer*, Vol. 43 (10), pp. 39-46, 2010.

[3] R.S. Chakraborty, F. Wolff, S. Paul, C. Papchristou and S. Bhunia, "MERO: A Statistical Approach for Hardware Trojan Detection," *Cryptographic Hardware and Embedded Systems (CHES)*, pp. 396-410, 2009.

[4] Y. Lin and Y. Makris, "Hardware Trojan Detection using Path Delay Fingerprint," *HOST*, pp. 51 57, 2008.

[5] X. Zhang and M. Tehranipoor, "Case study: Detecting Hardware Trojans in Third-Party Digital IP Cores," *HOST*, pp. 67-70, 2011.

[6] J. Rajendran, H. Zhang, O. Singangolu and R. Karri, "High-Level Synthesis for Security and Trust," *IEEE International On-line Testing Symposium (IOLTS)*, pp. 232-233, 2013.

[7] X. Cui, K. Ma, L. Shi and K. Wu, "High-Level Synthesis for Run-Time Hardware Trojan Detection and Recovery," *DAC*, pp. 1-6, 2014.

[8] C. Sturton, M. Hicks, D. Wagner, and S.T. King, "Defeating UCI: Building stealthy and malicious hardware," *IEEE Symposium on Security and Privacy*, pp. 64 77, 2011

[9] Z. Chen, X. Guo, R. Nagesh, A. Reddy, M. Gora, A. Maiti "Hardware Trojan Designs on BASYS FPGA Board," *Embedded System Challenge Contest in Cyber Security Awareness Week (CSAW)*, 2008

[10] D. Brooks and M. Martonosi, "Dynamic thermal management for high performance microprocessors," *Int. Symp. High-Perform. Comput. Arch*, 2001.

[11] C. H. Tsai and S. M. Kang, "Standard cell placement for even on-chip thermal distribution," *Int. Symp. Phys. Des.*, 1999.

[12] E. Boemo and S. Lopex Buedo, "Thermal Monitoring on FPGAs using Ring-Oscillators," *FPL*,pp.69-78, 1997.

[13] B. Carrion Schafer and A. Mahapatra "S2CBench:Synthesizable SystemC Benchmark Suite for High-Level Synthesis," *IEEE Embedded Systems Letters*, Vol.6-3, pp.53-56, 2014.

[14] NEC CyberWorkBench, "www.cyberworkbench.com," 2014.