Active Control and Digital Rights Management of Integrated Circuit IP Cores

Yousra Alkabani
CS Department
Rice University
Houston, TX 77005
yousra@rice.edu

Farinaz Koushanfar
ECE and CS Departments
Rice University
Houston, TX 77005
farinaz@rice.edu

ABSTRACT

We introduce the first approach that can actively control multiple hardware intellectual property (IP) cores used in an integrated circuit (IC). The IP rights owner(s) can remotely monitor, control, enable, or disable each individual IP on each chip. The approach introduces a paradigm shift in the microelectronic business model, nurturing smaller businesses, and supporting the design-reuse paradigm. The IPs can be controlled by the original designer or by the designers who reuse them. Each IP has a built-in functional lock that pertains to the unique unclonable ID of the chip. A control structure that coordinates the locking and unlocking of the IPs is embedded within the IC. We introduce a trusted third party approach for issuing certificates of authenticity, in case it is required for the applications. We present methods for safeguarding the approach against two attack sources: the foundry (fab), and the reuser. Experimental results show that our approach can be implemented with low area, power, and delay overheads making it suitable for embedded systems. The introduced control method is also low overhead in terms of the added steps to the current design and manufacturing flow.

Categories and Subject Descriptors

B.7 [**Integrated Circuits**]: Miscellaneous; B.6 [**Logic Design**]: Miscellaneous; K.6.5 [**MANAGEMENT OF COMPUTING AND INFORMATION SYSTEMS**]: Security and Protection

General Terms

Security, Design, Management

Keywords

IP Protection, Security, Active IP Control

1. INTRODUCTION

The state-of-the-art digital ICs are increasingly complex. The progressive demand for multiple applications, performance, and functionality for integrated circuits has resulted in extreme CMOS miniaturization that add to the complexity. Building, operating, maintaining and upgrading silicon fabs for the complex designs is prohibitively expensive, e.g., upgrade to the current technology, 45nm, costs about $4bn [4]. The leading edge design companies are fabless. Even the large semiconductor companies including Texas Instruments (TI) and Freescale that had in-house manufacturing recently started to outsource their fabrication.

Because of the complexity, adapting the design reuse paradigm is the key to address constraints such as low-power, real-time budgets, silicon efficiency, time-to-market, and low cost [14]. A consequence of the current shift towards the fabless business model and design reuse is increased horizontalization of the microelectronic industry. Integration of multiple functionalities, applications, and design techniques has lead to modularity and specialization of design houses.

Many fabless design companies, particularly the specialized IP core designers are small. Their major investment is the technical and engineering staff and human resources who work together to produce the IP product. If the IP is ever exploited the company loses its capital investment. It is also likely that the IPs accidentally or through negligence are misused. For example, a design engineer (who we call a "reuser") may not take the time to check each core's license agreement. The IP-core design companies only receive revenue when their core is licensed to reusers, regardless of the volume and profit of the end product(s) that typically include multiple IPs. The presence of smaller companies is essential for a competitive market, but those companies endanger consolidation in the current business model.

We propose a novel approach that allows the IP core providers to gain post-fabrication control over their IPs on each chip. The approach introduces a paradigm shift in the digital rights management (DRM) of integrated circuits IP cores for vendors, designers and foundries. Depending on the application, the method may be used to control the number of chips that implement the IP, to remotely and actively enable or disable the usage. The misuse of the IP products is not only detected, but also prevented. The method works by uniquely locking the functionality of the IP core embedded in the manufactured chips, such that the rights owner is the only entity who can provide the key to unlock it. Our contributions include:

- Introduction of the first architecture and implementation for individual control of each IP core, in a multi-IP design.
- Integration of locking into each IP's functionality and coordinating the IPs by the reuser's control core.

CASES'08, October 19–24, 2008, Atlanta, Georgia, USA.

• Control of the IP cores that may be done by the original core provider, the IP reuser, or both.

• Successful integration of the method within the standard synthesis flow, with a minimal addition of steps.

• Low-overhead and efficient implementation of the approach on chips containing multiple industrial benchmark IP cores.

• Ensuring trustworthiness of the key-exchange protocol by introducing a trusted third party providing certificate of authenticity.

• Discussion of attacks and providing safeguards.

• Introduction of a number of possible applications that are enabled by the new multi-IP protection method.

Motivational Example: Figure 1 presents a reuser's design which contains multiple IP cores. The cores denoted by IP_1, IP_2, ..., to IP_K are the protected ones. The functional control unit of each IP is represented by a finite state machine (FSM). The circuit designer (reuser) includes two new modules in her design. One added part is an identification (ID) circuitry that extracts the unique identification bits for the chip using the silicon variability [10, 18, 17]. The other addition is a control module that is embedded within the central controller of the chip. Each protected IP is directly connected to the the ID circuitry. Each of the protected IPs contains a lock within their functional states.

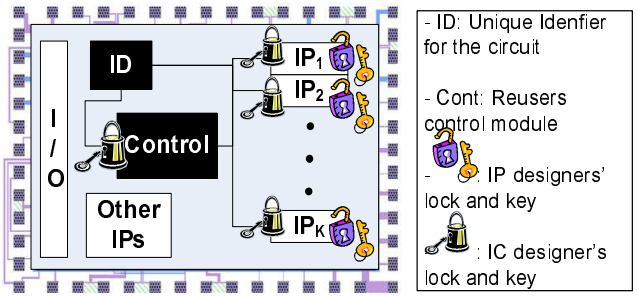

Figure 1: A reuser's design including multiple IP cores. Each IP may be locked/unlocked by the IP designer or the reuser, depending on the application.

The remote enabling/disabling provides two sets of locks and keys, one for the designer and one for the reuser. The locks are embedded within the control structure of each IP that can be represented by a finite state machine (FSM) [7]. There are two major advantages for the selected locking/unlocking mechanism: (i) the IDs come from the variations of the physical structure of silicon and are therefore random and unclonable, and (ii) the locks are integrated within the functional control structure, so removing or tampering the lock would tamper the functionality, rendering the IP unusable. Furthermore, as we will show, modifying the FSM does not result in a significant overhead.

Many protection, security, and DRM protocols can be enabled by the new IP locking/unlocking method. For example, the core providers can protect their IPs against overbuilding a licensed product, since each IP would be locked upon manufacturing. As another example, the reuser who has another set of locks/keys on the IP can select which IPs (or even features) are activated on the chip, e.g., for charging the customers who are willing to pay for added features.

In the remainder of the paper we show the details of the new approach, implementation, experiments, and applications.

2. RELATED WORK

Methods for digital design reuse and intellectual property trading are emerging [2, 22, 14, 5, 7, 6]. Protection of IPs in the reuse-based design flow is of paramount importance, but the prior work on individual IP protection has been limited. Most of the effort has been focused on FPGA soft IP core protection [25, 24]. A number of watermarking methods for IP identification have been proposed, but unlike our method that is active and uniquely locks each chip, a watermark is passive and is the same on all the chips implementing the same design [20, 21, 26]. A watermark can only be used to solve disputes about illegal usage of a design. It cannot identify, activate or disable individual ICs or IPs.

The inherent and unclonable silicon manufacturing variability has been used to uniquely identify each chip [16, 6]. Delay-based physically unclonable functions (PUFs) were constructed to extract the variability in circuit timing as a function of input (challenge) bits, generating a unique output (response) that can be used for identification and security [11, 18, 17]. PUFs were implemented in both ASICs and FPGAs [15, 12]. Several applications of PUFs are emerging, including RFID, proof of execution on a specific processor, securing processors, and active metering [10, 11, 12, 7, 8].

Recently, securing IPs in an ASIC design by individually tagging each core was proposed [19]. Since the tags are separated from the functionality, they are subject to removal attacks by both the reusers and the foundry. Note that approaches that use traditional implementations of cryptography protocols for securing at the low level are both high overhead and non-secure [23, 1], since the digitally stored keys are subject to physical and side-channel attacks [9].

Our new approach adapts the mechanism in [7], who integrated the unique identifiers of the chip into its control structure. The approach presented here includes several new aspects: First, multiple IP cores are controlled, not just one. Second, we consider interactions among the IP core designers, reusers, and the foundry, whereas the previous work only considered the designer-foundry relation. Third, unlike the previous work that only developed a control mechanism, we create a system-level secure IP integration solution and discuss the supply chain interactions. Fourth, we introduce the role of trusted integrator who will be useful for a secure design flow. Fifth, the reuser's role and possibilities of attacks are discussed for the first time. Lastly, the new approach directly applies to a number of novel system-level security, protection, and DRM methods that can be very useful for embedded systems (Section 8).

3. FLOW OF THE ACTIVE CONTROL FOR IP CORES

Figure 2 shows the overall flow of the new IP protection approach. There are four main entities involved: (i) IP rights owners (IP designers) who design, format and sell the individual IPs, (ii) IC rights owner (reuser) who integrates multiple IPs, including the open IPs and I/O interfaces, into one IC, (iii) The fabrication plant (fab), and (iv) an authorized system verifier; who we call a *certificate authority (CA)*. This entity ensures the trust between hardware IP providers, reusers, and the fab.

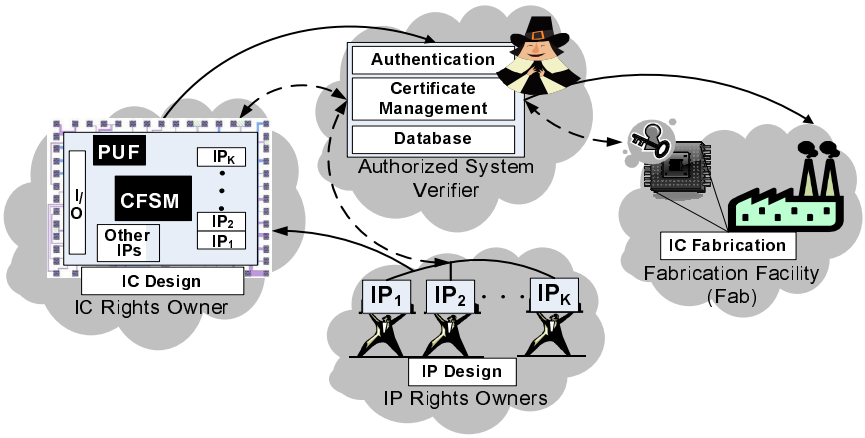

Figure 2: The flow of the active control for integrated circuits' IP cores.

While the first three components are commonly present in the IC design cycle, the last component is new. CA is the trusted third party component for many asymmetric cryptography protocols, including several public key infrastructure (PKI) schemes. The new model is an asymmetric security scheme based on the keys provided by the IP designers and system designer. The CA provides trust by authorizing the parties; preventing possible breaches.

The flow can be described as follows. The IP designer forms the FSM of the design by using the high level design description. Then, the lock(s) are strategically embedded in the FSM. The modified finite state machine is called the boosted finite state machine (BFSM). The reuser may integrate multiple locked IPs, in addition to other components, including her own designs, unlocked IPs, I/O peripherals, memory, and the master identification/control parts. The master identification/control consists of a controlling finite state machine (CFSM) and a PUF. The CFSM interacts and controls the various IPs; it can enable/disable the other components. The PUF provides a mean for identifying each IC implementing the design in a unique and unclonable way. The ready-to-fab designs are shipped to the CA who certifies the IP cores and the reuser. The material is then sent to the fab who makes the masks and produces a number of ICs as specified by the contract. The operations described so far are shown by solid arrows on the figure. The dashed arrows present the steps required for key exchange transactions.

The fabricated ICs are nonfunctional and have locks on the CFSM and on the protected IPs from the providers. For each IC, the fab tests the PUF input and runs it through the flip flops (FFs) scan chain. The state of the IC will be read out from the FFs and sent to the CA who will in turn supply the state of each chip to the authorized reusers and IP providers. Each of the contacted parties will produce the specific keys to unlock the component. Also, the IP provider computes the error correcting code (ECC) for the lock, to mask the possible few changes caused by the fluctuations in the PUF identifiers. The keys are then sent back to the CA, who certifies the consent of the rights owners before sending them to the fab.

4. IP CONTROL METHOD

In this section, we present the main modifications made to a multi-IP design to apply the method.

4.1 BFSMs

Each of the IP designers need to modify the FSM of their designed IP such that they embed a lock in it. The modified control structure is the BFSM. The BFSM is designed such that both its states and transitions are a function of the unique chip identifiers. The BFSM attempts to form a unique control path on each of the chips, while all the chips are from the same mask [7].

The BFSM of an IP core should satisfy the following properties.

- It must have incorrect functionality (locked) as long as the key is not provided.
- The key can be easily computed by the party who knows the BFSM structure and difficult to find otherwise.
- Knowing the key for one IC must not help in finding the key for another IC of the same design.
- Once the key is provided, the IP would function correctly.

Note that unlike symmetric cryptography where the keys are used to reverse a trap-door function and revealing the keys tampers the security, the keys here do not convey significant information about the lock. This is because the lock is in the structure of the state transition graph that is only known to the designer.

The same BFSM structure can be exploited to disable the chip during its operation. All what is needed is to modify the locks. For example, changing the PUF challenges will ensure that the functionality is trapped in a locked state.

4.2 CFSM

The overall FSM of the design that is devised by the reuse designer is also manipulated such that it embeds locks that allow the chip designer to lock/unlock her designed parts. Next, some states for controlling the other IPs are also included by the IC designer. We refer to these added IP control signals as CFSM. The CFSM gives the chip designer a level of control over the several IPs that are included in the design. For example, the CFSM receives signals from the IP cores about their locked/unlock status. The CFSM can also generate control signals that can enable or disable various IPs on the chip. There are many applications that can benefit from the CFSM (see Section 8).

4.3 PUF

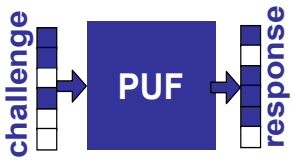

Figure 3: PUF challenge/response pairs.

PUF is the circuitry which generates random unique values per chip. Figure 3 demonstrates the high level block diagram of a PUF [10]. The PUF circuit generates a unique response (output) for each input vector (challenge) that is applied to it. Even though the response varies from one chip to the next, the response to the same challenge remains the same over time.

PUF has a much larger overhead compared with BFSM and CFSM. Thus, we share it among the IPs to reduce the overhead. There is a need to ensure that the PUF is properly connected to the IPs so that the IP rights owner receives her proper royalties. The trusted third party (authorized system verifier) ensures the proper interface of PUF to the BFSMs before sending the design files to the fab.

5. IMPLEMENTATION

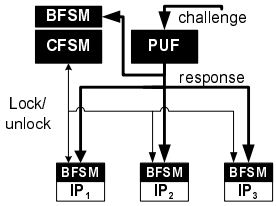

Figure 4: System block diagram.

Figure 4 shows the block diagram of the system components described earlier. Let us assume that we have three IPs denoted by IP_1, IP_2, and IP_3. The response of the PUF is connected to the IPs' BFSM, and the CFSM communicates with the BFSMs to control (lock/unlock) them. We outline the implementation of the BFSM, PUF, and CFSM.

5.1 BFSM Implementation

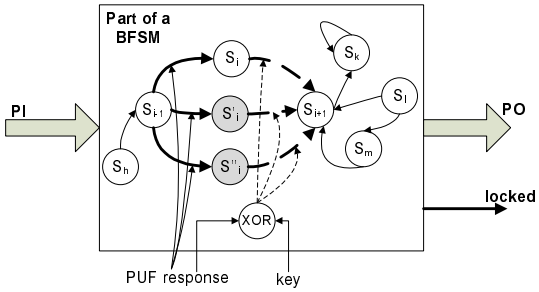

Figure 5: Implementation of the BFSM.

The implementation of BFSM is inspired by [7] but the BFSM was further adapted and modified to include more states and communications with the CFSM. Figure 5 shows a part of a BFSM on a sample IP core where a state S_i is replicated twice as S'_i and S''_i. The transitions to S_i from S_{i-1} are copied to its replicated states such that based on the PUF response, either S_i or one of its replica is reached. The reached state is only a function of the PUF response. However, the transitions from the replicated states to S_{i+1} are a function of both the PUF response and the key. The key and the response are XOR'd; if the output is correct, the valid state S_{i+1} will be reached. Otherwise, a wrong transition (not shown on the figure) will be taken. PI/PO represent the set of primary inputs/outputs to the BFSM. Whenever a wrong transition is taken, the flag signal from the IP's BFSM is set to 1 to inform the CFSM that the BFSM is still unlocked. The flag value is 0 otherwise. The BFSM implementation steps can be summarized as follows:

1. The n states with the least number of outgoing edges are selected for replication.
2. Each selected state is replicated m times.
3. Transitions to the replicated states are a function of the PUF response and are thus unique to each chip.
4. Transitions from the replicated states are a function of the PUF response and the key. Correct transitions are only taken if the key is properly set. Incorrect random added transitions are taken when the key is wrong.

5.2 PUF Implementations

We implement the delay-based PUF introduced in [11]. The response is found by comparing the delay of two parallel paths that must be the same, but vary because of manufacturing fluctuations. The signal starts at the common starting point of the two paths on the left and ends at an arbiter which is inserted at the right end of the two parallel lines. If the signal on the top path arrived earlier, the arbiter output will be zero; otherwise, its output would be one. The parallel paths are divided into multiple segments, such that each segment is controlled by a switch. Different combinations of the path segments are selected by the switches, causing the racing path pair and also the arbiter output (response bit) to change.

The above PUF is vulnerable to modeling attacks because of its linear structure. Feedforward arbiters are used to alleviate this problem [15]. The added arbiters compare the delays of two partial path pairs and use the arbiter output as the selector line for a forward switch in the circuit. Figure 6 shows an example of a two bit output delay-based PUF with random feedforward arbiters which may also connect different path pairs. Switches s[1] to s[n] represent the cascade of switches for the first output, and s'[1] to s'[n] are the switches for the second output. From each path pair, we randomly select the output of a few switches and connect them to arbiters, then connect the output of these arbiters to selection lines of other switches constructing a feedforward connection. The selection lines of switches that are not connected in a feedforward (not shown in the figure) represent the challenge to the PUF, while r[1] and r[2] represent the response of the PUF.

5.3 CFSM Implementation

The CFSM is implemented as a finite state machine that is embedded and hidden inside the main FSM (BFSM) of the IC. A block diagram of the CFSM control signals is shown in Figure 7. The CFSM inputs can be divided into two groups:

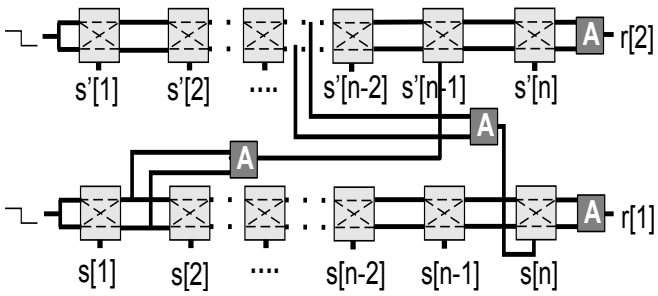

Figure 6: Implementation of the PUF.

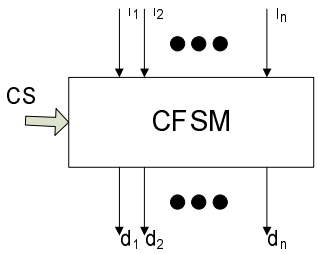

Figure 7: Implementation of the CFSM.

(1) inputs coming from the IP cores' BFSMs ($l_{1:n}$), and (2) external inputs (CS) that can be used to control the IC and enable/disable the IPs remotely. $l_{1:n}$ signals from BFSMs inform the CFSM if an IP is locked. The CS signals are used to upload the main key that determines which features (IPs) can be activated on a particular IC. Note that setting this key will not unlock the IPs, however, it will only help preventing the CFSM from disabling the whole chip when the IPs are locked. The output signals from the CFSM are $d_{1:n}$ that represent the disabling signals for the BFSMs in the IC. The CFSM continuously monitors all the BFSMs and if a BFSM that is supposed to be enabled is locked, the CFSM disables all the enabled IPs to detect and fix the control FSM.

6. ATTACKS AND SAFEGUARDS

We envision two categories of attacks on the proposed method: foundry level attacks and IC designer level attacks. The IC designer (reuser) and the foundry do not have the same knowledge about the design and do not share the same objectives. For example, the reuser may tamper with the communicated signals to the IPs to overrule the owners' rights. The foundry may also overlook the rights of the IC designer.

Foundry level attacks and countermeasures are:

- **Brute-force attack.** This attack can be performed by continuously applying random inputs to each IP until the correct value of the key is found. This attack is not feasible for one IP because the probability of guessing the correct key is extremely low [5]. Having more than one IP locked in addition to the main design renders the attack even more infeasible.
- **Reverse engineering of the BFSMs and the CFSM.** One might try to reverse engineer the BFSM and the CFSM by STG extration. However, the computation of the STG is a computationally intractable task especially that the BFSMs are enlarged versions of the FSMs of the IPs in the system, and the CFSM is obfuscated by hiding its states within the large state-space of the IC's main FSM [20].
- **PUF emulation.** This attack attempts to emulate the behavior of the PUF of one unlocked IC and replicate it on the others. However, this attack is infeasible in the state-of-the-art manufacturing and software emulation is much slower and can be detected [18, 17].
- **Combinational redundancy removal.** Using a combinational redundancy removal software, one can try to remove all the extra states added to the different parts of the design. However, since all the modifications are integrated within the functionality of the different IPs, they are not redundant and this attack will not be successful.

IC designer level attacks and countermeasures are:

- **Bypassing the PUF.** The adversarial reuser may try to bypass the PUF interface to the other IPs so that only one key is needed to unlock different IPs, maintaining only the connections of the PUF to the main BFSM to keep the reuser rights. However, it is the responsibility of the CA to check the interfaces and ensure that the PUF is properly connected to the IPs.
- **Tampering with the PUF.** The designer can tamper with the PUF such that one of the racing paths is much longer than the other. This can cancel out the effect of MV and produce deterministic output for all the ICs. However, the trusted system verifier should also test and certify the PUF's randomness [18, 17].

7. EXPERIMENTAL RESULTS

The proposed method is implemented and evaluated using the Berkeley SIS synthesis tool. All the programs are written in C. MCNC'91 sequential benchmarks are used to represent FSMs of different IPs. It should be noted that the FSM that contains the control part of any IP represents a very small fraction of the overall size of the design [13]. Thus, even tripling the overall area or power of these FSMs will not significantly affect the overall area and power of the IP. However, the delay of the FSM can affect the speed of the IP and thus, delay is the most important design metric in our implementation.

We show the overhead for using one and five IPs. Table 1 demonstrates the overhead when applying the metering method on one IP. The overhead number includes the overhead due to both the BFSM and the CFSM. The first column represents the benchmark number (C#) which will be used to refer to the benchmark in this section. The second column represents the name of the benchmark circuit. The third column shows the number of primary inputs (PIs) of the benchmark before modification. The fourth, fifth, and sixth column show the area, delay, and power overheads of the original benchmark. The rest of the columns show the area, delay and power of the modified IP and the percentage overhead of each parameter. It can be seen that the area and power overheads are on the average 143% and 131% respectively.Also, the delay overhead is low and is not affected by the number of IPs on the ICs. Thus, we do not report the delay overheads in our subsequent evaluations.

Next, we add a 16 stage random feedforward PUF with 64 cascaded switches per stage. The PUF has a total of 64

C#	area	%	power	%
7,4,1,5,8	13,281	106	50,702	121
2,2,4,2,7	7,611	101	29,129	112
3,7,8,1,8	9,161	123	33,545	135
3,5,8,6,1	13,858	139	47,135	128
2,5,6,6,5	17,573	137	62,276	129
6,7,1,2,3	8,187	110	31,528	127
7,2,5,3,8	12,650	106	47,970	117
8,5,1,3,2	13,309	124	45,789	115
mean	11,954	118	43,509	123

Table 2: CFSM overhead for integration of five IPs.

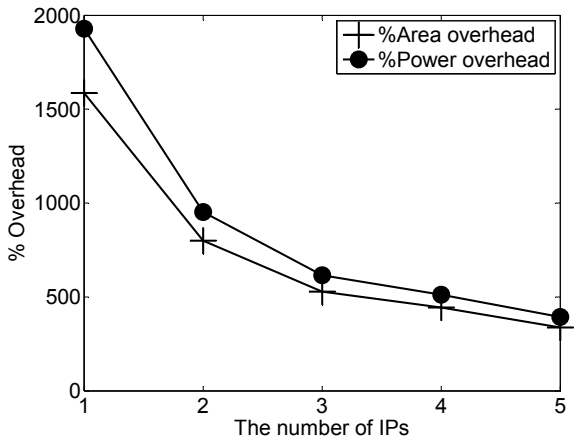

Figure 8: The change of the overhead with increasing the number of IPs sharing the PUF.

challenge bits and 16 response bits since we share the selector inputs that are below each other to keep the number of circuit inputs low. If we include the PUF in the overhead calculation, the overhead would be large. Note that the MCNC benchmarks are only control circuits and they do not include memory and I/O periphery/interfaces that are the area/power consuming components. Thus, the percentage of the added circuitry's overhead is much smaller than demonstrated.

Table 2 shows the overhead for integrating five benchmark circuits randomly selected from Table 1. It can be seen that the overheads without adding the PUF are almost constant. However, since the PUF's overhead is much larger than the FSM's overhead, adding the value of the overhead of the PUF to the system causes the overhead to decrease as we increase the number of the IPs sharing the PUF. Figure 8 shows the decrease of the overhead as we increase the number of IPs sharing the PUF.

8. APPLICATIONS

The ability to uniquely identify each copy of an IP in a design-reuse paradigm enables a range of new applications, inluding:

Protection against foundry overbuilding. The IP control and CFSM control methods eliminate the possibility of overbuilding and hence prevent piracy by requiring the consent of the original designer and IP providers for enabling/disabling of their cores.

Protection against licensed designers' overuse. A reuser may utilize a singly licensed core in multiple designs. Detection of misused IP cores in a large design is a very hard problem. With the new method, no IP will be activated without the consent of its original designer.

Interval licensing by remote enabling/disabling of IPs. Runtime disabling/enabling of IPs can be done since the chips that contain the IPs are identified and can be detected online. A possible application is interval licensing, where the product royalty must be frequently paid for continuous usage of the IP; otherwise, the IP is disabled.

Software/content metering. The unique IP identifiers can be further exploited for controlling the software and content running on the hardware.

Ownership proof. The original key for operating an IP core is given only for one set of PUF responses. A way to prove the ownership of the IC is to change the challenge inputs and then ask the designer to provide a new key which renders this device operational. The designer who has the full information of the STG can easily provide the new key, but other entities cannot. Thus, the IP rights owners can assert their ownership by online checking and authentication.

Multiple levels of protection. The approach introduces symmetry to the current asymmetric business model. Not only the reuser, but also the IP designer and the fab are protected by the symmetry. In addition to preventing piracy, the false accusations of overbuilding or overuse are prevented.

Enabling pay-as-you-configure method for the reuser. The chip designer embeds its locks in the functionality of the IP cores. The reuser can design its chips such that the IPs that provide additional functionality are disabled. Only the customers who pay the proper fees may enable those IPs.

Support for the design reuse paradigm. One of the greatest challenges in reuse-based design is protection of the rights of the IP owners. Since the proposed method targets digital rights management of IPs, it supports the design reuse paradigm that is essential to the development and evolvement of the modern designs and semiconductor industry [3].

9. CONCLUSION

We introduced the first approach, architecture and implementation for actively and uniquely controlling the functionality of each IP, in a multi IP core design and reuse paradigm. The approach protects the rights of the IP core owners, reusers, and the foundry by introducing a key exchange mechanism. The IC and each of its embedded IP cores are uniquely locked upon manufacturing. The method enables the designers and reusers to actively and remotely lock/unlock their IPs on each of the ICs post-manufacturing. We discussed a number of possible attacks, and provided countermeasures against them. Experimental evaluations on standard benchmark circuits demonstrate the low overhead and the applicability of the approach on industrial-strength designs. We introduced a number of newly enabled applications in protection, DRM, and security of the IP cores.

Acknowledgment

This work is supported by the Defense Advanced Research Projects Agency (DARPA)/MTO Trust in Integrated Circuits and Young Faculty Awards (YFA) under grant award W911NF-07-1-0198 and NSF CT-0716674.

C#	circuit	PI	states	area	delay	power	area	%	delay	%	power	%
1	planet	7	48	888	186.2	3,087	1752	97	70.2	-62.3	6428	108.2
2	s510	19	47	605	47.6	2,280	1426	136	49.9	4.8	4555.8	99.8
3	s1494	8	48	859	115.6	2,958	1746	103	65.8	-43.1	5178.8	75.1
4	s1488	8	48	880	134.9	3,011	2045	132	68.1	-49.5	6008.8	99.6
5	s298	3	135	2,951	201.5	10,798	5960	102	136.8	-32.1	22358.8	107.1
6	dk16	2	27	460	104.7	1,662	1970	328	49.3	-52.9	5886.8	254.2
7	sand	11	32	1,092	74.8	3,917	1092	0	59.6	-20.3	8584.8	119.2
8	styr	9	30	633	128.2	2,170	2180	244	52.7	-58.9	6218.8	186.6
	Mean						2271	143	69	-39	8,153	131

Table 1: The overhead of BFSM modifications for one IP.

10. REFERENCES

[1] "Certicom application note: Certicom security for fabless semiconductor design companies". http://www.certicom.com/download/aid-603/appnotes-fabless.pdf.

[2] "Design and reuse website", http://www.us.design-reuse.com/.

[3] "The international technology roadmap for semiconductors (itrs)", http://www.itrs.net/.

[4] "Defense science board (DSB) study on high performance microchip supply". http://www.acq.osd.mil/dsb/reports/2005-02-hpms_report_final.pdf, 2005.

[5] Y. Alkabani and F. Koushanfar. Active hardware metering for intellectual property protection and security. In *USENIX Security Symposium*, pages 291–306, 2007.

[6] Y. Alkabani, F. Koushanfar, N. Kiyavash, and M. Potkonjak. Trusted integrated circuits: A nondestructive hidden characteristics extraction approach. In *Information Hiding (IH)*, 2008.

[7] Y. Alkabani, F. Koushanfar, and M. Potkonjak. Remote activation of ICs for piracy prevention and digital right management. In *IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pages 674–677, 2007.

[8] Y. Alkabani, T. Massey, F. Koushanfar, and M. Potkonjak. Input vector control for post-silicon leakage current minimization in the presence of manufacturing variability. In *Design Automation Conference (DAC)*, 2008.

[9] R. Anderson. *Security Engineering: A guide to building dependable distributed systems*. John Wiley and Sons, 2001.

[10] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas. Silicon physical random functions. In *ACM conference on Computer and communications security (CCS)*, pages 148–160, 2002.

[11] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas. Delay-based circuit authentication and applications. In *ACM symposium on Applied computing*, pages 294–301, 2003.

[12] J. Guajardo, S. Kumar, G. Schrijen, and P. Tuyls. FPGA intrinsic PUFs and their use for IP protection. In *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, 2007.

[13] J. Hennessy and D. Patterson. *Computer architecture: a quantitative approach*. Morgan Kaufmann Publishers, 1996.

[14] M. Jacome and H. Peixoto. A survey of digital design reuse. *IEEE Design and Test of Computers*, 18(3):98–107, 2001.

[15] J. Lee, L. Daihyun, B. Gassend, G. Suh, M. van Dijk, and S. Devadas. A technique to build a secret key in integrated circuits for identification and authentication applications. In *Symposium of VLSI Circuits*, pages 176–179, 2004.

[16] K. Lofstrom, W. Daasch, and D. Taylor. IC identification circuits using device mismatch. In *International Solid State Circuits Conference (ISSCC)*, pages 372–373, 2000.

[17] M. Majzoobi, F. Koushanfar, and M. Potkonjak. Lightweight secure PUF. In *International conference on computer-aided design (ICCAD)*, 2008.

[18] M. Majzoobi, F. Koushanfar, and M. Potkonjak. Testing techniques for hardware security. In *ITC*, 2008.

[19] C. Marsh and T. Kean. A security tagging scheme for asic designs and intellectual property cores. *Design & Reuse*, January 2007.

[20] A. Oliveira. Techniques for the creation of digital watermarks in sequential circuit designs. *IEEE Trans. CAD of Integrated Circuits and Systems*, 20(9):1101–1117, 2001.

[21] G. Qu and M. Potkonjak. *Intellectual Property Protection in* VLSI *Design*. Kluwer Academic Publisher, 2003.

[22] J. Rowson and A. Sangiovanni-Vincentelli. Interface-based design. In *Design Automation Conference (DAC)*, pages 178–183, 1997.

[23] J. Roy, F. Koushanfar, and I. Markov. EPIC: Ending piracy of integrated circuits. In *Design Automation and Test in Europe (DATE)*, 2008.

[24] S. Trimberger. Trusted design in FPGAs. In *Design Automation Conference*, pages 5–8, 2007.

[25] T. Wollinger, J. Guajardo, and C. Paar. Security on FPGAs: State-of-the-art implementations and attacks. *IEEE Trans. on Embedded Computing Systems*, 3(3):534–574, 2004.

[26] L. Yuan and G. Qu. Information hiding in finite state machine. In *Information Hiding Workshop*, pages 340–354, 2004.