ON-THE-FLY ATTESTATION OF RECONFIGURABLE HARDWARE

Ricardo Chaves[1,2], Georgi Kuzmanov[2], Leonel Sousa[1]

[1] Instituto Superior Técnico/INESC-ID.
Rua Alves Redol 9, 1000-029 Lisbon,
Portugal. http://sips.inesc-id.pt/
email: {rjfc, las}@inesc-id.pt

[2]Computer Engineering Lab, TUDelft.
Postbus 5031, 2600 GA Delft,
The Netherlands. http://ce.et.tudelft.nl/
G.Kuzmanov@ewi.tudelft.nl

ABSTRACT

This paper presents a novel method to perform on-the-fly attestation of hardware structures loaded to reconfigurable devices. Given that a loadable hardware structure to a reconfigurable device is described by a binary bitstream, the hash value of this bitstream can be calculated to validate the hardware structure. To optimize this attestation, the hash value computation is implemented in hardware on the FPGA itself. To guarantee the integrity of the existing computation architecture, the proposed hardware module also enforces region delimitation. With the region delimitation, only the regions intended to be reconfigured can be modified. Implementation results suggest that this bitstream attestation can be performed without imposing an extra delay to the reconfigurable process and at an area cost of less that 10% of a Virtex II Pro 30 FPGA device.

1. INTRODUCTION

Recent FGPA devices allow its partial reconfiguration, while the rest remains active and working; this is called dynamic partial reconfiguration. This functionality allows for a very wide range of computational units to be used on a single device. This capability allows the use of smaller devices, that are cheaper and with a lower static power consumption, and the implementation of more complex computational structures that would otherwise not fit in the device. In several applications, the FPGA configuration data cannot be considered trustworthy, since it might be stored on an unsecured or unreliable location. With this fact comes the need to assure that whatever is being loaded into the reconfigurable device, is in fact being loaded into the intended location, and that once loaded it will behave as expected. Whenever a given computational core is required, the reconfiguration bitstream has to be downloaded from a storage device, such as a hard drive or a LAN, and uploaded to the FPGA de-

vice. However, a corrupted bitstream may reconfigure an unintended region of the device or implement an undesired functionality. For example, considering an FPGA system that controls a vehicle's Electronic Control Unit (ECU), the parking radar system can be replaced by the cruise control system when this is activated. However, we must be assured that the cruise control hardware is properly loaded, and that the remaining hardware is not altered, e.g., the system that controls the brakes.

This paper is focused on the attestation of the hardware structures loaded during the partial dynamic reconfiguration. The attestation module herein proposed is used not only to validate the correctness of the configuration data, but also to protect the remaining computational structures already loaded into the device. Taking into account that a hardware structure for a given reconfigurable device can be described by data in binary format, the verification of the hardware structure can be directly performed over the reconfiguration bitstream data. In this case, the proof that the reconfiguration bitstream has not been tampered with and that it is in fact the desired bitstream (i.e. the intended hardware structure), is achieved by computing and comparing the hash value of the reconfiguration bitstream. The hash value of a message, or in this case the reconfiguration bitstream, produces an unique identifying footprint of the processed data. In these algorithms the probability of two different input data streams generating the same hash value is very low, even if this is done intentionally in order to forge the signature of a given data stream.

Implementation results on a VIRTEX II Pro 30 FPGA suggest that:

- on-the-fly attestation of the reconfigurations bitstreams can be performed with no performance degradation.

- a device occupation of less than 10% using a SHA256 hash core is achieved.

- the bitstream attestation module can be easily integrated into existing designs.

This paper is organized as follows. Section 2 describes how dynamic reconfiguration on FPGAs is performed, in

---

particular for the Virtex II Pro Xilinx technology, and the main characteristics of the configuration bitstream. In Section 3, the hash functions are described, in particular the SHA algorithm. Section 4 describes the proposed attestation module, detailing the two major operations performed by this module, namely the region delimitation hardware and the hash generation of the bitstream. Implementation results are presented in Section 5 and Section 6 finalizes this paper with some concluding remarks.

2. FPGA DYNAMIC RECONFIGURATION

The current generation of reconfigurable devices has the capability to reconfigure part of its available resources while, at the same time, the remaining resources of the device continue active and performing computation. This operation is called dynamic partial reconfigurability. This type of reconfigurability allows for runtime reconfiguration, adaptive hardware algorithms, reduced power consumption, and a more efficient usage of the available resources.

The two main methods for partial reconfiguration are Difference-based and Module-based. In Difference-based partial reconfiguration, small changes to a design are supported by generating a bitstream based on only the differences in the two designs. In this paper only the Module-Based Partial Reconfiguration is considered, since this is the most useful for computational units' reconfiguration. In this reconfiguration mode, a fraction of the FPGA is completely reconfigured.

The reconfiguration of the Xilinx Virtex II Pro FPGA is performed by the Internal Configuration Access Port (ICAP), whose interface is depicted in Figure 1. The FPGA is re-
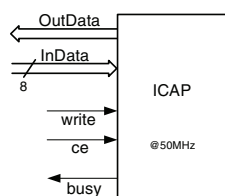

Fig. 1: Xilinx ICAP interface.

configured by sending the configuration bitstream through the 8 bit InData port. The configuration interface operates at 50MHz, and for a Virtex II PRO 30 a full configuration bitstream has approximately 1.7 Mbyte; the reconfiguration of the whole device requires approximately 34ms. On the Virtex II PRO technology, the data are loaded on a column basis with the smallest load unit being a frame, which varies in size depending on the targeted device [1].

The Virtex Pro devices have an on-chip DES (or Triple-DES) decryption core, used merely to decrypt the incoming bitstreams. The designer can encrypt the bitstream in soft-

ware, and the FPGA then performs the reverse operation, decrypting the incoming bitstream, and internally recreating the intended configuration. This method provides a very high degree of design security. Without knowledge of the encryption/decryption key or keys, potential attackers cannot use the externally intercepted bitstream to analyze or to clone the design [2]. System manufacturers can be sure that their Virtex-II Pro implemented designs cannot be copied and reverse engineered. However, it is also not possible to analyze the bitstream, since it is decrypted in the ICAP. It is thus not possible to asses what is being sent into the FPGA, nor which regions are being modified.

The attestation module is used to enforce reliability in the reconfigured computational modules and to protect existing modules, initially loaded into the device. This attestation procedure is mainly composed by two mechanisms: Region Delimitation and Hardware Attestation. This work assumes that the initial configuration of the FPGA with the attestation module can be securely performed, e.g., from a local ROM with the initial bitstream.

The following gives an overview on how the device is configured and the structure of the configuration bitstream.

2.1. Virtex II Pro Configuration Registers

The Virtex configuration logic was designed so that an external source may have control over the configuration functions by accessing and loading addressed internal configuration registers. In Table 1, the most significant internal configuration registers are presented. The *Frame Address*

Table 1: Virtex II Pro internal configuration registers.

Register Name	Description
CMD	Command Register
FLR	Frame Length Register
FAR	Frame Address Register
FDRI	Frame Data Input Register
CRC	Cyclic Redundancy Check
MFWR	Multiple frame Write Register
KEY	Initial Key Address Register
CBC	Cipher Block Chaining Register

Register (FAR) indicates the frame, and the location in the frame where the 32-bit configuration data are to be written. The register is automatically incremented after writing each word. The size of a frame is specified in the *Frame Length Register (LFR)*. The *Command Register (CMD)* is used to execute commands on the device. Table 2 presents the most relevant commands available in the FPGA configuration. These commands are executed by loading the CMD register with the respective binary code.

72

Table 2: Virtex II Pro CMD Register commands.

Command	Action
RCRC	Reset CRC Register
SWITCH	Change clock frequency
WCFG	Write Configuration Data
RCFG	Read Configuration data
LFRM	Last Frame Write
SHUTDOWN	Begins Shutdown sequence
START	Activates the reconfigurable hardware
MFWR	Activate Multiple Frame Write mode

2.2. Bitstream Packets

The configuration bitstream is stored in a file, the bitstream file. Since this file may include a header with information not relevant for the configuration itself (e.g. the file creation date), a synchronization tag is inserted in the bitstream before the actual configuration data. The synchronization tag is also used to synchronize the ICAP with the beginning of the 32-bit packet, since data are sent to the ICAP interface in blocks of 8 bits. Consequently, no actual processing takes place until the synchronization tag is detected by the ICAP. After synchronization, all data (commands, configuration data, etc.) are encapsulated in packets.

Due to the way data is processed by the ICAP, a 32-bit data block is only interpreted after the subsequent 32-bit data block is received. This implies a configuration delay of at least 4 cycles, which allows time for the attestation module to suspend the reconfiguration process before the packet is processed by the ICAP.

3. HASH FUNCTIONS

Cyclic Redundancy Check (CRC) functions are capable of detecting errors or variations in data streams, however, this detection is rather limited. Several data streams produce the same output, and collisions can be easily created. On the other hand, hash functions have an extremely low collision probability and collision attacks are not feasible in practice.

Currently, the most commonly used hash functions are the MD5 and the Secure Hash Algorithm (SHA), with 128-bit to 512-bit output Digest Messages (DM). While for MD5, collision attacks are computationally feasible on a standard desktop computer, current SHA-1 attacks still require massive computational power [3], (around 2^{69} hash operations), making attacks unfeasible for the time being. The SHA-1 (or SHA128) produces a 160-bit DM (the output hash value) from the input message. The input data stream is divided into multiple input blocks of 512 bits each. These 512-bit blocks are split into 80x32-bit words, one 32-bit word for each of the 80 computational round of the SHA-1 algorithm.

In 2002, the SHA-2 [4] was released as an official stan-dard. The SHA-2 uses larger DM, from 224 bits to 512 bits, making it more resistant to possible attacks and allowing them to be used with larger data streams, up to 2^{128} bits in the case of SHA512. The SHA-2 with a 256 DM is designated as SHA256. In each round of the SHA256 algorithm, 512 bit input data block are mixed with the current state. Each SHA256 data block is processed in 64 rounds, after which the current value is added to the previous hash value. The final DM for a given data stream is given by the final hash value obtained after the last data block.

4. HARDWARE ATTESTATION

In order to assure that the loaded modules are the correct ones, an attestation method has to be applied when these units are dynamically allocated into the reconfigurable device. As previously explained, the hardware structure loaded into the device is described by a set of packets. Therefore, the validation of the bitstream data allows for the attestation of a given hardware structure. The bitstream can be validated by computing its hash value, either in software or with a dedicated hardware structure. The existing mechanisms for software attestation could be used [5, 6], however the size of the bitstream can be significantly large, e.g. 1.4Mbit for 10% of a Virtex II Pro 30 FPGA. Therefore, the computational time of this software attestation would also be significant. In software, the bitstream would first need to be loaded into an internal memory, validated by a hashing algorithm, and sent to the configuration interface. Even assuming an internal memory large enough to store the entire desired bitstream and that the ICAP is directly connected to the internal memory, a significant computational overhead would still be imposed.

A few hardware structures have also been proposed to perform authentication [7, 8]. However, these are either performed offline or require the validation to be performed before the reconfiguration process is started.

In this paper, a hardware structure is proposed to perform on-the-fly attestation of the reconfiguration bitstreams. In this approach, the generation of the DM is performed in the device, as the device is reconfigured. Given that the DM computation can be performed at a faster rate than the device reconfiguration, and that the data is sent directly to the ICAP, no additional time is required for the attestable reconfiguration, regarding the standard dynamic reconfiguration of the device. However, since the DM of the unit can only be obtained after the complete hash value is computed, some region delimitation has to be enforced, in order to guarantee the non tampering of the remaining area of the circuit. The region of the device allowed to be reconfigured is limited to the area reserved for the reconfigurable modules. If an attempt is made to modify (write) any of the reconfigurable hardware outside the delimited region, an abort signal is generated and the reconfiguration process stopped. The
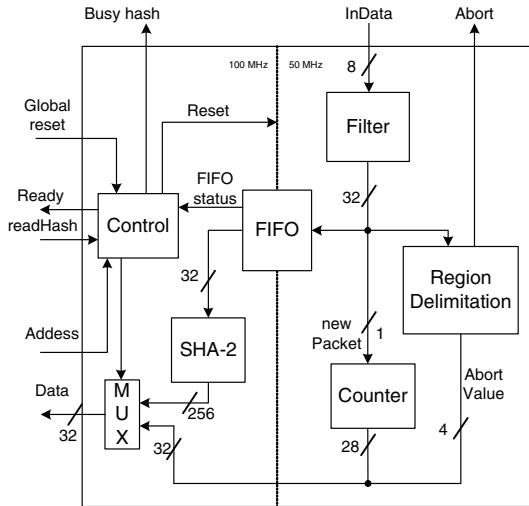
Authorized licensed use limited to: Technische Universiteit Delft. Downloaded on November 2, 2008 at 19:05 from IEEE Xplore. Restrictions apply.

Fig. 2: Attestation module structure.

following describes the hardware structures realized to implement the desired functionalities of the attestation module, namely: *i)* region delimitation; *ii)* hardware validation; and *iii)* interface to the attestation module. The structure of the attestation module is depicted in Figure 2.

4.1. Region Delimitation

To detect a write violation outside the delimited area, we must know: i) which frame is being written and ii) when is the frame being written. Since the ICAP does not directly provide this information, the bitstream must be interpreted. In order to assure that all the bitstream is analyzed, certain commands cannot be allowed to be processed by the ICAP, such as *Shutdown* or *Multiple Frame Write*.

To simplify the interpretation of the bitstream, the attestation module is organized into several units. First, the 8-bit input stream ($InData$), depicted in Figure 2, is filtered in order to eliminate all data before the synchronization sequence ($FF, FF, FF, FF, AA, 99, 55, 66$). This also identifies the beginning of the first 32-bit packet. After receiving 4 new 8-bit inputs, the 32-bit output is made available and the $NewPacket$ signal generated, indicating that a new packet is available.

The 32-bit packets are then sent to the *Region Delimitation* module that interprets the bitstream data. This parser mostly implements a state machine that identifies which kind of packet is received and their function.

Given that this unit has to identify where the Data Frame is being written to, writings to the Frame Length Register (FLR), Frame Address Register (FAR), and Frame Data Register Input (FDRI) have to be detected. The parser has its own internal frame register, which is updated every time a FAR packet is sent to the ICAP. The value written to the

FDRI is retrieved and used to calculate which frames will be written by the arriving data packets. To assure that a frame is not improperly written due to a wrong frame dimension value (FLR), the packet with the FLR value is compared with the correct value, which is known for the device in use. In this version of the attestation module, the FAR register is compared with a static region value. This defined a fixed reconfigurable region.

As mentioned above, it is not sufficient to test this type of instructions; command instructions also have to be interpreted, in order to assure a certifiable reconfiguration procedure. The following points out which commands are not allowed and why they cannot be executed (refer to Table 2 on page 3 for the commands).

SWITCH : By changing the clock frequency, an operating frequency could be set in which the full system could behave erroneously.

SHUTDOWN : While in shutdown mode, the configured logic in the device is rendered inoperable; consequently, during this period the Attestation Module would also be rendered inoperable.

MFWR : The implemented attestation module is only able to cope with one frame write at a time. This means that only one of the frame's writing would be within the Region Delimitation hardware testability. The other frame being written could be located anywhere within the device.

Whenever the internal frame address is outside the allowed region, or when an illegal command is detected, the *Abort* signal (see Figure 2) is generated and the device reconfiguration suspended. In the attestation module output, 4 bits are used to identify what kind of violation generated the *Abort* signal. Whenever a new packet is made available, an internal counter is incremented, indicating the number of packets received for a given bitstream. The 28-bit value of this counter is also outputted in order to be used in the validation test.

4.2. Hardware Validation

The hardware validation is performed by generating the Digest Message of the data being used to configure the device and comparing it with the expected value. The bitstream data preceding the synchronization pattern is not used to generate the hash value.

The SHA hash function is used to generate the DM. The SHA256 core used [9], computes the hashing of the bitstream in blocks of 512 bits and requires 65 clock cycles to compute each data block. The SHA256 core at 50MHz would only be able to compress data at a rate of 393Mbit/s, which is lower than the maximum input rate of the ICAP (of 400Mbit/s). Consequently, rather than having the ICAP

74

wait for the bitstream to be compressed, the hash core is operated at a higher frequency, for example at the frequency of the hardware interfacing the attestation module. This dual clock operation is depicted by the dotted line in Figure 2. A FIFO is used to interconnect the bitstream filter, running at the ICAP frequency, with the validation hardware. This FIFO, capable of receiving and sending data with different clocks, acts as a buffer between the two computation units. If the hashing core is not able to process the data at an adequate rate, the FIFO $full$ signal is used to halt the ICAP and the reconfiguration process, giving time for the core to process the data. If no data is available in the FIFO the hashing core is simply halted until a new 512-bit is available.

Instead of having the parser unit detect the end of the bitstream, the hash unit is always computing any incoming data. This is used as an additional security measure, since a new bitstream sequence can be added to the end of the configuration file sent to the ICAP. The attestation module only stops computing the hash value when it receives the signal to read the DM. When the hardware validation unit receives this $readHash$ signal, depicted in Figure 2, two situations can occur: *i)* the reconfiguration bitstream is a multiple of 512 bits and the current DM is the final DM value; *ii)* the reconfiguration bitstream is not a multiple of 512 bits. In this later case, the last partial input of the hashing core is concatenated with zeros, to form a full 512-bit block, and processed to generate the final DM. This concludes the computation of the attestation module, causing the $Ready$ signal to be asserted. Once the final DM is generated, the validation data can be read. These data are composed by the final DM; the dimension of loaded bitstream, given by the counter; and the error value.

To validate the loaded bitstream the DM read has to be compared with the expected value in order to determine the authenticity of the loaded hardware. This validation also checks if an error occurred, and the dimension of the bitstream.

The final DM is calculated without the concatenation of the length of the hashed message. The length of the bitstream is considered separately from the hash value. This increases the security level of the validation data, that is thus composed by the DM, the *Abort value* (if an *Abort* occurred), and the length of the loaded reconfiguration bitstream.

4.3. Attestation Module Interface

In order to facilitate the utilization of this attestation module, an interface identical to a register bank is used. 3 major port sets are used. *i)* The handshake port composed by the $readHash$ signal, that informs the attestation module that there is no more data to be hashed, and to conclude the computation; and the $Ready$ signal that indicates that the final DM is ready to be read. *ii)* A 32 bit bus to read the data and the address port, indicating which of the 32bit parcels of the validation data are to be read. *iii)* A reset signal ($Global reset$) used to reset the attestation module, preparing it to validate a new reconfiguration bitstream.

By reading the counter/abort value, a reset signal is sent to the region delimitation unit and packet counter. This reset initializes the part of the module running at 50 MHz, preparing it for the next bitstream. In order to assure that the part of the attestation module running at 50MHz is reset, the reset signal for these units is set active during the 8 cycles needed to read the final DM value. Once the DM value has been read, the hashing core is also reset, by the *Global reset* signal.

Given that the attestation module works passively by snooping the ICAP bus interface during the reconfiguration process, it can be easily integrated into existing designs. This module only requires that the reconfiguration unit, that sends the bitstream data to the ICAP, allows for the reconfiguration to be halted, in case the *Abort* signal is asserted. The data resulting from the attestation module is read by addressing the data as a register bank. With this interface the attestation module can be integrated into existing designs at a very low design cost.

5. IMPLEMENTATION RESULTS

The proposed attestation module was implemented on a Virtex II Pro 30 Xilinx FPGA, in order to analyze the amount of resources required to implement it. The results were obtained after place and route, considering the attestation module as an isolated component.

Implementation results suggest an occupation of approximately 8% of the FPGA (1113 slices and 2 BRAMs) for the complete attestation module supported by a SHA256 hash core. The FIFO in the attestation unit was realized using the Xilinx ISE IPCore generator. It requires one embedded BRAM and is capable of storing 128 words of 32 bits. This FIFO also outputs additional status information, e.g. 512 bits available, full and empty FIFO.

In order to analyze the cost of each of the functional blocks of the attestation module were implemented separately. Implementation results suggest that the bitstream interpretation and the region delimitation only require 195 slices, and 1 BRAM for the FIFO buffer. This suggests that an even more compact attestation module can be achieved with the use of a more area efficient hashing core. The SHA256 core used requires 849 slices and 1 BRAM, 6% of the available reconfigurable resources. More detailed figures for the FPGA occupation are presented in Table 3. The register based structure of the Filter and Region Delimitation components are capable of achieving a maximum frequency largely above 50MHz. The SHA256 core can be operated at frequencies up to 170 MHz.

With a SHA128 core, less than 800 slices are required,

Table 3: Attestation module occupation.

Component	Slices	BRAMs	Occupation
Filter	40	0	<1%
Reg. Delimit.	79	0	<1%
FIFO	0	1	-
SHA128	533	0	4%
SHA256	849	1	6%
Total-SHA128	797	1	6%
Total-SHA256	1113	2	8%

the occupation of the attestation core is already reduced to 6% of the total available resources of a Virtex II Pro 30.

6. CONCLUSIONS

In this paper, a novel method is proposed to validate the hardware structures loaded into reconfigurable device at run time, via dynamic partial reconfiguration. This method is based on the analysis of the bitstream used to configure the FPGAs. The attestation, of the computational structure being loaded, is performed by computing in hardware the hash value of the bitstream, and comparing it against the expected value. However, this operation is not sufficient to assure the correct loading of the desired structure; the validation of the loaded structure can only be performed after the bitstream has been completely uploaded into the FPGA. An additional mechanism has to be used in order to assure that an adulterated bitstream is not able to damage or modify the remaining computational structure. This is achieved by restricting the area where the reconfiguration can occur. An attempt to modify any resources outside the delimited area, leads to the halting of the reconfiguration. Since the attestation module works by snooping the ICAP bus interface, it can be easily integrated into existing designs at a very low design cost. Implementation results suggest that the proposed attestation module, using the SHA256 algorithm to produce the Digest Message, can be realized using less than 10% of the resources available in the Virtex II Pro 30 FPGA. If the SHA128 algorithm is used, 6% of the available resources are needed. With this attestation module, higher security in the dynamic partial reconfiguration of hardware structures can be accomplished with no performance degradation and at a low area cost. It allows, for example, the use of dynamic FPGA reconfiguration in the automotive industry [10], that strongly requires the certification of all used hardware and software components. In conclusion, the work herein proposed allows the on-the-fly attestation of incoming reconfiguration bit stream in existing FPGA devices with a low hardware cost. In future FPGA devices this attestation unit may be directly included in the device layout, no longer requiring reconfiguration resources.

7. REFERENCES

[1] *Two Flows for Partial Reconfiguration: Module Based or Difference Based*, Xapp290 (v1.2) ed., Xilinx, September 2004, Application Note: Virtex, Virtex-E, Virtex-II, Virtex-II Pro Families.

[2] *Virtex-II Pro and Virtex-II Pro X FPGA User Guide*, Ug012 (v4.1) ed., Xilinx, March 2007.

[3] X. Wang, Y. L. Yin, and H. Yu, "Finding collisions in the full SHA-1," in *CRYPTO*, ser. Lecture Notes in Computer Science, V. Shoup, Ed., vol. 3621. Springer, 2005, pp. 17–36.

[4] NIST, "FIPS 180-2, secure hash standard (SHS)," National Institute of Standards and Technology, Tech. Rep., August 2002.

[5] A. Seshadri, A. Perrig, L. van Doorn, and P. K. Khosla, "Swatt: Software-based attestation for embedded devices," in *IEEE Symposium on Security and Privacy*. IEEE Computer Society, May 2004, pp. 272–.

[6] E. Shi, A. Perrig, and L. van Doorn, "Bind: A fine-grained attestation service for secure distributed systems," in *IEEE Symposium on Security and Privacy*, May 2005.

[7] E. Simpson and P. Schaumont, "Offline hardware/software authentication for reconfigurable platforms," *Lecture Notes In Computer Science*, vol. 4249, p. 311, 2006.

[8] T. Eisenbarth, T. Güneysu, C. Paar, A. Sadeghi, D. Schellekens, and M. Wolf, "Reconfigurable trusted computing in hardware," *Proceedings of the 2007 ACM workshop on Scalable trusted computing*, pp. 15–20, 2007.

[9] R. Chaves, G. Kuzmanov, L. A. Sousa, and S. Vassiliadis, "Improving SHA-2 hardware implementations," in *Workshop on Cryptographic Hardware and Embedded Systems, CHES 2006*, October 2006, pp. 298–310.

[10] N. Chujo, "Fail-safe ECU System Using Dynamic Reconfiguration of FPGA," *R&D Review of Toyota CRDL*, vol. 37, no. 2, pp. 54–60, 2002.