ELectronics EXpress

LETTER

IEICE Electronics Express, Vol.9, No.22, 1735-1741

Efficient verification of IP watermarks in FPGA designs through lookup table content extracting

Jiliang Zhang[1a)], **Yaping Lin**[1b)], **Wenjie Che**[1], **Qiang Wu**[1], **Yongqiang Lu**[2], **and Kang Zhao**[3]

[1] *College of Information Science and Engineering, Hunan University, Changsha, China*

[2] *Research Institute of Information Technology, Tsinghua University, Beijing, China*

[3] *Intel Corporation, Beijing, China*

a) *hnu.zjl@gmail.com*

b) *yplin@hnu.edu.cn*

Abstract: Digital watermarking is an innovative technique for intellectual property protection (IPP) of Field Programmable Gate Array (FPGA) designs. However, many of these techniques usually need manually extract marks from binary bit-files by the FPGA tool or exhaustive search to find out marks in the design, which results in inefficiency of the watermark verification. This paper presents a method to fast verify the authorship through extracting the content of the watermarked lookup tables from a binary bit-file. We demonstrate the proposed method on several Xilinx Virtex-II devices, and experimental results on the watermarked designs from the IWLS 2005 benchmarks show that the verification of authorship has high efficiency.

Keywords: watermarking, intellectual property protection (IPP), field programmable gate array (FPGA)

Classification: Integrated circuits

References

[1] A. B. Kahng, et al., "Constraint-based watermarking techniques for design IP protection," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 20, no. 10, pp. 1236–1252, Oct. 2001.

[2] M. Fallahpour and D. Megias, "High capacity audio watermarking using FFT amplitude interpolation," *IEICE Electron. Express*, vol. 6, no. 14, pp. 1057–1063, July 2009.

[3] J. Lach, W. H. Mangione-Smith, and M. Potkonjak, "Fingerprinting techniques for field-programmable gate array intellectual property protection," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 20, no. 10, pp. 1253–1261, Oct. 2001.

[4] J. Lach, W. H. Mangione-Smith, and M. Potkonjak, "Signature hiding techniques for FPGA intellectual property protection," *Proc. 17th IEEE/ACM Int. Conf. Computer-Aided Design*, San Jose, USA, pp. 186–189, Nov. 1998.

[5] J. Lach, W. H. Mangione-Smith, and M. Potkonjak, "Robust FPGA

DOI: 10.1587/elex.9.1735
Received October 07, 2012
Accepted October 30, 2012
Published November 26, 2012

intellectual property protection through multiple small watermarks," *Proc. 36th Design Automatic Conf.*, New Orleans, USA, pp. 831–836, June 1999.

[6] D. Saha and S. Sur-Kolay, "Secure Public Verification of IP Marks in FPGA Design through a Zero-Knowledge Protocol," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20, no. 10, pp. 1749–1757, Oct. 2012.

[7] J. Zhang, Y. Lin, Q. Wu, and W. Che, "Watermarking FPGA Bitfile for Intellectual Property Protection," *Radioengineering*, vol. 21, no 2, pp. 764–771, June 2012.

[8] D. Ziener, S. Assmus, and J. Teich, "Identifying FPGA IP Cores based on Lookup Table Content Analysis," *Proc. 16th Int. Conf. Field Programmable Logic and Applications*, Madrid, Spain, pp. 1–6, Aug. 2006.

[9] K. Yip and T. Ng, "Partial-encryption technique for intellectual property protection of FPGA-based products," *IEEE Trans. Consum. Electron.*, vol. 46, no. 1, pp. 183–190, Feb. 2000.

[10] Xilinx Inc. Virtex-II Platform FPGA User Guide UG002. V2.2, 2007, pp. 273–360.

[11] T. Nie and M. Toyonaga, "An Efficient and Reliable Watermarking System for IP Protection," *IEICE Trans. Fundamentals*, vol. E90-A, no. 9, pp. 1932–1939, Sept. 2007.

1 Introduction

As reuse-based design methodology has prevailed in integrated circuit (IC) design field, the IC design industry is confronted with the increasing threat of intellectual property (IP) infringement which results in loss of revenue and market share. Hence, how to protect reusable IP cores effectively has become a serious problem. Watermarking is a comprehensive mechanism implemented to protect IP from illegal reuse in many fields, such as IC [1, 11] and audio [2]. In the open literatures, watermarking procedures for protecting IP cores can be roughly categorized into the constraint based methods and the additive methods.

The constraint based methods usually introduce additional constraints to yield the new watermarked design during the design procedure [1]. The major drawback of these techniques is the limitation on verification possibilities of the watermarked core. Hence, it is difficult to detect the watermark from the bit-file. The additive watermarking methods embed the owner's watermark into unused slices (i.e. non-functional slices) at physical layout level [4, 5] or embed the IP buyer's fingerprint together with the owner's watermark into unused slices to create a buyer-specific instance [3]. In 2012, D. Saha *et al.* [6] proposed a significant zero-knowledge protocol *Verify_ZKP* to ensure trustworthy yet leakage-proof public verification based on the marks hidden in unused CLBs to protect FPGA IP cores. Although [4, 5, 6] can extract the watermark in all its posterior design levels, they require manually extracting marks from the bit-file core by FPGA tools or exhaustive search to find out marks in the design to verify the authorship. Hence, the process of watermarking verification is inefficient.

In this paper, an efficient watermarking verification method is proposed to extract the watermark fast from the bit-stream file through lookup table content analysis so that the authorship can be fast and effectively proven.

DOI: 10.1587/elex.9.1735
Received October 07, 2012
Accepted October 30, 2012
Published November 26, 2012

Besides, our watermarking scheme has no traceability issue since the watermark is embedded into the FPGA bit-file.

2 The proposed watermarking method

2.1 Watermark embedding

The watermark preparation algorithm uses a secure hash function to reduce the length of signature S, and then the hashed signature is encrypted by a public key algorithm with the key K to generate the watermark W. Finally, W is embedded into the design I in the position R by creating a watermarked work $\urcorner I$. If the target FPGA of the design can store t-bit configuration string in each LUT, then W is divided into l segments (each segment stores t-bit mark). The value of t is 16 on Xilinx Virtex-II FPGAs, i. e., each ILUT accommodates 16 bit marks.

To embed the watermark, we need to get the information about the content of the watermark and the embedding positions. The content of the watermark is generated in the watermark preparation phase. The specific position where w_i is embedded is determined by a pseudo random sequence R, which is generated by a pseudo random sequence generator. The process of watermark embedding is the same as [6]. The main difference is that our method embeds marks into ILUTs [7] at bit-file level instead of non-functional slices at physical layout level.

2.2 Watermark verification

In this section, we discuss how to fast extract the content of watermarked ILUTs from a binary configuration bit-stream file. Firstly, we obtain the configuration bit-file of the FPGA in the design. Secondly, according to the method of extracting the content of LUTs from a bit-file [8], we show our method to compute the byte address of the watermarked ILUTs in the configuration packet of the bit-stream, and then we extract the content of the watermarked ILUTs to acquire the embedded watermark. Finally, decrypt W to provide authorship proof.

To control unauthorized access, the configuration bit-stream in the bit-file of FPGA designs may remain in encrypted form, but it is still possible to extract watermarks from it using the proposed approach. The encrypted bit-stream is loaded into FPGA and decrypted by the decryption unit in FPGA, and the bit-stream can be attained by monitoring the serial line between the FPGA and the bit-stream storing EPROM during configuration [5, 8]. Then the unencrypted bit-file is available and our proposed method is applied on it to extract watermarks. If partial encryption [9] is applied to generate the encrypted file where only the bit-stream except the marks is encrypted, our method can be directly applied to it.

We have experimented with the proposed method on Xilinx Virtex-II and Virtex-II Pro FPGA bit-files. A bit-stream file is structured in packets. One of these packets includes the configuration content, which are divided into frames. These frames are the smallest addressable segments of the Virtex-II configuration memory space. Virtex-II devices are configured by loading application-specific configuration data into internal memory. Configuration frames are grouped into six column types that correspond roughly to physical device resources. In all Virtex-II devices, there are the same configuration column types: IOB, IOI, CLB, GCLK, BRAM, and

DOI: 10.1587/elex.9.1735
Received October 07, 2012
Accepted October 30, 2012
Published November 26, 2012

BRAM Interconnect. In each configuration frame, there is a unique 32-bit address that is composed of a block address (BA), a major address (MJA), a minor address (MNA) and a byte number. Block Address 0 contains all GCLK, IOB, IOI, and CLB configuration columns [10]. The major address identifies a specific column within a block, and the minor address identifies a specific frame within a column.

All Virtex-II devices have the same number of IOB, IOI, and GCLK columns. The number of columns and frames per column type for several Virtex-II devices is given in [10]. Figure 1 depicts the CLB frame columns in the Packet of a Virtex-II FPGA's Bit-file. #CLB is the number of CLB columns. The two specific frames (frame 2 and frame 3) in each of the CLB column are used for storing the content of LUTs. Figure 2 clearly shows the position arrangement of Slices in the i-th CLB column for a Virtex-II FPGA. The correction between Figure 1 and Figure 2 is that the frame 2 (denoted by the blue rectangle) of the i-th CLB frame column in Figure 1 is used to store the content of LUTs of the left slice column in Figure 2, and the frame 3 (denoted by the orange rectangle) is for that of the right slice column in Figure 2. S_r is the number of slice rows in the FPGA.

The positions of LUTs in a frame are shown in Figure 3. The LUT content for one slice consists of 5 bytes, including two bytes G-LUT, two bytes F-LUT and a separate byte. The G-LUT bit order is reversed, while the F-LUT bit order is not.

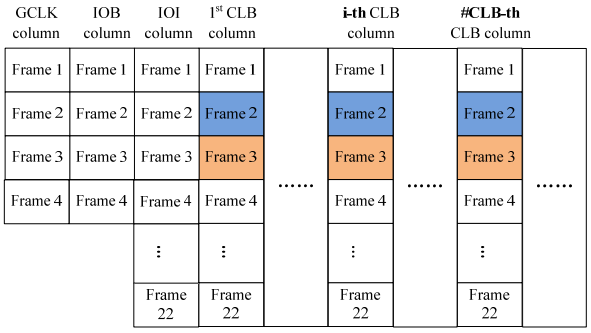

Fig. 1. CLB frame columns in the Packet of a Virtex-II FPGA's Bit-file

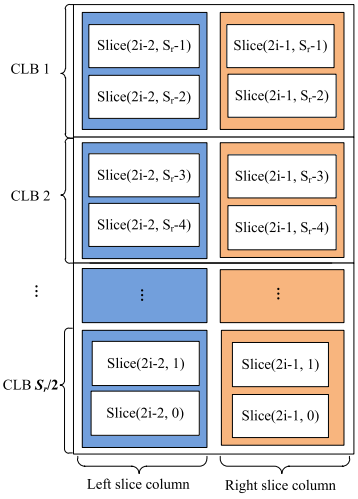

Fig. 2. The position arrangement of Slices in the i-th CLB column for a Virtex-II FPGA

DOI: 10.1587/elex.9.1735
Received October 07, 2012
Accepted October 30, 2012
Published November 26, 2012

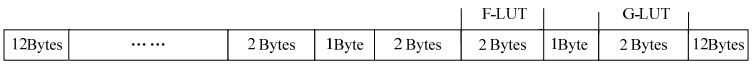

Fig. 3. The positions of LUTs in the third frame of the i-th CLB column

According to the structure of the bit-stream file described above, the calculation method for the byte address of the G-LUT in the slice (x, y) is given in Equation (1).

$$Addr_{G-LUT}(x, y) = N_{offset}(y) + N_{Frames}(x) \times F_{Len} \tag{1}$$

where, $N_{offset}(y)$ denotes the byte offsets from the beginning of the target frame, as shown in Equation (2). The target frame is defined as the frame that includes all the content of the LUTs in slice column x. $N_{Frames}(x)$ indicates the total number of frames before the target frame, as shown in Equation (3). F_{Len} denotes the frame length, which is given in [10].

$$N_{offset}(y) = 12 + ((S_r - 1) - y) \times 5 \tag{2}$$

$$\begin{aligned} N_{Frames}(x) &= 1 \times N_{GCLK} + 1 \times N_{IOB} + 1 \times N_{IOI} + CF_{LUT0} + \lfloor x/2 \rfloor \times N_{CLB} + x \bmod 2 \\ &= 4 + 4 + 22 + 1 + \lfloor x/2 \rfloor \times 22 + x \bmod 2 \\ &= 31 + \lfloor x/2 \rfloor \times 22 + x \bmod 2 \end{aligned} \tag{3}$$

where, N_{GCLK}, N_{IOB}, N_{IOI}, and N_{CLB} denote the number of frames of each GCLK, IOB, IOI and CLB column respectively. CF_{LUT0} denotes the first frame in the CLB column, thus the value of CF_{LUT0} is 1. $(x \bmod 2)$ indicates that the LUT content of the left slice column (x is even) is stored in the frame 2, while that of the right slice column (x is odd) stored in the frame 3.

Combining (1), (2) and (3), we can use (4) to calculate the byte address of G-LUTs in the configuration packet of the bit-stream.

$$Addr_{G-LUT}(x, y) = (31 + \lfloor x/2 \rfloor \times 22 + x \bmod 2) \times F_{Len} + 12 + ((S_r - 1) - y) \times 5 \tag{4}$$

As shown in Figure 3, the byte address of F-LUT is three bytes more offsets than G-LUT in the same slice. Hence,

$$Addr_{F-LUT}(x, y) = Addr_{G-LUT}(x, y) + 3 \tag{5}$$

The position set of the watermarked ILUTs R is recorded in the watermark embedding process (section 2.1). The position of the i-th watermarked ILUT $r_i = (x_i, y_i, z)$, where $z = 0$ or 1, (x_i, y_i) denotes the coordinate of watermarked slice which the i-th watermarked ILUT consists in. If $z = 0$, G-LUT is watermarked, else FLUT is watermarked. With r_i, we can calculate the addresses of the i-th watermarked ILUT in $\neg B$ (the watermarked bit-file) by (4) or (5), then the watermark can be extracted and decoded from the bit-stream easily according to the byte addresses of the watermarked ILUTs.

The watermark extracting algorithm is shown in Figure 4. The time complexity is $O(l)$, where l is the number of watermarked LUTs. For each watermarked ILUT, marks are extracted in steps 2 ~ 15. Finally, we recombine all the extracted marks into W (step 16), and then decode W into S (step 17), which can provide the authorship proof. When the IP vendors suspect that their IP has been infringed, they can apply for a neutral third-party organization to extract the watermark from the IP core.

DOI: 10.1587/elex.9.1735
Received October 07, 2012
Accepted October 30, 2012
Published November 26, 2012

```
1   BitV (K,¬B, l, R) {
2       for (i = 0; i < l; i ++){
3           get r_i from R;
4           if ( z = 1 ) {
5           Compute: Addr_F-LUT (x_i, y_i);
6           W_F-LUT = Ext (Addr_F-LUT (x_i, y_i), ¬B);
7           Store W_F-LUT into the set of WF;
8           }
9            if ( z = 0 ) {
10           Compute: Addr_G-LUT (x_i, y_i)
11           temp =Ext (Addr_G-LUT (x_i, y_i) , ¬B);
12          W_G-LUT = Reverse(temp);
13           Store W_G-LUT into the set of WG;
14           }
15       }
16       WB = Recombine(WG, WF);
17       S = Dec(K, WB);
18  }
```

Fig. 4. The watermark extracting algorithm

3 Experimental results

We have performed a set of experiments to evaluate the effectiveness on Xilinx Virtex-II FPGAs with 1024 bits watermarks. Test circuits come from the IWLS 2005 benchmarks. Each circuit is synthesized and implemented on the smallest possible FPGA device that can accommodate the circuit along with the watermark. The watermark extracting algorithm is implemented in C on a 3.0 GHz Intel® Pentium® E5700 machine with Windows XP OS.

The experimental results in Table I and Table II show that our watermarking embedding method introduces zero area and timing overhead. Area overhead is measured by the added Slice. ILUTs are used to embed marks and consist in functional slices which have been used in the FPGA design. Additionally, these ILUTs have no function and have been reported "used" before watermarking in the *Place and Route Report*, which is generated by Xilinx ISE tool [7]. Hence, there is essentially no area overhead required. Timing overhead is measured by the Minimum Period (MP) degradation. Timing Analyzer (Xilinx tool) is used to analyze minimum period. As shown in Table II, minimum period is unchanged, so the method has zero timing overhead. However, as shown in Table III, some of the "setup to clock clk" of clk in the design are affected after embedding 256 bits marks, so our method has routing overhead. This can be explained that we change part of the net connection of adjacent functional LUTs since more interconnection is added between watermarked ILUTs and don't care inputs of functional LUTs, which slightly impacts to routing of the original design.

As shown in Table IV, the CPU time requirements of the watermark extracting remain in the range from 0.031 to 0.078 s for the chosen benchmark.

DOI: 10.1587/elex.9.1735
Received October 07, 2012
Accepted October 30, 2012
Published November 26, 2012

Table I. Area overhead comparison with [6]

Circuits	FPGA	Original Slice	Method in [6]		ours	
			Added Slice	OH (%)	Added Slice	OH (%)
spi	XC2V250	396	32	8.081	0	0
vga_lcd	XC2V1500	828	32	3.865	0	0
b14	XC2V250	1231	32	2.600	0	0
pci_bridge	XC2V1500	1395	32	2.294	0	0
S38584	XC2V1500	1626	32	1.968	0	0
Average				3.762		0

Table II. Timing overhead comparison with [6]

Circuits	FPGA	Original MP	Method in [6]		ours	
			Added MP(ns)	OH(%)	Added MP(ns)	OH(%)
spi	XC2V250	10.197	0.012	0.12	0	0
vga_lcd	XC2V1500	6.027	-0.028	-0.46	0	0
b14	XC2V250	17.123	-1.913	-11.17	0	0
pci_bridge	XC2V1500	8.874	0.205	2.31	0	0
S38584	XC2V1500	5.585	0.591	10.58	0	0

Table III. Impacts to routing: The change of "setup to clock clk" (IP core: des_perf)

Source	key1<5>	key1<6>	key1<7>	Key2<25>	Key3<55>
Original	3.106	3.721	3.069	1.657	3.058
256 bits	3.137	3.782	3.070	1.676	3.091

Table IV. The CPU time requirements of the watermark extracting algorithm (IP core: des_perf)

#Marks	64bit	256-bit	512-bit	1024-bit	2048-bit
CPU(s)	0.031	0.032	0.032	0.047	0.078

Watermarking methods that embedding marks into unused slices (i.e. non-functional slices) have been proven robust in [5, 6]. Hence, our method has the same security level with them since our method embeds marks into ILUTs instead of non-functional slices.

5 Conclusion

In this paper, we have presented a method to efficiently extract the watermark from binary bit-files to address the issue of inefficacy of verification. The extraction of the embedded watermark was demonstrated on Xilinx Virtex-II FPGAs. For other FPGA devices or vendors we expect that similar techniques can be used to find rules of how to extract the embedded watermark. The experimental results showed that the process of watermark extracting is fast and does not require any extra hardware.

Acknowledgments

This work was supported by the Postgraduate Research and Innovation Project of Hunan Province of China under grant No CX2012B142. National Natural Science Foundation of China under Grant No 60973031, 61173038 61106030 and 61202462. National Science and Technology Major Project of the Ministry of Science and Technology of China under Grant No 2013ZX01039001-002-003.

DOI: 10.1587/elex.9.1735
Received October 07, 2012
Accepted October 30, 2012
Published November 26, 2012