

Fast Reverse Engineering for Xilinx FPGAs

Soyeon Choi
Dept. of Semiconductor Systems
Engineering
Hanbat National University
Daejeon, South Korea
sychoi@hanbat.ac.kr

Yujin Eom
Dept. of Electronics Engineering
Chungnam National University
Daejeon, South Korea
yjeom.cas@gmail.com

Yunjin Noh
Dept. of Electronics Engineering
Chungnam National University
Daejeon, South Korea
yjnoh.cas@gmail.com

Yoonjeong Kim
Dept. of Electronics Engineering
Chungnam National University
Daejeon, South Korea
yjkim.cas@gmail.com

Hoyoung Yoo
Dept. of Electronics Engineering
Chungnam National University
Daejeon, South Korea
hyyoo@cnu.ac.kr

Abstract— In this paper, we propose a method to reduce the time required to obtain the mapping tables for PLP and PIP by minimizing the number of manually created mapping tables and utilizing an offset calculation formula. The previous work requires creating mapping tables for every tile manually, necessitating hundreds of days even for the smallest Spartan-3 S50 device. This paper analyzes the FPGA and bitstream structure and evaluates the similarity between mapping tables based on these analyses. A method is proposed in this paper that creates only the reference mapping table manually and utilizes offset calculation formulas to obtain mapping tables for reverse engineering the entire FPGA. Experiments using the S50 device showed that all mapping tables could be generated in about 21 days, which is approximately 20 times faster than the previous method. The results show that mapping tables can be acquired within a practical timeframe, which makes reverse-engineering attacks feasible. Consequently, it is important to develop and deploy effective countermeasures for systems using FPGAs.

Keywords—Reverse Engineering, PLP, PIP, Xilinx, FPGA

I. INTRODUCTION

The most widely used SRAM-based FPGAs require configuration data to be loaded from external memory at power-up, and this configuration phase is particularly vulnerable to attacks [1]. To counter FPGA attacks using reverse engineering, several studies have been conducted to verify the feasibility of FPGA reverse engineering [2–4]. Among these, Ding’s team proposed reverse engineering of PLPs (Programmable Logic Points) and PIPs (Programmable Interconnect Points) using mapping tables. They demonstrate that this method yields a high recovery rate for Xilinx FPGAs [4]. A mapping table defines the relationship between the hardware and the bitstream by describing how each programmable points such as PLP and PIP in the FPGA is represented in the bitstream. It is an essential component of most reverse engineering tools. Ding’s team proposed a method for generating PLP mapping tables by configuring a single PLP in different ways, generating a bitstream for each configuration, and comparing them with a reference bitstream in which the PLP is unused [4]. Here, PLP refers to the basic logic building blocks within the FPGA hardware, such as multiplexers, gates, and flip-flops. For example, in the case of a flip-flop, which is a type of PLP, the mapping table requires three bitstreams: one reference bitstream where the PLP is unused, and two bitstreams corresponding to the PLP implemented as a flip-flop or as a latch. PIP is a hardware element that connects two wires located on different tiles, where a tile refers to the largest block unit constituting an FPGA. When constructing a circuit on an FPGA, a net is formed by connecting two points, the input and output of a logic element, and consists of one or more PIPs. Ding’s team creates all possible nets for the same

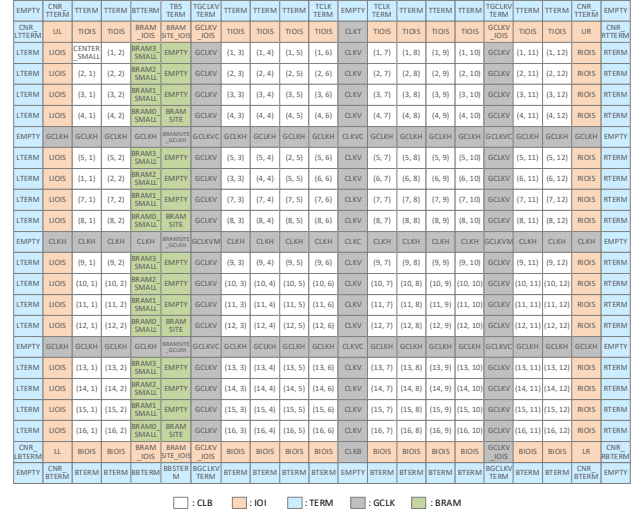


TABLE I. TILE TYPE ON S50 DEVICE.

Category	Tile type	# of tiles
TERM	EMPTY	12
	CNR_LTTERM	1
	CNR_LBTERM	1
	CNR_TTERM	2
	CNR_RTTERM	1
	CNR_RBTERM	1
	CNR_BTERM	2
	TTERM	10
	BTERM	10
	LTERM	16
	RTERM	16
	TGCLKVTERM	2
	BGCLKVTERM	2
	TCLKTERM	1
	BCLKTERM	1
	BTTERM	1
	BBTERM	1
TSTERM	1	
BSTERM	1	
GCLK	GCLKH	30
	BRAMSITE_GCLKH	3
	GCLKV	32
	GCLKVC	4
	GCLKM	2
	CLKH	15
	CLKV	16
	CLKVC	2
	CLKC	1
	CLKT	1
	CLKB	1
IOI	UL	1
	UR	1
	LL	1
	LR	1
	TIOIS	12
	BIOIS	12
	LIOIS	16
	RIOIS	16
	BRAM_IOIS	2
	BRAMSITE_IOIS	2
GCLKV_IOIS	4	
BRAM	BRAM3_SMALL	4
	BRAM2_SMALL	4
	BRAM1_SMALL	4
	BRAM0_SMALL	4
	BRAMSITE	4
	EMPTY	12
CLB	CENTER_SMALL	192

In this paper, we analyze the FPGA and bitstream structures to reduce the number of mapping tables that need to be generated, thereby shortening the time required. We also propose an offset calculation formula for reconstructing all FPGA hardware elements, aiming to improve the feasibility of FPGA reverse engineering.

II. ANALYSIS

To reduce the number of mapping tables that must be created first, we conducted a preliminary analysis of the relationship between the FPGA structure and the bitstream structure. This paper uses the S50 device of Spartan-3 family, shown in Fig. 1, as an example.

The tiles that make up the FPGA are classified into five categories, as illustrated in Fig. 1. These include: TERM (Terminal) tiles, which form the edges of the FPGA; IOI (Input/Output Interface) tiles, which implement the external I/O interfaces; CLB (Configurable Logic Block) tiles, which implement combinational and sequential circuits; and BRAM (Block Random-Access Memory) tiles, which implement large internal memory within the FPGA [6]. The tile types belonging to five categories are listed in Table 1. For tiles included in the TERM and IOI categories unlike BRAM and CLB, they are distinguished as left (L), right (R), top (T), or bottom (B) based on their position.

[illegible]

Fig. 2. Bitstream file for Spartan-3 family S50 device.

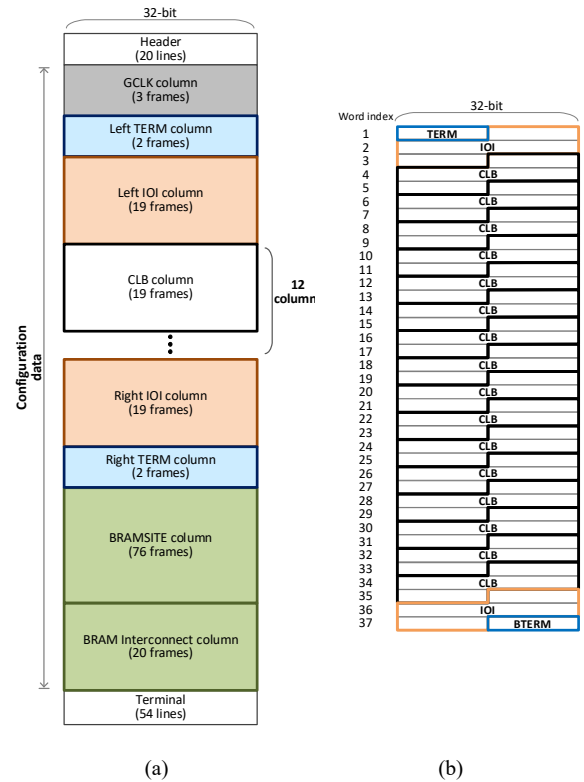


Fig. 3. (a) Bitstream and (b) frame structure for S50 device.

A bitstream is a file containing information about all components of an FPGA represented in binary format, organized in word units as shown in Fig. 2. The structure of the binary representation in Fig. 2 consists of a header and terminal indicating the start and end of the bitstream, and configuration data representing the hardware information that constitutes the FPGA, as shown in Fig. 3(a). The header section contains information such as FPGA device code required to configure the circuit on the FPGA, along with commands. The terminal section consists of commands to finalize the circuit configuration on the FPGA, including a CRC to verify that the

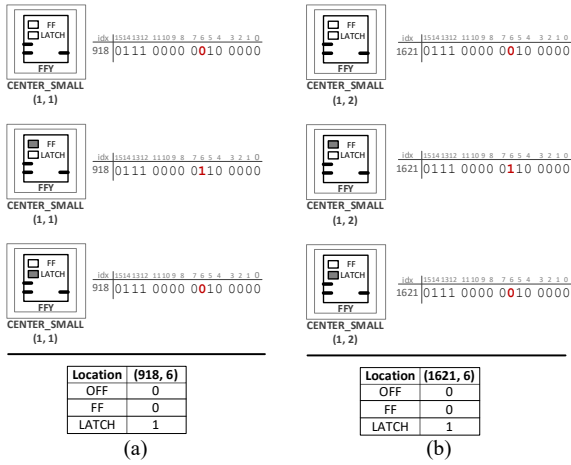


Fig. 4. Example of PLP mapping table for CLB location (a) (1,1) and (b) (1,2).

configuration completed without issues. The configuration data section representing the FPGA structure displays the remaining tile information in columns corresponding to the FPGA structure. Multiple frames are required to fully represent the information of all tiles in each column. A frame is the basic unit of configuration data and consists of multiple words. The number of words in a frame varies depending on the number of tiles in the column. For example, the frame for the S50 device consists of 37 words, as shown in Fig. 3(b). Note that the number of words composing a frame varies with the FPGA size, even within the same family.

The number of frames required to represent each column varies depending on the primary tile types composing the column, as shown in Fig. 3(a) and Fig. 3(b). However, the GCLK column only aggregates the information from the CLK, CLKB, and both GCLKV and GCLKVM tiles among the GCLK tiles shown in Fig. 1, appearing at the very top of the configuration data section. For the remaining GCLK tiles, excluding the four types mentioned, even if they contain PIP, the signal flow is fully determined. Therefore, PIP information does not appear in the bitstream. As shown in Fig. 3(b), the remaining tiles in the column appear in the bitstream sequentially from top to bottom, excluding the GCLK tile. The TERM tile at the top, such as TTERM tile, represents the upper half word of the first word in the frame, and the TERM tile at the bottom, such as BTERM tile, represents the lower half word of the last word in the frame. The remaining tiles display their values sequentially from top to bottom, as shown in Fig. 3(b), that illustrates the configuration for one of CLB column. Meanwhile, the BRAMx_SMALL tile contains only the PIPs connected to the BRAMSITE tile and appears in the BRAM interconnect column of the bitstream in Fig. 3(a). The BRAMSITE tile in Fig. 1 is used to implement the MULT18X18 primitive for complex operation processing or to configure block memory [7]. It represents the initial values of RAM implemented in the block memory or the data stored in ROM within the bitstream. Therefore, the frame configuration in the BRAMSITE column reflects all data stored in each BRAMSITE tile, divided by the number of BRAMSITE tiles, excluding the portions representing TERM and IOI tiles in the frame configuration of Fig. 3(b).

III. PROPOSED METHOD

To reduce the number of mapping tables required, we generate and compare mapping tables for tiles of the same type within the same FPGA. As shown in Fig. 4, we confirm

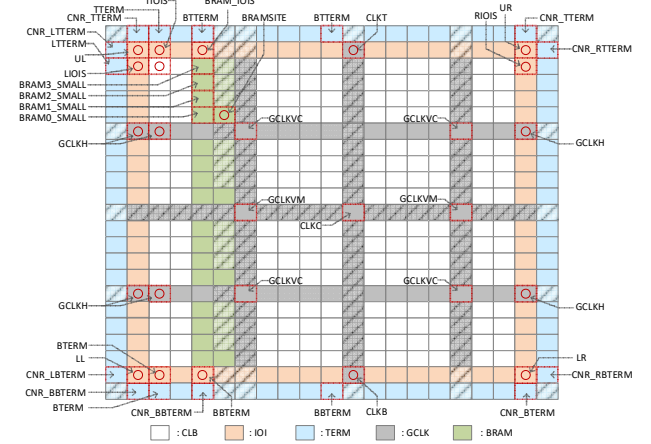


Fig. 5. Reference tile for proposed method.

that tiles of the same type always exhibit identical patterns in the two words representing the tile, regardless of their position. This indicates that differences in hardware location affect only the word index. This means that by creating a single mapping table for each tile type and substituting the word index difference corresponding to the tile position in the bitstream, it is possible to obtain the mapping information for all tiles without directly creating mapping tables for every tile. In addition, the word index differences can be determined based on bitstream analysis, allowing the derivation of an offset calculation formula.

First, for tiles of the same type within a single column, as shown in the structure of Fig. 3(b), the bitstream differs by 2 words for each increment of 1 in row index R of tile. That is, the offset formula for obtaining the mapping tables of tiles within a single column can be determined as shown in (1).

$$(w, b) = (w_{ref} + (R-1) \times 2, b_{ref}), \quad (1)$$

The bitstream position (w, b) where information for tiles of the same type in the same column appears is $(R-1) \times 2$ positions away from the bitstream position (w_{ref}, b_{ref}) where the information of reference tile appears. For example, to obtain the mapping table for CLB (5, 1), (w, b) is $(w_{ref} + 4 \times 2, b_{ref})$.

For tiles of the same type located in different columns, specifically CLB tiles, both the row index R and column index C are taken into account, affecting only the word index in the bitstream. Using this approach, (w, b) can be determined from (2), where $\#frame$ denotes the number of frames in the column, and $\#word_{frame}$ denotes the number of words in a frame.

$$(w, b) = (w_{ref} + (R-1) \times 2 + (C-1) \times \#frame \times \#word_{frame}, b_{ref}), \quad (2)$$

As an example, the CLB column consist of 19 frames, and each frame contains 37 words. Therefore, for CLB (5, 12), the mapping table entry (w, b) is $(w_{ref} + 4 \times 2 + 11 \times 19 \times 37, b_{ref})$.

Equations (1) and (2) appear identically in all tile categories except the GCLK tile. Therefore, only the mapping table for the reference tile needs to be secured for each tile type. In Fig. 5, tiles that require direct mapping table generation are indicated. The red squares and red circles indicate tiles requiring PIP and PLP mapping tables, respectively, while the slashed tiles represent those without any PIP or PLP. Note that, for comparison with previous works [4–5] that performed reverse engineering of PLP and PIP, this paper does not consider the time required to obtain mapping tables for

TABLE II. THE NUMBER OF CONFIGURATIONS FOR PLPs AND PIPs.

Category	Tile type	# of tiles	# of cfgs	# of PIPs
TERM	EMPTY	12	0	0
	CNR_LTERM	1	0	64
	CNR_LBTERM	1	0	64
	CNR_TTERM	2	0	64
	CNR_RTTERM	1	0	64
	CNR_RBTERM	1	0	64
	CNR_BTERM	2	0	64
	TTERM	10	0	64
	BTERM	10	0	64
	LTERM	16	0	64
	RTERM	16	0	64
	TGCLKVTERM	2	0	0
	BGCLKVTERM	2	0	64
	TCLKTERM	1	0	0
	BCLKTERM	1	0	64
	BTTERM	1	0	72
	BBTERM	1	0	0
	TSTERM	1	0	0
	BSTERM	1	0	72
GCLK	GCLKH	30	90	16
	BRAMS-	3	0	0
	ITE_GCLKH	32	0	0
	GCLKV	4	0	16
	GCLKVC	2	0	16
	GCLKM	15	0	0
	CLKH	16	0	0
	CLKV	2	0	0
	CLKVC	1	0	8
	CLKC	1	43	120
	CLKT	1	43	120
	CLKB	1	43	120
IOI	UL	1	4	596
	UR	1	4	886
	LL	1	4	596
	LR	1	27	901
	TIOIS	12	541	1597
	BIOIS	12	541	1593
	LIOIS	16	538	1604
	RIOIS	16	538	1601
	BRAM_IOIS	2	141	1025
	BRAMSITE_IOIS	2	0	0
BRAM	GCLKV_IOIS	4	0	0
	BRAM3_SMALL	4	0	1940
	BRAM2_SMALL	4	0	1991
	BRAM1_SMALL	4	0	2097
	BRAM0_SMALL	4	0	1940
	BRAMSITE	4	55	136
	EMPTY	12	0	0
CLB	CENTER_SMALL	192	452	2598

programmable data points that the data stored in look-up tables within BRAM and CLB tiles.

IV. EXPERIMENTAL RESULT

For comparison with previous studies and the proposed method, the Spartan-3 S50 device is used, and a PC equipped with an Intel Core i7 processor (@ 2.50 GHz) and 64 GB of RAM is employed to generate the mapping tables. The bitstreams are created using ISE Design Suite version 14.7.

First, the number of tiles, PLPs, and PIPs contained within each tile type constituting the FPGA are summarized in Table 2 that required for the previous works. Some tiles contain only PIPs or do not contain both PLPs and PIPs. Since the number of possible configurations (cfg) for each PLP varies, Table 2 concisely shows only the total number of configurations considered to create the PLP mapping table for a single tile. For the previous work, the number of bitstream files that need to be generated for previous work is given by the product of the number of tiles and the number of configurations or PIPs. Otherwise, Using the proposed method, generating a

TABLE III. TIME FOR MAPPING TABLE GENERATIONS.

	PLP		PIP	
	Previous [days]	Proposed [days]	Previous [days]	Proposed [days]
GCLK	0.00	0.00	2.01	0.55
TERM	0.08	0.05	0.39	0.16
IOIS	14.46	1.09	43.47	3.52
BRAM	0.10	0.03	15.38	3.85
CLB	55.83	1.38	288.85	9.27
Total	70.49	2.54	360.11	17.35

mapping table for a single tile of each tile type is sufficient.

Based on the analysis of the FPGA internal structure, the time required to create the mapping table for the S50 device is summarized in Table 3. To compare with the method proposed in this paper, the method from Paar's team [5] among previous works [4–5] is adopted; thus, the previous work in Table 3 refers to Paar's team method [5]. According to Table 3, the time required to create the mapping table is reduced by approximately 94% for PLP and 95% for PIP. This means that what previously took 71 days for PLP and 350 days for PIP is now completed in 2.6 days and 18 days, respectively. This validates that all mapping tables can be obtained in about 21 days without requiring additional hardware resources that about 20 times faster than the previous work. Furthermore, since the mapping tables generated using the proposed method match those generated using the previous method, the reverse engineering results are also identical.

V. CONCLUSION

In this paper, we proposed a method to reduce the time required to obtain PLP and PIP mapping tables for reverse engineering Xilinx FPGAs. By analyzing FPGA and bitstream structures and leveraging similarities between mapping tables, only reference tables are generated directly, and the remaining tables are derived using an offset-based approach. This method enables all mapping tables to be obtained in approximately 21 days, a 20-fold reduction compared with Paar's team method [5], which requires roughly 421 days. These results demonstrate the practical feasibility of reverse engineering attacks on FPGA designs and highlight the need to develop effective countermeasures.

REFERENCES

- [1] S. Wallat, M. Fyrbiak, M. Schlögel and C. Paar, "A look at the dark side of hardware reverse engineering - a case study," 2017 IEEE 2nd International Verification and Security Workshop (IVSW), 2017, pp. 95-100.
- [2] J.-B. Note and E. Rannaud, "From the bitstream to the netlist," in Proceedings of the 16th international ACM/SIGDA symposium on Field programmable gate arrays, 2008, pp. 264-264.
- [3] F. Benz, A. Seffrin and S. A. Huss, "Bil: A tool-chain for bitstream reverse-engineering," 22nd International Conference on Field Programmable Logic and Applications (FPL), 2012, pp. 735-738.
- [4] Z. Ding, Q. Wu, Y. Zhang, and L. Zhu, "Deriving a n NCD file from an FPGA bitstream: Methodology, architecture and evaluation," Microprocessors Microsystems-Embedded Hardware Des., vol. 37, no. 3, pp. 299-312, 2013.
- [5] M. Ender, P. Swierczynski, S. Wallat, M. Wilhelm, P. M. Knopp, and C. Paar, "Insights into the mind of a trojan designer: the challenge to integrate a trojan into the bitstream," in *ASPDAC 2019*. ACM, 2019, pp. 112-119.
- [6] Xilinx, "Spartan-3 Generation FPGA User Guide (UG331)," 2011.
- [7] Xilinx, "Spartan-3 Libraries Guide for HDL Designs (UG607)," 2013.