PAPER *Special Section on Information and Communication System Security*

A Secure Content Delivery System Based on a Partially Reconfigurable FPGA

Yohei HORI[†a)], **Hiroyuki YOKOYAMA**[††b)], **Hirofumi SAKANE**[†c)], ***and*** **Kenji TODA**[†d)], ***Members***

SUMMARY We developed a content delivery system using a partially reconfigurable FPGA to securely distribute digital content on the Internet. With partial reconfigurability of a Xilinx Virtex-II Pro FPGA, the system provides an innovative single-chip solution for protecting digital content. In the system, a partial circuit must be downloaded from a server to the client terminal to play content. Content will be played only when the downloaded circuit is correctly combined (= *interlocked*) with the circuit built in the terminal. Since each circuit has a unique I/O configuration, the downloaded circuit interlocks with the corresponding built-in circuit designed for a particular terminal. Thus, the interface of the circuit itself provides a novel authentication mechanism. This paper describes the detailed architecture of the system and clarify the feasibility and effectiveness of the system. In addition, we discuss a fail-safe mechanism and future work necessary for the practical application of the system.

key words: *field-programmable gate array (FPGA), partial run-time reconfiguration (RTR), content protection, digital rights management (DRM)*

1. Introduction

The expansion of broadband networks and the spread of Internet-access-ready mobile terminals have brought about the rapid growth of the on-line content market [1]. Digital content delivery systems, however, are always threatened by prevalent piracy, *i.e.*, system cracking and illegal copying of copyrighted content. Therefore, technology of Digital Rights Management (DRM) is of primary concern for content providers.

Based on the above considerations, various studies have examined the secure distribution of content on the Internet [2]–[4]. As observed in these studies, the security of content is guaranteed mainly by cryptographic technology, *e.g.*, RSA [5] and AES [6]. However, even though the implemented cryptographic algorithms are theoretically unbreakable, vulnerabilities could emerge when the algorithm is instantiated in the real world [7]–[11]. Since it is nearly impossible to completely eliminate design defects from recently developed complicated systems, confidential data will be made vulnerable by these flaws. This means that, due to physical and structural defects of a system, rather than algorithmic weakness, the strength of cryptography cannot be guaranteed. Since attack methods that exploit such vulnerabilities continue to advance, content delivery systems require not only security but also flexibility in order to employ countermeasures against new piracy.

To develop a secure and flexible content delivery system, we have proposed utilizing the partial reconfigurability of a Field-Programmable Gate Array (FPGA) [12]–[15]. An FPGA is one of the most popular reconfigurable devices. Some FPGAs support run-time partial reconfiguration (RTR), which allows a small area of the whole circuit to be replaced with another module without stopping the rest of the circuit. One of the advantages of an RTR-based content distribution system is that functionality of a module is changed on demand. For example, a flexible system is possible to be developed where a hardware decoder of music or video is dynamically replaced according to the content. Another merit is that security modules in the system can be reactively updated when a vulnerability is found or a new attack technique is invented.

The main concern of our research is to enhance security of a content distribution system using RTR. Our strategy of utilizing partial reconfiguration is quite different from those in related studies [16], [17]. We focus on the signal interface between the reconfigurable part and fixed part of the circuit. The system properly works only when the signals in the two circuits are correctly connected. In other words, information of the I/O configuration between the partial and fixed circuit can behave as a secret key to activate the system. Thus, the goal of our study is to say realizing a hardware-based lock-and-key authentication for a content delivery system with a partially reconfigurable FPGA. In our system, a partial circuit must be downloaded from the server to the client terminal in order to play content. Content is properly played only when the downloaded circuit is correctly combined with the circuit built in the terminal. To the authors' best knowledge, this is the first approach that uses the reconfigurability of an FPGA in this way to enhance security of a system.

In our previous work, we mainly considered the architecture and technical feasibility of an RTR-based client terminal. The previous system was able to play content, however, it was vulnerable to erroneous inputs. The purpose of this study is to experimentally verify the newly developed RTR-based content delivery system. This paper is organized as follows. Section 3 explains the feature of partial reconfiguration of an FPGA. Section 4 describes an overview of the system and the mechanism of the content protection.

Manuscript received July 26, 2007.
Manuscript revised December 11, 2007.

[†]The authors are with National Institute of Advanced Industrial Science and Technology (AIST), Tsukuba-shi, 305–8568 Japan.
[††]The author is with KDDI R&D Laboratories, Inc., Fujimino-shi, 356–8502 Japan.
a) E-mail: hori.y@aist.go.jp
b) E-mail: yokoyama@kddilabs.jp
c) E-mail: hirofumi.sakane@aist.go.jp
d) E-mail: k-toda@aist.go.jp

DOI: 10.1093/ietisy/e91–d.5.1398

Section 5 describes the detailed architecture and the implementation result of the system. Section 6 experimentally demonstrates the feasibility of the RTR-based content delivery system. Section 7 discusses the current implementation of the system based on the experimental results, and finally Sect. 8 concludes this study.

2. Related Work

In the on-line content distribution business, a method to securely transfer secret information is the most important concern. There are various studies reported so far to examine the secure content distribution, as described in Sect. 1. In this section, we briefly introduce Digital Cinema Initiative (DCI) [18]. DCI is a group of the seven major film companies and has published the specifications for packaging, distributing, and playing digital cinema.

In the DCI specifications, content are encrypted with the AES-128 CBC symmetric cipher. The content key is transferred as the pay load of Key Delivery Message (KDM). The KDM is encrypted and exchanged based on the RSA asymmetric key cipher with a 2048-bit key. The secret key to decrypt the KDM must be stored in the secure silicon device which each auditorium/projector is equipped with. The secure silicon device is required to be tamper-evident, tamper-resistant, tamper-detecting and tamper-responsive. Therefore the secret key embedded in the silicon device is considered as unextractable.

As for our system, the concern is the design defect, human error/malevolence, advancing attack technique and other unpredictable failure that will leak original plain data of the transferred message. So we propose a method that would minimize the risk of data leakage by utilizing the partial reconfigurability and the intractable routing information of the FPGA. The carefulness for the accidental and unpredictable leakage will gain the market's and popular acceptance. We emphasize that our approach is *not* to replace the cryptographic protection with the bitstream intractability, but to add the further strength of security to the system with the flexibility and the bitstream intractability.

3. Partial Reconfiguration of FPGAs

This section briefly describes the features of partial reconfiguration of a Xilinx FPGA employed in the proposed system. For more detailed information on Xilinx partial reconfiguration, see [19].

3.1 Features of Xilinx FPGA Reconfiguration

A partially reconfigurable design of a Xilinx FPGA consists of three major modules: *Top Module*, *Base Module*, and *Partially Reconfigurable Modules (PRMs)*. The Top Module includes the Base Module and the PRMs, and the Base Module usually contains several smaller submodules. The Base Module is a set of non-reconfigurable modules, and the PRM is a dynamically reconfigurable part of the design. The area of the device in which a PRM is implemented is called the *Partially Reconfigurable Region (PRR)*. A conceptual structure of a partially reconfigurable circuit is given in Fig. 1.

All signals between a PRM and a fixed module must pass through *bus macros* to lock the wiring. The bus macro is a unidirectional 8-bit-wide pre-routed macro. The bus macro must be placed on the module boundary between a PRM and a fixed module.

Virtex-II [20] and newer Virtex series devices support self-reconfiguration with the *Internal Configuration Access Port* (*ICAP*). Since user logic can access configuration memory through the ICAP, partial reconfiguration of the FPGA can be controlled by internal circuits.

3.2 Design Flow of a Reconfigurable Circuit

The design flow of a partially reconfigurable circuit is quite different from that of an ordinary circuit. The procedure for designing a reconfigurable circuit is summarized as follows:

1. Budgeting of Top Module: All global primitives, (*e.g.*, clock primitives, I/O buffers, and bus macros), are placed in specific positions (Fig. 2 (a)). All submodules are declared as black boxes.
2. Base Module implementation: Fixed modules are placed and routed based on the Top Module budgeting. Note that, in the PRR, fixed modules are allowed to use routing resources but are prohibited from using logic resources (Fig. 2 (b)). The routing resources occupied by the fixed modules are recorded to a file named *static.used*.
3. PRM implementation: The PRM is placed and routed based on the Top Module budgeting and the file static.used. The file is copied and renamed *arcs.exclude*. The implementation tool avoids using the routing resources written in arcs.exclude (Fig. 2 (c)).

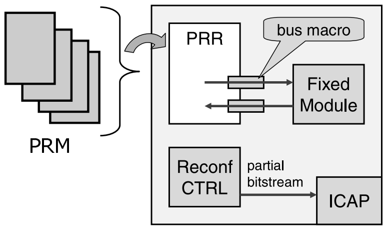


Fig. 1 Structure of a partially reconfigurable circuit with a Xilinx FPGA.

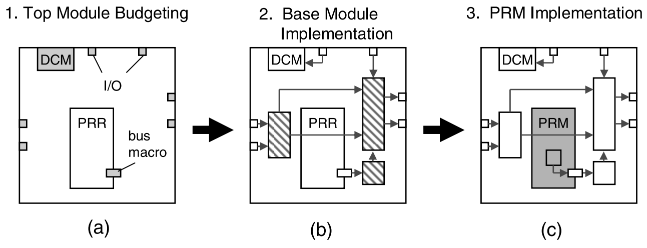


Fig. 2 Design flow of a partially reconfigurable circuit.

4. Merge: The configuration data (= bitstream) of the completed circuit is built from the Base Module and PRMs. The partial bitstream of each PRM is also generated here.

Top Module budgeting and Base Module implementation are common for all PRMs in the same design. As long as the Top Module design does not change, Steps 3 and 4 are performed repeatedly in order to generate bitstreams of different PRMs. If the positions of global primitives, for example bus macros, are changed, all of the above-described steps must be performed again.

3.3 Bitstream of FPGAs

Configuring and reconfiguring an FPGA is realized by writing and rewriting a bitstream to the configuration memory in the FPGA. Since the bitstream of the FPGA is an electronic message, the FPGA will be easily copied if the bitstream is eavesdropped. Therefore some recent FPGAs support DES/AES encrypted bitstreams so that plain bitstreams will not be exposed to adversaries. Such an FPGA has a dedicated decryptor and key storage for decrypting cipher bitstreams.

Contrary to our objective, when a Xilinx FPGA is configured with a cipher bitstream, the partial reconfigurability is automatically disabled. It means that the boot-up with a cipher bitstream is unavailable in partially reconfigurable designs. Therefore in our system, the FPGA is started up with a plain bitstream at first, and then partially reconfigured with an encrypted partial bitstream that is decrypted with user logic.

The boot-up with a plain bitstream could be a vulnerable point of the FPGA-based system. We expect that this restriction will be removed in the future FPGA release, and the encrypted bitstream will be available for both entirely and partially reconfigurable designs. Another way to eliminate the weakness is to use a flash-embedded FPGA that can contain the whole bitstream inside the chip, though such FPGAs released so for are too small to implement our design. The weakness is also overcome by packing the FPGA core and ROM containing the initial bitstream into one package. If the package is broken, the data in the ROM will be erased. Based on these considerations, the problem of the boot-up with a plain bitstream is expected to be resolved in the future system.

The bitstream encryption is a technique to prevent plain configuration data from being eavesdropped “on the networks and bus lines”. Note that the decrypted bitstream does exist in the configuration memory during the FPGA is working. An FPGA usually provides the function to read back the bitstream from configuration memory, so the designer must be sure to disable the read-back functionality.

There could be still a way to extract a bitstream in the FPGA, though it is extremely costly. For example, a bitstream in the configuration memory could be obtained by removing the package, eliminating the metal layer, and observing the bits with a scanning electron microscope (SEM). This kind of tampering, however, is considered as impracticable for most attackers [21] and will not be discussed in this paper. So it is safe to say that the plain bitstream will not be extracted from the configuration memory as long as the FPGA is correctly set up and. Based on the above consideration, we assume that the plain bitstream of the FPGA is unextractable.

The complexity of the bitstream is also available to protect the secret information in the bitstream. Though there are some documents giving information about FPGA bitstreams [22], [23], there is no report that a bitstream is successfully reverse-engineered [21]. Therefore if a plain bitstream is leaked due to the design defect or other accidental reasons, the intractability of the bitstream will allow enough time to replace the leaked configuration data with new one. However, as is also mentioned in [21], depending on the tedium and the complexity of bitstream reverse-engineering is risky. Note that our system’s security is supported by the partial reconfigurability of the FPGA, and mainly guaranteed by the theoretical strength of cryptographic algorithms (AES).

4. System Architecture

In this section, we first present the overview of the system architecture. We then explain the content protection mechanism based on partial reconfiguration of an FPGA.

4.1 Overview of the System

In the on-line content delivery system, the most significant concern is to securely transfer the secret key. In a typical system, the content key is encrypted by the server and transferred to the user terminal. As is mentioned in Sect. 2, our apprehension is that the original plain data of the transferred message, or the content key, might leak due to the design defect, human error/malevolence, advancing attack technique, and other unpredictable reasons.

To minimize the risk of the data leakage, the original plain data of the transferred message should

1. be unworkable on the unauthorized terminals and worthless for adversaries,
2. be intrinsically intractable so as not to be analyzed by adversaries, and
3. contain secret information as little as possible.

The partial reconfigurability of the FPGA and intractability of the bitstream is effective to meet the requirements given here.

Figure 3 shows the block diagram of our FPGA-based content delivery system. The system consists of a server, client terminals, and networks connecting them. To play content on a client terminal, a user must download a partial circuit from the server. The downloaded circuit is called the *Content-Specific Circuit* (*CSC*), and the circuit built in the client terminal is called a *Terminal Built-in Circuit* (*TBC*).

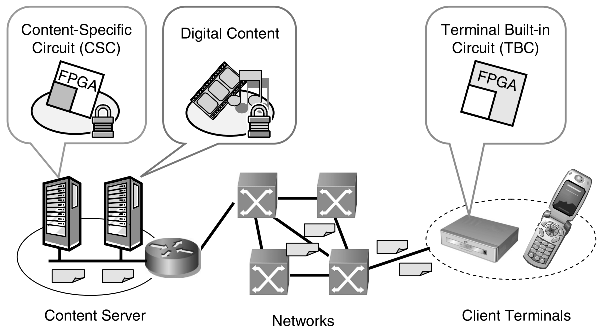

Content-Specific Circuit (CSC)
FPGA
Digital Content
Terminal Built-in Circuit (TBC)
FPGA
Content Server
Networks
Client Terminals

Fig. 3 Overview of the content delivery system.

We use the term *interlock* to describe the condition in which CSC and TBC are correctly combined to work as intended.

The key ideas of the systems are that

1. each TBC has the different I/O configuration so that only the expected CSC interlocks with the TBC and the CSC will not be abused by unauthorized terminals,
2. a partial bitstream of the FPGA is transferred, because the bitstream is intractable enough to thwart adversaries in extracting secret information from it, and
3. a part of the key generating circuit, not a whole circuit nor the key itself, is transferred so that the information of the secret key will not be extracted.

The method of configuring the terminal specific I/O interface is explained in Sect. 5.

4.2 Mechanisms of Content Protection

We suppose that the CSC-TBC architecture will give further strength of security to the system. This section explains how the interlocking mechanism protects digital content with partial reconfigurability.

4.2.1 Authentication with I/O Configuration

To play content on the client terminal, proper CSC must be configured and interlocked with TBC, in other words, signals between CSC and TBC must be correctly connected. Since each TBC has a unique I/O configuration, CSC interlocks only with the TBC of a specific user. For this reason, even if CSC is leaked and distributed on the network, the leaked CSC will not work on the other terminals.

4.2.2 Content-Specific Hardware Architecture for Illegal-Play Prevention

As mentioned earlier, algorithms implemented on the CSC vary depending on the content to be played. The CSC can be used for playing only specific content. Thus, playable content is determined by the architecture of the downloaded CSC. For this reason, even if a plain CSC bitstream is distributed on the network, it is difficult to determine which content is playable with the CSC.

4.2.3 Data and Algorithm Obfuscation

In the system, a partial bitstream of a key generating circuit, not a key itself, is transferred from the server. Even if the encrypted bitstream is obtained surreptitiously and decrypted for some reason, the bitstream is sufficiently intractable to most attackers. In addition, the behavior of the entire circuit will not be determined from the partial bitstream because it is merely a small fraction of the entire configuration data.

4.2.4 Single-Chip Wiretapping-Resistant Architecture

With partial reconfigurability of an FPGA, CSC and TBC are implemented on a single chip. Therefore, any communication between CSC and TBC cannot be wiretapped on the external buses. During the processing of key generation or content decryption, neither decipher keys nor intermediate data will be exposed to attackers.

4.2.5 Reactive Design Modification

As the architecture of recent devices and systems becomes increasingly complicated, it is nearly impossible to completely eliminate defects and security vulnerabilities in consumer electronics. In fact, new attack techniques exploiting such vulnerabilities have been frequently reported. With the reconfigurability of an FPGA, we can repair defects and vulnerabilities in a product even after shipment.

5. Implementation

We developed a prototype CSC-TBC-based content delivery system with off-the-shelf FPGA boards. This section describes the architecture of the prototype system, details of processing performed in CSC-TBC, and the implementation results of the CSC-TBC authentication mechanism.

5.1 Architecture of the Prototype System

Figure 4 shows the architecture of the prototype system. When a request is sent from the client terminal, the server authenticates the terminal using a challenge-response authentication protocol. Since the purpose of this implementation is to verify the feasibility of CSC-TBC interlocking authentication rather than existing authentication protocol, challenge and response are simply computed by XORing constant values. A CSC bitstream and content are encrypted with AES128-CBC [6]. Data used in the system are denoted as follows:

D_{tid} :	a 128-bit ID number given to each terminal.
D_{cid} :	a 128-bit ID number given to each content.
D_{seed} :	a 128-bit random number sent from the server.
D_{csc} :	a configuration bitstream of CSC.
D_{cont} :	original content data.
K_{csc} :	a 128-bit key to encrypt/decrypt D_{csc}.
K_{cont} :	a 128-bit key to encrypt/decrypt D_{cont}.
$\mathcal{E}\{D\}_K$:	data D encrypted with the key K.

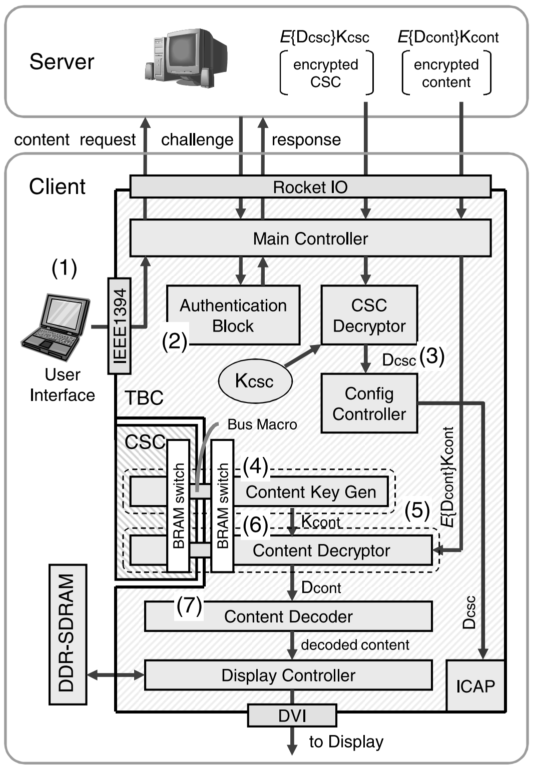


Fig. 4 Block diagram of the prototype system.

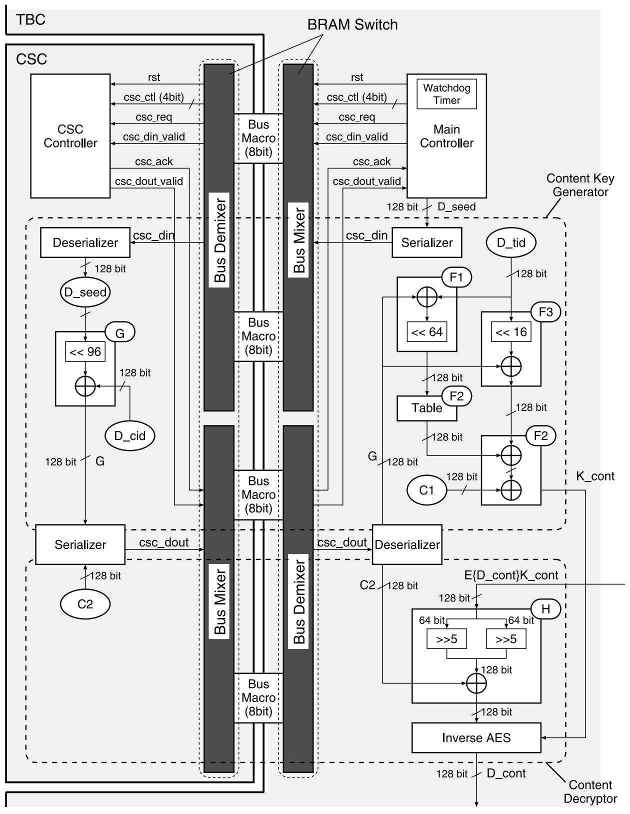


Fig. 5 Diagram of the CSC-TBC processing.

Note that the system uses two different keys: K_{csc} embedded in TBC and K_{cont} generated with the CSC-TBC mechanism.

In the present implementation, the *Content Key Generator* and the *Content Decryptor* are implemented across CSC and TBC. All signals between CSC and TBC pass through bus macros. The connection of the signals is shuffled and deshuffled in BRAM Switch. BRAM Switch is a Block-RAM-based bus line mixer/demixer that realizes a terminal specific I/O configuration by changing values in the BRAM. In the previous work, the connection between bus macros and modules is fixed and the terminal specific CSC-TBC interface is realized by changing the positions of bus macros. In that case, however, the whole circuit must be re-compiled to change the placement of bus macros as explained in Sect. 3.2. With the BRAM Switches, flexible connection is realized without moving bus macros.

The procedure for playing content on the prototype system is summarized as follows:

1. Request for content is sent from the client terminal to the server.
2. Challenge-response authentication is performed.
3. $\mathcal{E}\{D_{csc}\}_{K_{csc}}$ is downloaded from the server and decrypted with the embedded key K_{csc}. The decrypted data (D_{csc}) are sent to ICAP and CSC is configured.
4. If the CSC correctly interlocks with TBC, the proper K_{cont} is generated.
5. $\mathcal{E}\{D_{cont}\}_{K_{cont}}$ is downloaded from the server.
6. $\mathcal{E}\{D_{cont}\}_{K_{cont}}$ is decrypted with the generated K_{cont} to obtain the original content data D_{cont}.
7. D_{cont} is decoded and played.

5.2 Details of the Implemented Functions

Figure 5 shows the detailed block diagram of the interlocked CSC-TBC in the client terminal. CSC and TBC are connected by two left-to-right bus macros and and two right-to-left bus macros. The signals passing through bus macros are monitored by a watchdog timer. If the connection is not established within a specific period, the running process will be aborted. Each bus macro is 8-bit-wide and therefore the bus width for each direction is 16 bits. The signals from CSC to TBC is 3 bits, and the signals from TBC to CSC is 8 bits.

As mentioned earlier, the purpose of this implementation is to verify the feasibility of the CSC-TBC interlocking mechanism, and so the strength of the functions is not considered at this time. We implemented typical operations, (*e.g.*, exclusive OR, cyclic shift, and table reference), on CSC and TBC in order to estimate hardware utilization of a completed system. The functions implemented on CSC and TBC are described in the following equations:

$$F_1 = (G \oplus D_{tid}) \ll 64 \qquad (1)$$

$$F_2 = TABLE(F_1) \qquad (2)$$

$$F_3 = (G \oplus (D_{tid} \ll 16)) \ll 32 \qquad (3)$$

$$F_4 = F_2 \oplus F_3 \oplus C_1 \tag{4}$$

$$G = (D_{seed} \ll 96) \oplus D_{cid} \tag{5}$$

$$H = \{TopHalf(\mathcal{E}\{D_{cont}\}_{K_{cont}}) \gg 5, \\ BottomHalf(\mathcal{E}\{D_{cont}\}_{K_{cont}}) \gg 5\} \oplus C_2 \tag{6}$$

In these equations, $TABLE(i)$ is a function in which the table is referenced by address i, and the size of the table is 32 kbits (= 128 bits × 256 entries). C_1 and C_2 are 128-bit random numbers embedded in TBC and CSC, respectively. The operators '$\oplus$', '$\ll$', and '$\gg$' express *bitwise logical exclusive OR*, *cyclic left-shift*, and *cyclic right-shift*, respectively. In function H, the top half and bottom half of the input data are shifted circularly to the right by 5 bits, and the concatenated result is XOR-ed with C_2.

In the completed system, D_{tid} and C_1 are embedded in TBC, D_{cid} and C_2 are embedded in CSC. In addition, K_{csc} is generated in TBC. In order to facilitate debugging, these constant values and K_{csc} are configurable via desktop computer.

5.3 Bus Complexity and System Security

In the current implementation, the positions of the bus macros are fixed, and the connection of the signals between CSC and TBC is changed by BRAM Switches. The width of the left-to-right and right-to-left buses are both 16 bits. The signals from CSC to TBC is 3 bits, therefore the number of the possible patterns of bus usage is ${}_{16}C_3 \times 3! = 3360$. The signals from TBC to CSC is 8 bits and the number of possible patterns of bus usage is ${}_{16}C_8 \times 8! = 518918400$. Simply considering, the number of the possible patterns of the I/O configuration between CSC and TBC is totally $3360 \times 518918400 = 1.743565824 \times 10^{12}$.

As is mentioned in Sect. 3.3, we assume that the plain bitstream of TBC is unable to be extracted. On the other hand, we assume that the plain bitstream of CSC might leak to the networks due to the design defect, human error/malevolence, advancing attack technique, or other unpredictable reasons. Although the plain bitstream itself is intractable to be analyzed, we can not ignore the possibility of the exposure of the structural information.

If the configuration of the BRAM Switch is revealed, the positions of the active bus lines can be identified. In this case, the number of the pattern of the bus usage is $3! \times 8! = 241920$. This number seems small, but actually the number is safe enough for the system. Note that this is the number of the trials of configuring CSC, and is not simulated on a computer. That is, the adversary needs to try configuring the CSC on the terminal. The trial of configuring CSC will take a few seconds, but it is too long to test all patterns of the bus usage. Additionally, the system will be protected by the following method:

- The terminal records the number of the failure of CSC configuration.
- The terminal refuses further configuration trial if the number of the failure reaches the pre-defined limitation.

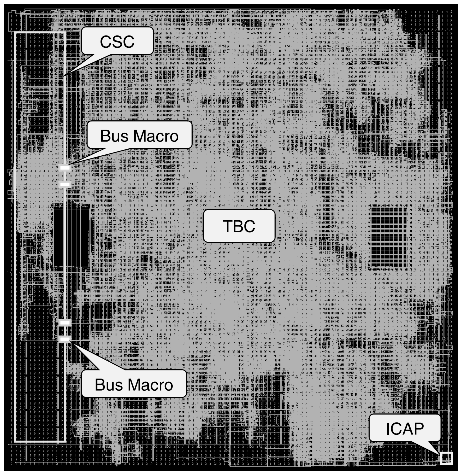


Fig. 6 Routing result of the prototype system.

Implementation of these functionality is effective for the practical use of the system, and is left as the future work.

5.4 Implementation Results

We developed a prototype system with FPGA boards produced by REXEON Technology, Inc.[†]. The main board of the system is equipped with a Virtex-II Pro FPGA (XC2VP70) [24]. Figure 6 shows the routing result of the entire circuit, including TBC and CSC.

The implementation results of the present system, (with BRAM Switch), the previous system (without BRAM Switch), and a referential system that does not use partial reconfiguration are shown in Table 1. In the referential system, CSC decryptor, reconfiguration controller, bus macros, BRAM Switches and other related logic are deleted. The explanations of the items in the table are found in [24]. The items from IOB through ICAP are global primitives and only total utilization is given. As Table 1 shows, difference between the present and previous system is remarkable in BRAM utilization, while utilization of other logic resources are nearly the same. When compared to the reference system, the present system uses about 1.4 times flip flops, 1.3 times LUTs and 1.5 times BRAMs.

The size of the partial bitstream depends mostly on the area of the CSC, not the hardware utilization. In the implementation with/without BRAM Switch, the size of the bitstream of CSC is 265 KB. The partial reconfiguration, including AES decryption, of CSC takes about 0.95 msec at 100 MHz.

6. Experiment

In order to demonstrate the concept whereby the content is played only when (1) the signals between CSC and TBC are

[†]REXEON Technology, Inc. is a venture company supported by AIST. For more information, see http://www.rexeon.com/.

Table 1 Implementation results and comparison of hardware utilization of the systems.

	Method with BRAM Switch			(%)	Method without BRAM Switch			(%)	Not using partial reconf.	(%)
	CSC	TBC	Total		CSC	TBC	Total			
Flip Flop	1,062	9,699	**10,761**	**(16%)**	1,051	9,693	**10,744**	**(16%)**	**7,526**	**(11%)**
LUT	622	11,341	**11,963**	**(18%)**	577	11,324	**11,901**	**(18%)**	**9,078**	**(13%)**
BRAM	28	212	**240**	**(73%)**	0	184	**184**	**(56%)**	**160**	**(48%)**
MULT18x18	0	49	**49**	**(14%)**	0	49	**49**	**(14%)**	**49**	**(14%)**
IOB	-	-	**139**	**(13%)**	-	-	**139**	**(14%)**	**139**	**(13%)**
GCLK	-	-	**6**	**(37%)**	-	-	**6**	**(37%)**	**6**	**(37%)**
DCM	-	-	**4**	**(50%)**	-	-	**4**	**(50%)**	**4**	**(50%)**
GT	-	-	**5**	**(25%)**	-	-	**5**	**(25%)**	**5**	**(25%)**
ICAP	-	-	**1**	**(100%)**	-	-	**1**	**(100%)**	**0**	**(0%)**

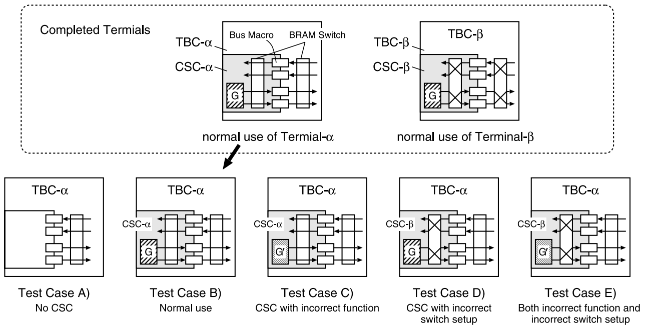


Fig. 7 Test cases of the experiment.

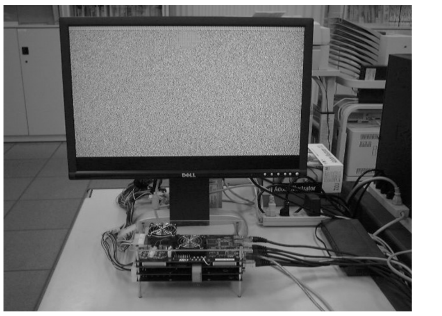


Fig. 8 Content improperly played on a terminal on which CSC failed to interlock with TBC.

correctly connected and (2) CSC has proper functions for the content, we play a high-definition (1920×1080, 30 fps) movie using various CSC-TBC configurations. This section explains the verified CSC-TBC configurations and the results of the experiment.

6.1 Test Cases

To verify the experimentally obtained results, we prepare two client terminals: Terminal-α and Terminal-β. Terminal-α works correctly if TBC-α interlocks with CSC-α, which has function G. Similarly, Terminal-β works correctly if TBC-β interlocks with CSC-β, which has function G.

The only difference between Terminal-α and Terminal-β is the setup of the BRAM Switch. In order to intentionally provide an erroneous function to CSC, we implement G' which returns the inverted value of G. With the symbols defined here, the test cases performed in the experiment are listed below and are also illustrated in Fig. 7.

A) CSC is not configured on the Terminal-α. In this case, the terminal is simply booted up and CSC is not yet downloaded.
B) CSC-α with function G is configured on TBC-α. This is a normal case, in which an authorized user is to play content on an appropriate terminal.
C) CSC-α with function G' is configured on TBC-α. In this case, a user attempts to play content, but the CSC is for different content.
D) CSC-β with function G is configured on TBC-α. In this case, a user attempts to play content, but the CSC is for a different user.
E) CSC-β with function G' is configured on TBC-α. In this case, an unauthorized person obtains a terminal for some reason and attempts to play content with a fake CSC.

In the experiment, not all combinations of TBC-α/-β, CSC-α/-β, G, and G' were tested because some combinations are essentially identical. For example, the combination of TBC-α, CSC-β and function G is fundamentally identical to the combination of TBC-β, CSC-α, and function G. Therefore, the latter combination is omitted from the test cases.

6.2 Experimental Results

The following results were obtained from the experiment:

A) When the system tried to generate a content key, computation was terminated by the watchdog timer because communication between CSC and TBC was not established. The movie was not properly played and white noise was displayed as shown in Fig. 8.
B) Content was properly played as shown in Fig. 9. The result shows that
   - $\mathcal{E}\{D_{csc}\}_{K_{csc}}$ was successfully decrypted,
   - partial reconfiguration properly worked to configure CSC, and
   - CSC and TBC were correctly interlocked to generate K_{cont} and decrypt $\mathcal{E}\{D_{cont}\}_{K_{cont}}$.
C) Configuration of CSC and generation of a content key finished with no error. However, the content was not

Fig. 9 Content played on a terminal on which CSC correctly interlocks with TBC.

correctly played and white noise was displayed. The result indicates that the generated key was incorrect for the content.

D) Configuration of CSC finished with no error. When the system tried to generate a content key, however, computation was terminated by the watchdog timer because communication between CSC and TBC was not established. The movie was not properly played and white noise was displayed.

E) The same result as the test case D) is obtained.

7. Discussion

This section first discusses the feasibility of the system, and then considers the prevention of the system halt. Lastly this section discusses the fail-safe mechanism requisite for the practical application of the system.

7.1 Feasibility of the Interlocking Mechanism

We here discuss the feasibility of the interlocking mechanism with CSC-TBC architecture based on the experimental results described in Sect. 6.2.

- Content is played in the case B) but not played in A). The results show that CSC is requisite for playing content.
- Content is played in the case B) but not played in C). The results show that content will not be played when functions implemented on CSC do not correspond to the content, even if CSC is combined with TBC. Therefore the content-specific function on CSC can control key generation and content decryption so that only the targeted content is played on the terminal.
- Content is played in the case B) but not played in D). The results show that content will not be played if signals between CSC and TBC are not correctly connected, even if the implemented functions are proper for the content. Therefore the CSC-TBC interlocking mechanism can prevent content from being improperly played on unauthorized terminals.

In summary, in order to play content on the terminal, (1) functions implemented on CSC must be proper for the content to be played, and (2) signals between CSC and TBC must be correctly connected. Therefore the interlocking mechanism with CSC-TBC architecture successfully works to control play of content as we intend.

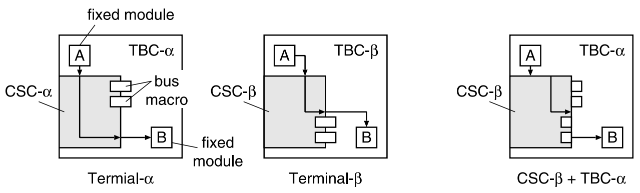


Fig. 10 Inconsistent wiring in a wrong configuration of bus macros.

7.2 Considerations on the System Halt

When CSC and TBC do not interlock, content is not properly played and white noise is displayed in the present system. However, the system is still running and ready for another CSC bitstream. If the correct CSC is configured afresh, the content is properly played. On the other hand, the previous system stops if the signal connection between CSC and TBC is incorrect [15]. In this case, another CSC can not be configured on the system anymore.

A conclusive reason for the system halt is not yet figured out, but the significant difference between the present and previous system is that the former can change the I/O configuration of CSC without re-compilation, while the latter must be re-compiled to change the positions of the bus macros. As explained in Sect. 3.2, the Base Module is allowed to use routing resources in the PRR, and the PRM uses the rest of the routing resources. This means that the PRM also contains interconnections among fixed modules. The routing changes every time the circuit is compiled, thus the wiring among fixed modules is not maintained after recompilation. This inconsistent wiring presumably causes the system halt (Fig. 10).

Considering the discussion above, the BRAM switch serves important functions to realize both flexible signal connection and fail-safe architecture. Even though the overhead of the BRAM Switch is slightly large as shown in Table 1, the fail-safe architecture of the present system against the erroneous CSC bitstream is a compensating advantage.

7.3 Fail-Safe Mechanism of the System

In a system in which a circuit is partially/entirely reconfigured, countermeasures against unexpected errors must be carefully devised. The current system avoids system halt with the BRAM Switches, however, an erroneous bitstream could still cause fatal damage to the system because the architecture of the circuit itself is changed by reconfiguration. In particular, in a reconfigurable system connected to the Internet, the system must be protected against malicious bitstreams sent by attackers.

To defend the system against malicious bitstreams, the confidentiality and integrity of a downloaded bitstream must be inspected so that only a proper circuit is configured in the system. In the prototype system, both correct and malicious bitstreams are processed in the same manner. To avoid a system halt, an authentication procedure to confirm whether the bitstream is admissible to the terminal is necessary.

A system failure can be caused by both malicious attacks and defects introduced during the design process. For example, the voltage of the I/O pins in the partial circuit can be incorrectly set by a designer. In this case, the bitstream is trusted by the server and, consequently, an erroneous bitstream cannot be eliminated by an authentication procedure. Therefore, it is important to inspect the conditions of the circuit *before* it is configured on the terminal. Information of the detailed structure of the bitstream is requisite for the prior inspection of the partial circuit.

8. Conclusions

We developed an FPGA-based content delivery system to securely distribute digital content on the Internet. The system effectively utilizes the partial reconfigurability of an FPGA to protect digital content. In the system, a partial circuit must be downloaded from the server to the client to play content. The content is properly played only when the downloaded circuit is correctly combined (= *interlocked*) with the circuit built in the terminal.

We implemented the interlocking mechanism on a Virtex-II Pro FPGA and tested the feasibility of the system. From the experimental results of the system, we conclude the current implementation as follows:

- The CSC-TBC interlocking mechanism can be utilized for controlling key generation and content decryption so that only the targeted content is played on the terminal.
- The I/O configuration of CSC and TBC can be utilized for terminal authentication so that content is played only on the targeted terminal.
- The Block-RAM-based switch provides important functions to realize a terminal specific I/O configuration and fault-tolerant architecture against erroneous CSC bitstreams.

Summing up, the interlocking mechanism with partial reconfigurability of an FPGA successfully works to control play of content. The prototype system, however, has some room to be improved for practical application.

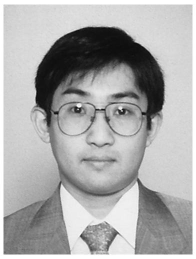
First, fail-safe architecture against malicious and erroneous inputs should be studied further. Since a content delivery system is necessarily connected to the Internet, the system is always threatened by attacks from malicious persons. Additionally, a system failure can be caused by not only malicious attacks but also design defects. Therefore a mechanism to defend the system against various unintended inputs are requisite for practical use of the system.

Second, cryptographic functions should be implemented on CSC and Authentication Block of TBC. As our main concern of the current implementation is with the CSC-TBC interlocking mechanism, security of a function itself is insufficiently discussed. One-way hash functions such as SHA-256 should be implemented on the modules.

References

[1] Digital Content Association of Japan, "Digital content white paper," 2006.
[2] A. Waller, G. Jones, T. Whitley, J. Edwards, D. Kaleshi, A. Munro, B. MacFarlane, and A. Wood, "Securing the delivery of digital content over the Internet," Electron. Commun. Eng. J., vol.14, no.5, pp.239–248, 2002.
[3] J. Lee, S.O. Hwang, S.W. Jeong, K.S. Yoon, C.S. Park, and J.C. Ryou, "A DRM framework for distributing digital contents through the Internet," ETRI J., vol.25, no.6, pp.423–436, 2003.
[4] J. Zhao, Y. Qi, and Z. Ma, "Secure multimedia streaming with trusted digital rights management," LCN'05, pp.817–821, 2005.
[5] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," Commun. ACM, vol.21, no.2, pp.120–126, 1978.
[6] National Institute of Standards and Technology, "Announcing the advanced encryption standard (AES)," FIPS PUB 197, Nov. 2001.
[7] R. Anderson, "Why cryptosystems fail," Commun. ACM, vol.37, no.11, pp.32–40, 1994.
[8] S.H. Weingart, "Physical security devices for computer subsystems: A survey of attacks and defenses," CHES'00, pp.302–317, 2000.
[9] M. Bond and R. Anderson, "API-level attacks on embedded systems," Computer, vol.34, no.10, pp.67–75, 2001.
[10] S. Smith, "Fairy dust, secrets, and the real world," IEEE Security & Privacy, vol.1, no.1-2, pp.89–93, 2003.
[11] R. Anderson, M. Bond, J. Clulow, and S. Skorobogatov, "Cryptographic processors—A survey," Proc. IEEE, vol.94, no.2, pp.357–369, 2006.
[12] H. Yokoyama and K. Toda, "FPGA-based content protection system for embedded consumer electronics," RTCSA'05, pp.502–507, 2005.
[13] Y. Hori, H. Yokoyama, and K. Toda, "Secure content distribution system based on run-time partial reconfiguration," FPL'06, pp.637–640, 2006.
[14] Y. Hori, H. Yokoyama, H. Sakane, and K. Toda, "Secure content distribution system with self run-time partial reconfiguration of an FPGA," IEICE Technical Report, CPSY2006-86, 2007.
[15] H. Yokoyama, Y. Hori, and K. Toda, "Design and implementation of FPGA-based content protection system," J. IPSJ, vol.48, no.9, pp.3253–3265, 2007.
[16] M. Ripley, C.B.S. Traw, S. Balogh, and M. Reed, "Content protection in the digital home," Intel Technology J., vol.4, no.6, pp.49–56, 2002.
[17] I. Gonzalez, S. Lopez-Buedo, F.J. Gomez, and J. Martinez, "Using partial reconfiguration in cryptographic applications: An implementation of the idea algorithm," FPT'03, pp.194–203, 2003.
[18] Digital Cinema Initiatives, LLC, Digital Cinema System Specification Version 1.1, April 2007.
[19] P. Lysaght, B. Blodget, J. Mason, J. Young, and B. Bridgford, "Enhanced architectures, design methodologies and CAD tools for dynamic reconfiguration of Xilinx FPGAs," FPL'06, pp.12–17, 2006.
[20] Xilinx, Inc., Virtex-II 1.5 V Field Programmable Gate Arrays v1.7, 2001.
[21] S. Trimberger, "Trusted design in FPGAs," DAC 2007, pp.5–8, 2007.
[22] Xilinx, Inc., Virtex FPGA Series Configuration and Readback, XAPP138, v2.8, 2005.
[23] Xilinx, Inc., Virtex Series Configuration Architecture User Guide, XAPP151, v1.7, 2004.
[24] Xilinx, Inc., Virtex-II Pro and Virtex-II Pro X Platform FPGAs: Complete Data Sheet v4.6, 2007.

Yohei Hori received his B.E., M.E. and Ph.D. degrees from the University of Tsukuba, Ibaraki, Japan in 1999, 2001 and 2004, respectively. He is currently a researcher of Real-Time Embedded Systems Semi-Group at National Institute of Advanced Industrial Science and Technology (AIST). His research interests include reconfigurable computing and real-time embedded systems. He is a member of IPSJ.



Hiroyuki Yokoyama received B.E., M.E., and Ph.D. degrees in Electrical Engineering from Kyoto University, Kyoto, Japan, in 1992, 1994, and 2006, respectively. He joined the Research and Development Laboratories of Kokusai Denshin Denwa Company Ltd. (currently KDDI Corp.) in 1994 and has engaged in research on communications network planning. His current research interests include service system architecture and terminal platforms of IP-based networks.

Hirofumi Sakane received his B.E. degree from Yamaguchi University in 1990 and M.E. degree from the University of Electro-Communications (UEC) in 1992. He has been working on parallel systems and logic emulation methods at the National Institute of Advanced Industrial Science and Technology (AIST). While working at AIST, he joined the doctoral program of the Graduate School of Information Systems of UEC in 1998 and received his Ph.D degree in 2001. He also studied an advanced logic emulation method at University of Delaware from 2002 through 2005. He is a member of IEEE-CS and IPSJ.

Kenji Toda received the M.S. degree from the Keio University, Japan in 1982. He is the leader of Real-Time Embedded System Semi-Group in National Institute of Advanced Industrial Science and Technology (AIST). His research interests are real-time computing, embedded systems, and network applications. He is a member of IPSJ.