HAL
open science

Automatic low-cost IP watermarking technique based on output mark insertions

Bertrand Le Gal, Lilian Bossuet

To cite this version:

Bertrand Le Gal, Lilian Bossuet. Automatic low-cost IP watermarking technique based on output mark insertions. Design Automation for Embedded System, Springer, 2012, 16 (2), pp.71-92. 10.1007/s10617-012-9085-y . hal-00753211

HAL Id: hal-00753211

https://hal.science/hal-00753211v1

Submitted on 18 Nov 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Des Autom Embed Syst
DOI 10.1007/s10617-012-9085-y

Automatic low-cost IP watermarking technique based on output mark insertions

Bertrand Le Gal · Lilian Bossuet

Received: 12 October 2010 / Accepted: 11 May 2012
© Springer Science+Business Media, LLC 2012

Abstract Today, although intellectual properties (IP) and their reuse are common, their use is causing design security issues: illegal copying, counterfeiting, and reverse engineering. IP watermarking is an efficient way to detect an unauthorized IP copy or a counterfeit. In this context, many interesting solutions have been proposed. However, few combine the watermarking process with synthesis. This article presents a new solution, i.e. automatic low cost IP watermarking included in the high-level synthesis process. The proposed method differs from those cited in the literature as the marking is not material, but is based on mathematical relationships between numeric values as inputs and outputs at specified times. Some implementation results with Xilinx *Virtex-5* FPGA that the proposed solution required a lower area and timing overhead than existing solutions.

Keywords IP protection · IP watermarking · War against illegal copying · Design automation · High level synthesis

1 Introduction

To cope with increasing system complexity, companies reuse more and more IP cores. As a consequence of the increase in the IP core business, IP theft is also on the rise [1]. The trade group founded by Cisco, HP, Nortel and 3COM ("*alliance for gray market and counterfeit abatement*" (AGMA) [2]) estimates that legitimate electronic companies loose almost $100 billion in revenues every year due to counterfeiting. The prevention of counterfeiting requires a technological solution in addition to the legal process i.e. patents and trade agreements.

B. Le Gal (✉)
IMS Laboratory—UMR CNRS 5218, ENSEIRB-MATMECA, University of Bordeaux, Talence Cedex, France
e-mail: bertrand.legal@ims-bordeaux.fr

L. Bossuet
Laboratoire Hubert Curien, UMR CNRS 5516, Telecom Saint Etienne, University of Lyon, Lyon, France
e-mail: lilian.bossuet@univ-st-etienne.fr

Published online: 29 May 2012

A novel IP watermarking technique to identify IP theft (i.e. an illegal copy of IP), is presented in this article. To reduce the watermarking overhead (area, delay, power consumption and design time), a watermark is automatically inserted in the design. The watermark is inserted during the behavioral synthesis process by using a automatic high level synthesis (HLS) tool.

HLS [3, 4] resembles the software compilation transposed to the hardware domain. The HLS tool automates the design process of generating the register transfer level (RTL) architectures from the behavior specifications for the system algorithm. These computer aided design (CAD) tools are necessary to tackle the actual area/throughput tradeoffs. Otherwise, it is impossible to solve this issue with common hand-coded designs that are based on massive pipeline hardware architectures. HLS tools that target ASIC and/or FPGA, have already been developed by companies like Mentor Graphics, *Catapult-C*, Hewlett-Packard *PICO-NPA* [5] and Forte Design Systems *Cynthesizer* [6]. Moreover, HLS tools developed by academic laboratories such as *SPARK* (University of Irvine, CA, USA) [7], *GraphLab* (University of Bordeaux, France) [8] and *GAUT* (University of South Brittany, France) [9] also contribute. HLS tools generate RTL-IPs described by VHDL or Verilog language. The IP designer uses generated RTL-IPs as inputs for place and route tools to generate an ASIC netlist or a FPGA bitstream.

This paper presents a design method that fulfils IP security requirements by design reuse, but in an original way. The paper is organized as follows. Section 2 describes state-of-the-art IP watermarking solutions. Section 3 presents the HLS concepts. Section 4 details the new proposal for IP watermarking illustrated in Sect. 5 by a didactic example. Section 6 describes the main parts of the modified HLS automatic flow. Section 7 gives experimental results with signal processing benchmarks on a Xilinx *Virtex-5* SRAM-based FPGA target. Finally, Sect. 8 discusses the watermarking chain from the point of view of security.

2 State of the art

2.1 IP protection by watermarking

According to [10] the goals of IP protection are:

1. To enable IP providers to protect their IPs against unauthorized use;
2. To protect all types of design data used to produce and deliver IPs;
3. To detect unauthorized use of IPs;
4. To trace unauthorized use of IPs.

Detecting the unauthorized use of IPs involves the ability to determine that an unauthorized use has occurred and then to trace the source of the theft. To solve the problem of detection, IP providers use an embedded digital signature. This is a finite sequence of symbols drawn from a finite alphabet. An IP digital signature system can use normal cryptographic services such as provider authentication (by using public key cryptography [11]) and signature integrity (by using hash function [12]). Nevertheless, for an embedded IP digital signature, cryptographic services are performed by pre-processing and they do not ensure the security of the signature itself.

Fingerprinting (passive watermarking) and watermarking are currently the best known digital signature solutions. Digital IP watermarking is an indirect protection scheme that proves the ownership of an IP. The concept of active watermarking consists of inserting a digital signature into an IP. This watermark makes use of the intrinsic features and architecture of the IP [10].

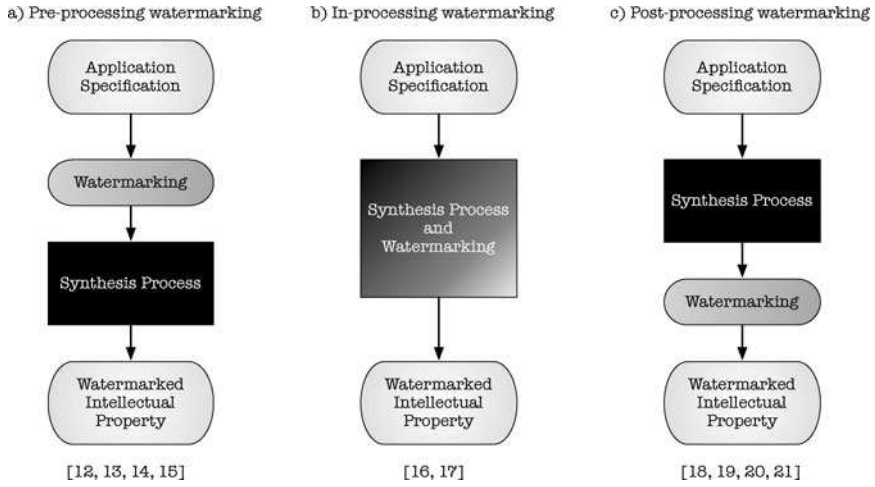

Fig. 1 Published watermarking schemes for IP protection with a synthesis process

Figure 1 shows published watermarking schemes according to the level of abstraction of the application or circuit. First, pre-synthesis IP watermarking leads to algorithmic modifications of the application to hide the signature (Fig. 1a). However, pre-synthesis IP watermarking is too algorithm dependent; it may be suitable for some digital signal processing (DSP) applications but it is not at all practical for rapid marketing. In-synthesis IP watermarking benefits from an automatic synthesis (behavioral or logic synthesis) tool to add a digital signature to the IP without significant overhead (Fig. 1b). Finally, post-processing IP watermarking uses the designer's knowledge of the circuit to change (by hand) the IP hardware architecture and add a digital signature (Fig. 1c). The following paragraph gives more details on the different published solutions.

2.2 Existing solutions

This section reviews some existing watermarking solutions. It is not exhaustive, and interested readers will find more information in [10] and [13].

Examples of pre-synthesis IP watermarking can be found in [14–17]. DSP IP watermarking is described in [14]. In [15], the authors target algorithm level IP watermarking in the design flow. Both approaches [14] and [15] are based on slightly changing the digital filters parameters, without affecting system behavior. Two different pre-synthesis IP watermarking techniques at the behavioral level are described in [16] and [17]. Both techniques are based on adding new input/output sequences to the IP finite state machine (FSM).

In-synthesis IP watermarking can be found in [18, 19]. In [18], Hong presents IP watermarking combinational logic synthesis solutions. IP watermarking behavioral synthesis techniques are described in [19].

Post-synthesis IP watermarking in [20] and [21] mostly describes constraints. One example is the addition of extra hardware, like a buffer [22] or a dedicated embedded tester [23].

Clearly, for the solutions listed above, extraction of the IP watermark is not described in detail. Watermark extraction and testing can be difficult, especially when the IP is embedded in a larger system on chip (SoC). Watermark extraction concerns only a small number of suspicious IPs in a court of law [13]. In this case, enough time is available to prove the legal-

ity of the IP (respect of copyright). Like extraction time, the cost of extracting a watermark is not prohibitive in the case of law.

2.3 Conclusion

Pre-synthesis IP watermarking techniques are application dependent, and over costs are hard to measure. Post-synthesis IP watermarking techniques are time consuming, handmade, and design/device dependent. In-synthesis IP watermarking techniques introduce a power/area/timing overhead. Generalizing watermarking usage for IP identification is important. However, it requires sufficient generic IP watermarking techniques with very low area and timing overheads. Such techniques must be implemented in automatic design flows for rapid component tagging in a designer friendly process. For these reasons, a new in-synthesis IP watermarking technique is presented in the following section.

3 Architectural synthesis concepts

The proposed methodology targets custom hardware IPs dedicated to computationally intensive applications (i.e. signal and video processing, digital communications, etc.). These IPs are composed of three units:

1. The processing unit contains the data path and a controller to perform the required computations.
2. The memory unit manages pipeline access to memories.
3. The communication unit sends and receives data to/from the input/output ports. It also manages internal communications.

This kind of architectural descriptions could be automatically generated in hardware description language (i.e. VHDL) from a high-level language (i.e. C language) by using a HLS tool. The objective of HLS tool is to exploit the application parallelism and to schedule computation and data storage. The HLS tool has to respect design constraints, including power consumption, area, and throughput. Usually, there is a trade-off between low area sequential hardware architecture and high-speed parallel (full pipeline) hardware architecture. The HLS tool uses a centric trade-off design flow, as illustrated in Fig. 2.

For example, in the case of customary video and signal processing applications, full pipeline hardware architecture is often inefficient. Actually, full pipeline hardware architecture is too area and power consuming.

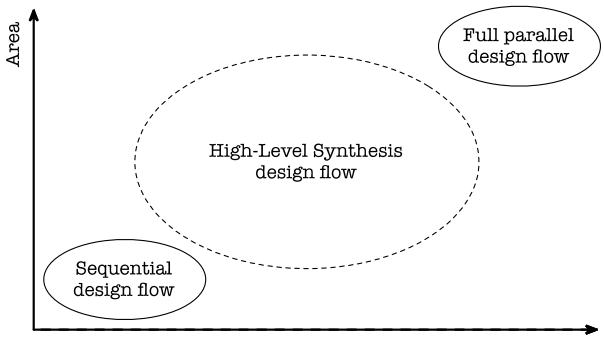

Fig. 2 High-level synthesis in a centric area-throughput trade-off design methodology

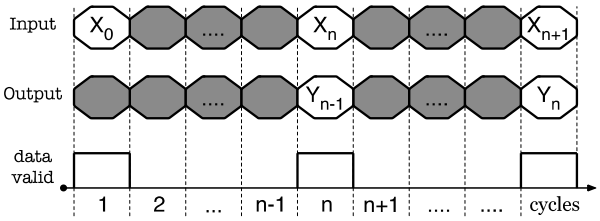

Fig. 3 Example of the input/output behavior of a digital FIT filter IP

The trade-off is between high performance and low cost (area, power consumption) hardware architectures [9] and [24]. An efficient architecture shares hardware resources (operators and registers) during the execution of an application. Sharing resources temporarily frees some input and output slots that are then ready to tag.

The inputs and outputs allow IP to receive and send data from/to the system. Figure 3 presents the input/output behavior of a digital FIR filter. It receives input data (X_n), performs computations and then provides output data (Y_n). The output time slots between two successive active states (high levels) of "*data valid*" signal are not used. Out of active states of "*data valid*", the "temporally free" output slots are colored gray in Fig. 3.

The proposed IP watermarking technique is to use the "temporally free" output slots to give watermarking computation results. The proposed IP watermark is a set of mathematical relations between the IP input data, the initial values of the internal computation and the IP output. Each mathematical relation is called a sub-mark.

The sub-marks are read like output data from some "temporally free" output slots. With the proposed technique, the IP watermark is invisible for the IP buyer, the IP integrator and the IP user. This is because the sub-marks results look like dynamic transient output data and the watermark area and timing overhead are very low (as will be shown in the experimental results in Sect. 7). Consequently, the IP watermark remains invisible during static analysis.

The proposed IP watermarking technique is suitable for general purpose applications like digital signal, image or video processing. This technique is not suitable for data-security applications such as data encryption, data integrity or data authentication. In such cases, the IP watermark can cause a dramatic data security failure by outputting internal computation data. The next section details the IP watermarking technique.

4 New IPP proposal by watermarking

4.1 Presentation of the IP watermarking technique

The IP watermarking technique is based on the "temporally free" IP output slot behaviors. During the "temporally free" output slots, output data are modified by introducing custom-design singularities (which are IP internal computation values).

The proposed technique uses the following assertion: the hardware IP has "temporally free" output slots. According to the presentation of HLS centric trade-off design methodology (cf. Sect. 3), this assertion is true with HLS tools.

Depending on the level of protection required and the watermark cost allowed, two IP watermarking algorithms (with different area and timing overheads) are proposed: a low-cost watermark and a costless watermark.

A low-cost watermark is characterized by a set of randomly chosen internal computation values of the IP. A special data path generates each sub-mark by transferring the selected

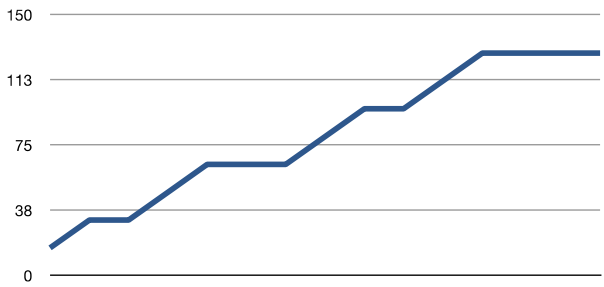

Fig. 4 Multiplexer area (number of LUTs, *y*-axis) on the Xilinx *Spartan-3E* device (90 nm SRAM FPGA) as a function of the number of input ports (*x*-axis)

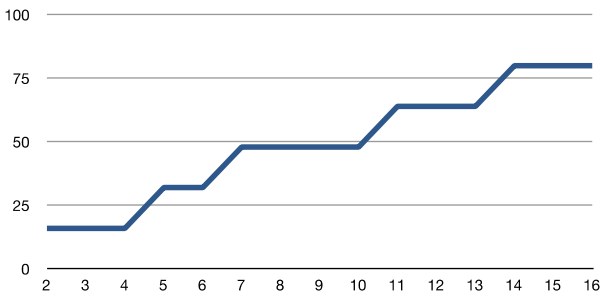

Fig. 5 Multiplexer area (number of LUTs, *y*-axis) on the Xilinx *Virtex-5* device (65 nm SRAM FPGA) as a function of the number of input ports (*x*-axis)

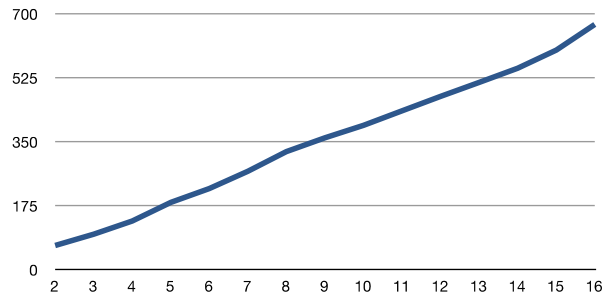

Fig. 6 Multiplexer area (2-input NAND gate equivalents, *y*-axis) on 65 nm low-power ASIC technology as a function of the number of input ports (*x*-axis)

internal values to a "temporally free" output slot. The IP watermarking area overhead is due to the cost of the data path area and output multiplexer resizing, but such area overhead is very low. This is particularly true for FPGA implementations, because, when an FPGA is used, the reconfigurable data paths do not cost area. Instead, the set of data paths is directly available in the FPGA. However, increasing input multiplexer size costs area. Figures 4 and 5 show that for a FPGA implementation, the multiplexer area (in the number of look-up tables (LUT)) depends on the number of inputs. These results were obtained using the Xilinx *ISE* 10.1 synthesis tool. Using the latest FPGA generations with larger LUT reduces the multiplexer cost and provides larger steps (see Figs. 4 and 5). Actually, the latest FPGA generations, such as Xilinx *Virtex-5* 65 nm FPGA and *Virtex-6* 40 nm FPGA, use 6-input LUTs. Whereas 90 nm FPGAs (such as Xilinx *Spartan-3E*) and other older FPGAs use 4-input LUTs. For ASIC implementation, Fig. 6 gives the multiplexer area (in the number of NAND gates), which increases linearly with respect to the number of inputs to the multiplexer. These results were obtained using the Synopsis *Design Compiler* synthesis tool.

According to these results, FPGAs are more suited than ASICs to using the proposed IP watermarking technique with low area overhead. FPGAs are more suitable because of their

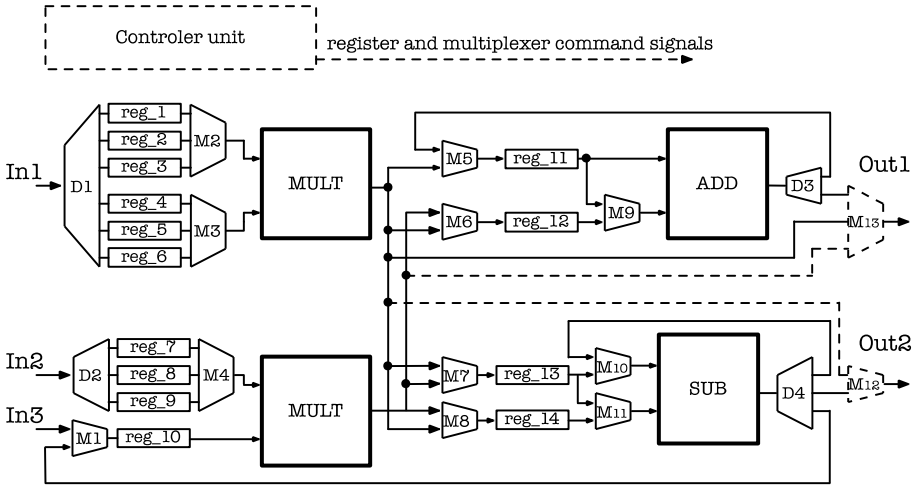

Fig. 7 Example of low-cost IP watermarked architecture reusing existing resources, including new dedicated data path allocations (*dotted line*)

multiplexer area-cost function: adding an input to an existing multiplexer does not always increase its area (number of LUT elements). This characteristic (shown in Figs. 4 and 5 for Xilinx *Spartan-3E* and *Virtex-5*, respectively) is due to the internal structure of the FPGA LUT.

A costless watermark is a low-cost watermark with a reduced set of internal values. A costless watermark uses dynamic transient outputs when the slots are temporarily free. To introduce this watermark, the only change required is modifying the IP control unit to drive selected data to output ports. HLS tools design the IP control unit with a FSM. The costless watermark only impacts the FSM. The concept will be illustrated in Sect. 6, with experimental results on the FPGA target. The results ensure that the modifications are costless and may even reduce the area occupied by the IP. Actually, the area increases or decreases depending on the FSM modification and the logical-synthesis process. Hence, it is hard to predict.

Using a didactic example, Figs. 7 and 8 show the main distinction between low-cost and the costless IP watermarked architecture. In both figures, the necessary modifications to the IP architecture are represented by dotted lines. Figure 7 shows the low-cost IP watermarking technique, which allows the HLS process to allocate new data paths (multiplexers and wires). The new data path drives the internal computation data to selected outputs. Figure 8 presents the costless IP watermarking technique, which only affects the IP design controller (FSM).

4.2 Proposed IP watermarking technique terminology

To formalize the proposed IP watermarking technique, this subsection defines the terminology used.

Sub-mark, $M_i(n)$, defined in (1), is a mathematical relation F_i, for the nth sample, between the IP inputs $in(i)$ and the IP internal computation initial values *InitialValues*. The sub-marks result is an internal data (low-cost IP watermark) or an output value (costless IP

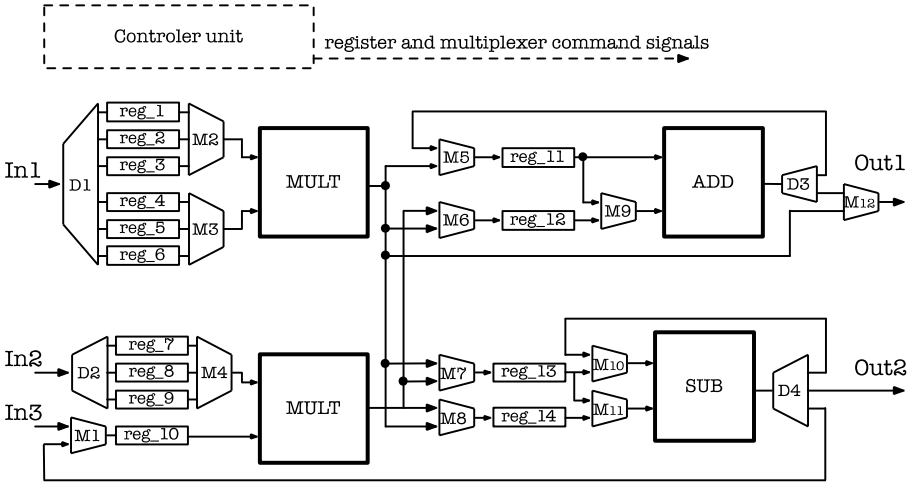

Fig. 8 Example of costless IP watermarked architecture reusing the existing path controller by modifying the multiplexer controller (*dotted line*)

watermark). This sub-mark is used like an IP implementation singularity.

$$M_i(n) = F_i(in(n), InitialValues) \tag{1}$$

IP watermark defined in (2) is a set of two-data elements:

- The sub-mark $M_i(n)$,
- The clock cycle number C_i, during which the sub-mark is sent to a "temporally free" IP output port.

The IP watermark is composed of 1 to p sub-marks, p range is 1 to k, where k is the number of IP "temporally free" output slots during execution of an application.

$$Watermark = \left\{\left(M_1(n), C_1\right), \ldots, \left(M_p(n), C_p\right)\right\} \quad \text{with } p \in [1, k] \tag{2}$$

Watermark results are the IP outputs values generated on IP "temporally free" output slots. Watermark results depend on IP input data and on the IP initial internal computation conditions. In addition, watermark results depend on the sub-mark. Without specific knowledge, nothing differentiates the watermark results from normal non-marked IP output data during "temporally free" output slots, as these output data evolve "freely". The watermark results are the data used to prove IP ownership.

Watermark length, W_{length}, is characterized by the maximum number of sub-marks used for the IP watermarking.

Number of possible watermarks, W_{count}, depends on the number of possible usable data (internal computation data) at each clock cycle; $var(m)$, where m is the clock cycle number. W_{count} also depends on the number of output ports, $out(m)$. With h clock cycles ($h \geq W_{length}$), W_{count} is defined by (3).

$$W_{count} = \prod_{m=0}^{h-1} \mathrm{var}^{out(m)} \tag{3}$$

To further illustrate the proposed approach, a didactic IP watermarking example is given in the next section.

5 Didactic example of IP watermarking

To illustrate the proposed IP watermarking technique an 8-tap digital FIR filter is used. Figure 9 shows the filter i.e., CDFG model and input-output behavior. The four dotted edges represent the data links between the internal selected computation data and the output slots. Each data link is a sub-mark, and W_{length} is equal to four. Information C_n on the left-hand side of the graph model provides the control step in which operation nodes are scheduled.

For the example given (an 8-tap FIR filter), the four mathematical relations corresponding to the four sub-marks are given below:

$$M_1(n) = x_2 \times h_5$$

$$M_2(n) = (x_n \times h_7) + (x_1 \times h_6)$$

$$M_3(n) = (x_6 \times h_1) + (x_7 \times h_0)$$

$$M_4(n) = (x_2 \times h_5)$$

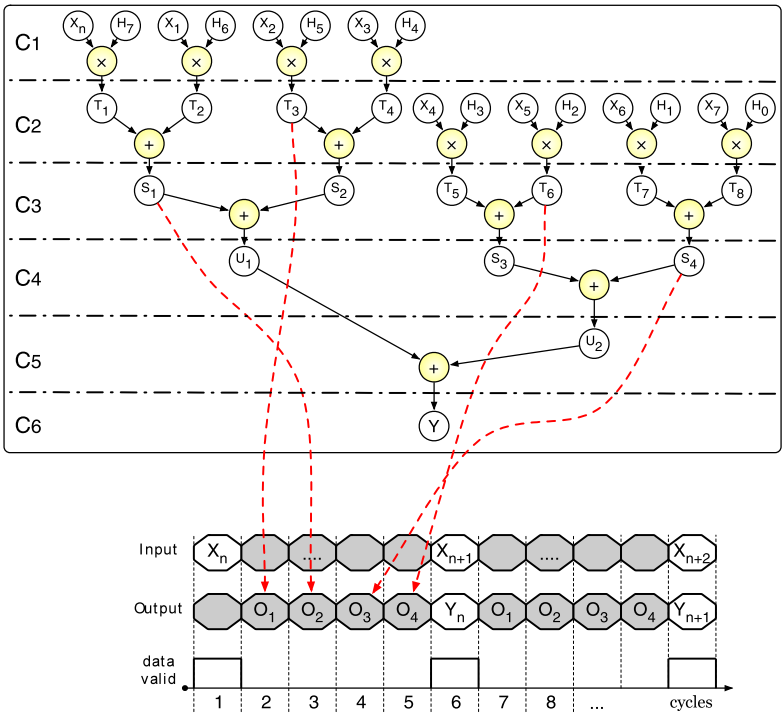

Fig. 9 Illustration of data links (*dotted edges*) between the computation internal data and the IP "temporally free" output slots used to watermark the IP, for an 8-tap digital FIR filter

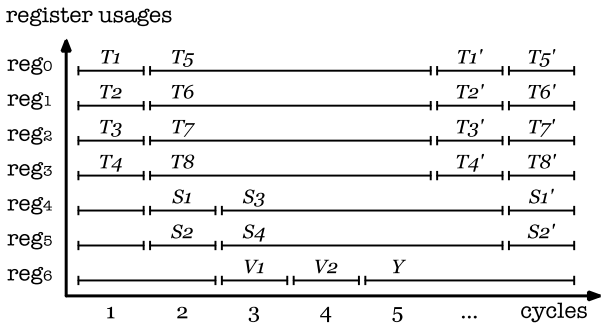

Fig. 10 Register usages and internal data availabilities depending on the clock cycle

For a given set of inputs $(x_n, x_1, x_2, x_6, x_7)$, O_i is the output of sub-mark $M_i(n)$, as shown in Fig. 9.

According to (2), the IP watermark is defined by the following set of two-data elements:

$$IPwatermark = \{(M_1(n), C_1), (M_2(n), C_2), (M_3(n), C_3), (M_4(n), C_4)\}$$

The example in Fig. 9 is quite simple and the number of possible watermarks is low. The number of useful computation internal data for the IP watermarking process is 23 $(X_n, x_1, \ldots, x_7, T_1, \ldots, T_7, S_1, \ldots, S_4, U_1, U_2, Y)$. Constant coefficients $(h_0, \ldots, h_7)$ are intentionally discarded. The number of output ports is equal to 1 and the number of clock cycles that are usable for watermarking is equal to 4 (4 "temporally free" output slots are available when *data-valid* is inactive). Nevertheless, as shown in Fig. 10, the overall internal values are not available at each clock cycle. Due to register sharing, internal data remain in registers until a new memorization is performed. Figure 10 models such phenomena i.e. T_1 is available at only one clock cycle due to T_5 memorization, though T_5-which is used only once- is stored for 4 clock cycles (until the next T_1 computation, which produces T_1').

Following (3), the number of possible low-cost watermarks is computed for the 8-tap digital FIR filter example. This results in $W_{count} = 50625$ i.e. $W_{count} \geq 215$. However, more a complex application would produce a higher number of possible distinct watermarks.

For the 8-tap digital FIR filter example, the number of possible cost-less watermarks is 4 (1 usable output, Y, for 4 clock cycles) and so is not sufficient.

The number of possible low-cost and costless watermarks for the implementations of some larger DSP applications was evaluated and is listed in Table 1. Synthesis results were obtained using an HLS tool named *GraphLab* [25], which includes the watermarking technique. The costless columns in Table 1 correspond to the maximum number of watermarks that can be obtained without modifying the IP data path (multiplexer allocation). Conversely, the low-cost columns in Table 1 give results with multiplexer allocation (creation of one path) for each IP output. The results in Table 1 show that (i) the proposed IP watermarking technique may provide enough different watermarks for design implementation to avoid IP watermark collisions (ii) the number of watermarks depends on the IP architecture.

The results in Table 1 are for a known HLS tool. Such a tool accepts various synthesis constraints and options in order to respect system integration constraints. Each constraint may produce distinct IP hardware architecture. The architectural variation results from the use of a timing constraint vs. an area constraint [3], scheduling and binding algorithms [26], transformation of word length [25], power optimization [27], graph transformations [28], multi-mode design [29], etc. Each synthesis constraint produces a new set of watermarks. For example, changing the timing constraint for a digital FIR filter modifies the number of

Table 1 Evaluation of possible costless and low-cost watermarks for several DSP applications

Application	# of FSM states	# of I/O ports	Maximum mark length	Watermark length = 50 %		Watermark length = 100 %	
				# of cost-less watermarks	# of low-cost watermarks	# of cost-less watermarks	# of low-cost watermarks
FIR 64-taps	26	1/1	25	2^{13}	2^{46}	2^{13}	2^{50}
	38	1/1	37	2^{34}	2^{70}	2^{21}	2^{74}
LWT 16-taps	25	2/2	34	2^{63}	2^{72}	2^{68}	2^{87}
	64	2/2	114	2^{278}	2^{317}	2^{342}	2^{421}
SSD 16 × 16	35	8/1	34	2^{31}	2^{65}	2^{25}	2^{68}
	81	1/1	80	2^{76}	2^{156}	2^{37}	2^{160}
1d DCT 8 taps	15	4/4	56	2^{80}	2^{125}	2^{56}	2^{144}
2d DCT 8 × 8 taps	20	1/1	13	2^{22}	2^{26}	2^{26}	2^{33}
	80	8/8	584	2^{866}	2^{1323}	2^{584}	2^{1509}
Matrix product 8 × 8	160	1/1	97	2^{362}	2^{346}	2^{544}	2^{547}
	86	8/4	280	2^{275}	2^{880}	2^{1120}	2^{1210}
	141	1/1	77	2^{292}	2^{443}	2^{293}	2^{445}
FFT 64 taps	90	8/8	600	2^{1400}	2^{1634}	2^{1632}	2^{2106}
	180	2/2	234	2^{796}	2^{814}	2^{1142}	2^{1180}

"temporally free" output slots. Thus, the possible number of watermarks is changed by the design exploration.

6 IP watermarking automation flow

6.1 Specifying the mathematical watermark parameters

The proposed technique is a part of an HLS tool. This tool allows the designer to automatically include the IP watermark. To do this, the IP designer has to provide the following watermarking constraints:

- The mathematical watermark length (W_{length}),
- The number of clock cycles to mark (h in (3), $h \geq W_{length}$),
- The watermarking technique algorithms; costless or low-cost.

After the automatic IP watermarking step, the tool provides the IP designer with a file containing the IP watermark characteristics. This file contains the output time slots that match the time the sub-mark results produced. This file also contains the mathematical relations that describe the sub-marks. The IP watermark characteristics may be kept secret from a consumer, as they are only used to prove circuit ownership. As proposed in [11] and [12], the IP designer can securely store the IP watermark characteristics by using cryptographic services.

Note that the IP watermarking is possible only if enough "temporally free" IP output slots are available. As a consequence, if the IP watermarking constraints (W_{length}, h) provided by the IP designer do no match the IP hardware architecture (i.e., a limited number of "temporally free" IP output slots), a watermarking process error will appear.

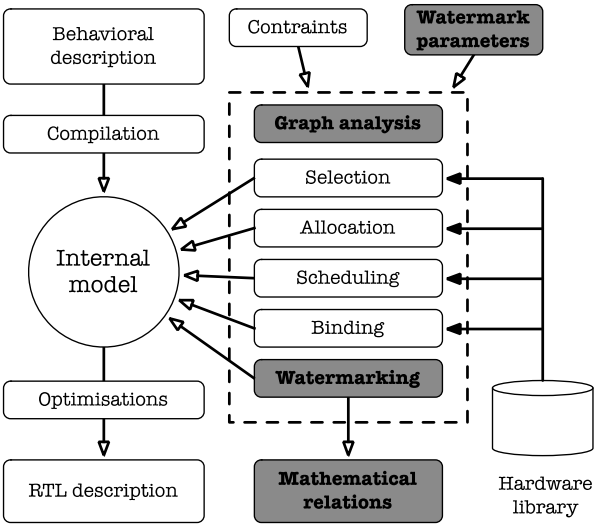

Fig. 11 HLS flow including the proposed watermarking technique (in gray). *In the figure, it should be* optimization

6.2 HLS flow modifications

Figure 11 presents the HLS flow modifications (original HLS flow can be found in [30]). The IP watermarking process includes four steps:

- graph analysis, to find the usable internal data,
- watermark enumeration, to determine W_{count},
- selection of internal values,
- IP architecture modifications: addition of data paths, multiplexer sizing and changing the FSM.

In the experiments presented in Sect. 7, the watermarking process takes less than 1 % of the total HLS runtime.

6.3 Selecting the watermarks and modifying the design

By using the IP designer parameters provided, an automatic process analyzes the number of possible watermarks. Depending on this result, it computes the average number of watermarks to introduce per clock cycle. The watermarks are then distributed randomly to tag the required number of clock cycles. Once these computations are performed, a mapping algorithm is applied to detect remarkable internal data from the design (all internal data are considered by the low-cost algorithm, only transient output data are considered by the costless algorithm). The mapping algorithm is applied (i) to select internal data for sub-mark usage (ii) to drive the selected internal data to one of the output ports.

For each internal data mapping to output, the tool analyzes the required logical glue overcost in order to find the best pair (sub-mark, clock cycle) to reduce the area overhead. An overview of the IP watermarking algorithm is provided in Fig. 12 for both low-cost and costless solutions.

This process is repeated for each sub-mark that the tool must insert in the design to respect the W_{length} constraint given by the IP designer.

```
outList ← IdentifyDesignOutputsPorts (graph)
fosList ← SearchFreeOutputSlotsDuringExecution (outList, graph)
if CountSlots(fosList) < W_count or CountClockCycles(fosList) < h then
    return false
end if
ufosList ← RandomlySelectFreeOutputSlots(W_count, h)
regList ← SearchRegistersHavingAnExistingPathToOutputs(graph, ufosList)
if LowCostMode = true then
    regList ← regList + RegistersWithoutExistingPathToOutputs(graph, ufosList,
    authorized_cost)
end if
dataList ← ListUsableInformationFromRegisters (regList, ufosList)
ufosList ← RandomlySelectDataForOutputWatermarking (dataList, W_count, h)
ModifyCircuitAccordingToWatermarkingChoices (graph, ufosList)
StoreRelationsBetweenInternalComputationAndOutputs (ufosList, filename)
return true
```

Fig. 12 Generic costless and low-cost IP watermarking algorithm

6.4 Authorized detection of the IP watermark

Once an IP watermark has been inserted, it must enable a check to be made for an illegal IP copy. A suspicious IP can be embedded in a larger system and glue logic can be added to it. Thus, it may seem hard to isolate the IP and to check the IP watermark to check if the IP is an illegal copy. However, it is shown that with the RTL IP it is always possible to find and test the IP original output. Moreover, a watermark is used to prove the copy is illegal in a court of law. So, the IP provider only tests a small number of suspicious IPs. Consequently, the watermark extraction time and cost are not considered to be serious drawbacks to the method.

Random inputs and the IP watermark characteristics (given in the IP watermark file provided by the HLS tool) allow the IP provider to check the input-output relations. This process is described in Fig. 13. The equality-checking tests the "temporally free" IP output slots to validate the desired mathematic relations that the HLS tool has created to watermark the IP.

7 Results

7.1 Experimental results

To evaluate the area and timing overhead of the proposed IP watermarking technique, experiments were conducted with signal and image processing benchmark FPGA. Results of costless IP watermarking implementation are presented in Table 2. Table 3 presents results of low-cost IP watermarking implementation.

For each IP, the following parameters are provided: the number of FSM states, the number of I/O ports, the number of "temporally free" output slots, the length of the introduced watermark (0 % for the reference design, 50 % or 100 % for watermarked ones). In both Table 2 and Table 3, the right-hand columns list the area and timing overhead. Penalties for watermarked IP are obtained from a comparison with the unprotected IP. Logical synthesis results were obtained using the Xilinx *ISE 10.1* tool.

Area and timing overhead depend on the type of watermark (low-cost or costless). As shown in Sect. 3, area and timing overheads result from changes to the data path (some multiplexers are allocated) and from changes to the control unit (the FSM instruction decoder is modified to drive data and control new multiplexers).

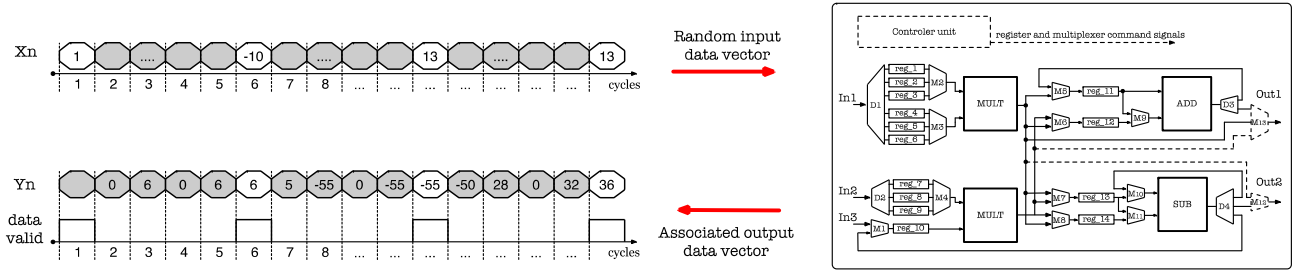

Fig. 13 Checking an IP watermarking using known or random values to extract its input-output behavior

Table 2 Proposed costless IP watermarking area and timing overhead with a XILINX *Virtex-5* SRAM FPGA

Application	# of FSM states	# of I/O ports	# of free slots	Without watermark		Watermark length = 50 %		Watermark length = 100 %	
				Area (# slices)	Critical Path (ns)	Area overhead	C. Path overhead	Area overhead	C. Path overhead
FIR 64-taps	26	1/1	25	12351	15.499	0.05 %	0.01 %	0.02 %	−0.01 %
	38	1/1	37	6612	14.613	−0.02 %	0.00 %	0.05 %	0.01 %
LWT 16-taps	25	2/2	34	14079	16.312	0.04 %	−0.27 %	−0.18 %	−1.31 %
	64	2/2	114	12028	16.346	−0.36 %	1.05 %	0.17 %	−0.61 %
SSD 16 × 16	35	8/1	34	11078	16.103	0.09 %	0.00 %	−0.01 %	0.00 %
	81	1/1	80	3193	15.689	−0.06 %	0.00 %	0.09 %	0.00 %
1d DCT 8 taps	15	4/4	56	8818	15.185	0.10 %	−0.05 %	0.15 %	−0.02 %
	20	1/1	13	6384	14.991	0.03 %	−0.03 %	0.03 %	−0.03 %
2d DCT 8 × 8 taps	80	8/8	584	31428	17.259	−0.32 %	0.41 %	0.02 %	−0.24 %
	160	1/1	97	25469	17.355	−0.30 %	−0.06 %	0.08 %	−0.59 %
Matrix product 8 × 8	86	8/4	280	62784	16.104	−0.24 %	0.63 %	−0.11 %	0.71 %
	141	1/1	77	31117	17.201	−0.01 %	0.34 %	0.05 %	0.13 %
FFT 64 taps	90	8/8	600	51086	17.255	0.02 %	−0.01 %	0.04 %	0.05 %
	180	2/2	234	31589	17.181	0.04 %	0.02 %	0.17 %	−0.84 %

Table 3 Proposed low-cost IP watermarking area and timing overhead with a XILINX *Virtex-5* SRAM FPGA

Application	# of FSM states	# of I/O ports	# of free slots	Without watermark		Watermark length = 50 %		Watermark length = 100 %	
				Area overhead	Critical Path (ns)	Area overhead	C. Path overhead	Area overhead	C. Path overhead
FIR 64-taps	26	1/1	25	12351	15.499	0.08 %	−0.30 %	0.17 %	−1.24 %
	38	1/1	37	6612	14.613	0.35 %	0.46 %	0.83 %	0.53 %
LWT 16-taps	25	2/2	34	14079	16.312	0.57 %	−1.29 %	0.77 %	0.69 %
	64	2/2	114	12028	16.346	0.59 %	−0.32 %	0.64 %	−0.18 %
SSD 16 × 16	35	8/1	34	11078	16.103	0.15 %	0.98 %	0.55 %	−0.44 %
	81	1/1	80	3193	15.689	0.13 %	1.45 %	0.91 %	0.25 %
1d DCT 8 taps	15	4/4	56	8818	15.185	0.68 %	−0.97 %	1.00 %	1.13 %
	20	1/1	13	6384	14.991	0.22 %	−0.07 %	0.34 %	−0.03 %
2d DCT 8 × 8 taps	80	8/8	584	31428	17.259	0.71 %	−0.61 %	1.02 %	0.75 %
	160	1/1	97	25469	17.355	0.13 %	−0.14 %	0.37 %	−0.82 %
Matrix product 8 × 8	86	8/4	280	62784	16.104	0.07 %	0.29 %	0.13 %	0.71 %
	141	1/1	77	31117	17.201	0.07 %	0.66 %	0.08 %	0.13 %
FFT 64 taps	90	8/8	600	51086	17.255	0.10 %	−0.12 %	0.24 %	0.01 %
	180	2/2	234	31589	17.181	0.17 %	0.13 %	0.62 %	−0.33 %

Table 2 shows that costless IP watermarking has a very low impact on global IP characteristics. IP area ranges from *−0.36 %* up to *0.17 %* when the IP critical path progresses from *−1.31 %* to *1.05 %*. However, Table 1 (Sect. 5) shows that the watermark length is much shorter (not appropriate for high-level IP security).

For some designs, such as *SSD* 16 × 16, IP watermarking reduces area and timing costs. This results from FSM signal command modifications which may unintentionally bring about a better logical function simplification during logical synthesis.

With low-cost IP watermarking, the increase in the area and timing overheads depends on the watermark length as presented in Table 3. The design area overhead ranged from *0.07 %* to *1.02 %* while the timing overhead ranged from *−1.29 %* to *1.45 %*. In these experiments, in the watermarking process, maximum area overhead was limited to one multiplexer cost for each design output. However, the IP area overhead was larger than allocated multiplexer area costs due to new controller signals in the FSM. Unlike costless IP watermarking, according to Table 1, the watermark length is suitable for the use of low-cost IP watermarking with high level IP security.

The power consumption overhead, which is not detailed in Table 2 and Table 3, is very close to the area overhead (but always lower). Actually, in the worst case, IP watermarking uses a few multiplexers, and requires a limited number of new control signals in the FSM. Moreover, the static and dynamic power consumption of multiplexers is low compared to arithmetic resources i.e. multipliers.

These experimental results confirm the interest and the low cost of the two proposed IP watermarking algorithms.

7.2 Comparison with previously published automatic IP watermarking schemes

As explained in Sect. 2.2, pre-synthesis and post-synthesis hand-made IP watermarking techniques are not suitable for efficient IP protection. Pre-synthesis solutions are strongly dependent on the algorithm and the watermark is hard to detect during legal checking. Post-synthesis solutions are difficult to implement at the layout level. The IP designer needs to have a good technical knowledge. As a result, targeting an automatic watermarking scheme embedded in a high-level synthesis tool seems to be the best solution. Consequently, this section compares the proposed IP watermarking techniques with the most appropriate automatic in-synthesis IP watermarking (in-logical-synthesis [18] and in-behavioral-synthesis [19]).

In [18], Kirovski et al. suggest embedding specific information in a logic network while performing multi-level logic minimization and technological mapping. The copyright information is hashed using hash function to create an initial vector used to seed a pseudo-random number generator. The resulting stream of pseudo-random bits is used to generate a unique set of design constraints. Some logic gate outputs are pseudo-randomly chosen to be assigned to an additional dummy logic network. In a small IP, this dummy logic network can be easily detected by an attacker. But, if the IP is embedded in a larger system, by the authors' own admission, the detection of the forensic watermark is very hard. Moreover, added synthesis pseudo-random constraints can be simplified by the synthesis process. As a result, some of the watermark information can be deleted by the logic optimizations. Nevertheless, these added synthesis constraints could decrease the IP performances. Kirovski et al., do not provide performance results in terms of throughput overhead to confirm this assumption. Finally hash function and a pseudo-random generator usage, contrary to what these authors say, do not secure the watermarking scheme at all (see Sect. 8 for a security analysis).

In [19], Koushanfar et al. suggest adding edges to the colored interval graph for scheduled application CDFG. These added edges make it possible to embed a signature in the register allocation solution. The signature number of bits can be high for a 2000-node graph,

Table 4 Comparison of the performances of automatic IP watermarking techniques during the synthesis process for a significant application such as a DCT 2D

Watermark scheme	Graph modification	Watemark length (# bits)	Area overhead	Timing overhead	# possible marks
[18]	273 added logic edges	256	4.40 %	–	2^{1E637}
[19]	18 170 added edges	2047	–	–	2^{1E25}
Our work (costless)	None	584	0.02 %	−0.24 %	2^{584}
Our work (low-cost)	Datapath	584	1.02 %	0.75 %	$2^{1.5E3}$

Koushanfar et al. showed that they can embed between of the graph 2047 and 16383 bits. Nevertheless, this method seriously increases the complexity. For a DCT 2D, this watermarking scheme adds more than 24 thousand edges in the primary colored graph. The increase in number of edges impacts the synthesis results due to synthesis optimization and register allocation. As a result, such graph modification lead to significant area and timing overheads. Unfortunately, the authors do not provide any information about hardware overheads. Unlike their work, the watermarking technique proposed in this article does not alter the application graph (CDFG or colored graph). To make the best synthesis optimization, the proposed technique acts only after a preliminary CDFG scheduling and register allocation.

Comparing the proposed IP watermarking technique with the two works presented above is hard. First, each published work gives results focused on different aspects of performance: modifications to the application graph, watermark length, number of possible watermarks, area and timing overheads. Second, the benchmarks and synthesis tools used are too different to allow accurate comparison. Nevertheless, we tried to collect some information on each watermark technique with a significant benchmark. We chose a complex algorithm such as DCT 2D to make the comparison. Concerning the previous works, more information on the experimental results and benchmark can be found in the literature [18, 19]. Table 4 presents the result of the previously presented watermarking techniques and the proposed technique (with costless and low-cost solutions). The proposed technique has less impact on IP area and does not generate significant timing overheads. This is mainly due to the fact that the proposed IP watermarking technique drives any application graph modification before the first synthesis optimization. Whereas the watermarking schemes proposed in [18] and [19] increase graph complexity. As a result, these two watermarking schemes alter synthesis optimization. Nevertheless, using the two watermarking schemes in the literature, the number of possible watermarks is larger than with the method proposed here. As a result, an interesting tradeoff between the number of possible watermarks and the area/timing overhead is possible with these four techniques.

8 Security analysis

8.1 Analysis—typical attacks

There are several general ways to attack the proposed IP watermarking. Here, we discuss the most serious ways: *overwriting output*, *tampering*, *reverse engineering*. We analyze these attacks using the following scenario: Eve legally purchased an IP from Alice. Alice protected the IP using the proposed watermarking technique. This IP is provided as a placed and routed netlist. Eve wants to sell an unauthorized copy of the IP to Bob. Without an attack,

the unauthorized copied IP embeds Alice's watermark. To hide her dishonesty from Bob and Alice, Eve tries to attack the embedded watermark. The aim of Eve's attack is to mask or to (partially or completely) remove Alice's IP watermark. Depending on her technical knowledge, she can use three kinds of attack (the first is knowing that the IP has embedded a watermark and the type of watermark used by Alice), her technical facilities, her time to perform the attack, and the money she has to spend. Possible attacks by Eve are the following:

(1) *Overwriting output attack*: It is relatively simple for Eve to overwrite IP outputs by using additional logic to replace the "temporally free" output slot values by fixed or random values. Eve has to re-design an IP. It is made up of Alice's IP and the additional logic to hide/overwrite the watermark results. This simple attack does not threaten Alice's IP watermark. Actually, Eve has not destroyed the embedded watermark (remember watermarks are mathematical relationships between numeric value as inputs and outputs) but only overwritten the watermark results. As long as the watermark is not destroyed (or removed), in a court of law it is easy to bypass the additional logic used by Eve and find Alice's IP watermark. The main legal difference between Eve's attack and watermarking checking performed by Alice, is that in the second situation Alice can use all her knowledge such as knowledge of the original IP layout And the IP watermark characteristics.
(2) *Tampering with watermarking*: Ideally, such tampering would completely remove Alice's IP watermark and add Eve's own watermark. Removing an embedded watermark requires working on the back-routed circuit description. Nevertheless, watermarking is not located in an easily erasable element. The proposed watermarking scheme distributes the watermark everywhere in the IP. Actually, the watermark is located in the control unit (by changing FSM state and output register) and in the data path. As a result, it is unlikely that Eve can easily tamper with the watermark. In order to perform this kind of attack, Eve has to first perform reverse engineering. Such an attack is more difficult and is described in the 3) below.
(3) *Reverse engineering attack*: Here we consider that Eve has sufficient knowledge, facilities, time, and money to make it possible for her to reverse Alice's IP design. First, Eve hopes to earn more money through sales of the illegal IP copy than the cost of the attack. Second, in order to remove Alice's IP watermark, Eve has to completely re-design Alice's IP. As a result, the removal of the watermark results in a task that is as difficult as completely designing the specified functionality. Reverse engineering is a strong attack; nevertheless Eve's profit is reduced by the cost of and time required for the attack. We do not say that the attack is impossible, but it is unlikely to occur in normal circuits. It is a serious threat for complex expensive circuits and specific circuits (dedicated to cryptographic applications, for example). In the latter case, all published IP watermarking solutions do not work everywhere and more robust security solutions have to be used (such as ciphering the FPGA bitstream [31]). It is important to grasp that watermarking is a solution only against illegal copying and not against IP reverse engineering.

8.2 Analysis—security properties

To be validated from a security point of view, the watermark must have a number of security properties. Those used for applications such as multimedia [32] can easily be adapted to the field of IP protection:

(1) *Security does not reside in the secrecy of the algorithm*. As long as an attacker does not know the mathematical relationships that form the sub-marks, he cannot know the

sub-marks. The watermarking design algorithm does not fit the watermark secret. It is the watermark itself that contains the secret as an encryption key.

(2) *The level of trust in the watermark and its detection is high*. The watermark inserted in the IP with the proposed method is independent of the target technology. The watermark is independent of the environment in the normal operation of the device. The watermark is very reliable and can be checked very quickly. Thus, with a large number of tests, the IP provider can be sure of the watermark value and hence of the evidence it provides.

(3) *The watermark does not affect the functionality of the IP*. The watermark inserted in the IP does not affect its operation. Internal processing is not affected. This is fully transparent during the use of the IP.

(4) *It is not possible to change or remove the watermark*. Modifying the watermark requires acting on the back-routed IP. In the event that an attacker detects a watermark inserted in an IP following the proposed process, he cannot remove the routed-&-placed IP since the watermark is diffuse. Indeed, watermark inclusion in the IP is not located in an easily erasable element. The only option for the attacker would be to put a shield between the outputs of the illegal IP copy before selling it. In this case, the detection of the watermark is more difficult. However, in a court of law, during an expertise on the detection of the mark, the technical knowledge and the time devoted to this task are not limited.

(5) *The amount of information contained in the watermark is sufficient*. As we have shown, the quantity of information in the watermark depends on the application. Once implementation is a little complex, the number of available watermarks and their sizes are very important. Practically speaking, there is no significant obstacle to establishing watermarks of sufficient size.

(6) *The cost of watermarking is low*. The results of implantations clearly demonstrate that the proposed technique is very cheap and has a limited impact on IP performance (no change of latency).

(7) *Detection and tracing of the watermark is easy*. With the file containing the watermarking characteristics (provided by the HLS tool at the end of the watermarking process), legal detection of the watermark is very simple. Some combination of input data from an initial state leads to some changes in outputs. During some "temporally free" output slots, these changes depend directly on the sub-marks. It is not necessary to undertake a physical action on the component (unless a shield has been positioned to mask the watermark results). The detection of the watermark is simple: it only requires fast, inexpensive equipment.

(8) *During the design of the brand, the security level is high*. The design of the watermark lies in the *GraphLab* tool, since it is this tool that automatically computes and inserts the watermark. The IP designer is subsequently responsible for secure storage of the watermark by using cryptographic services.

In conclusion, the proposed solution meets the essential security criteria for a secure watermarking system able to protect an IP against the sale of unauthorized copies.

9 Conclusion

In this paper, a new IP watermarking technique to embed in a HLS flow has been presented. The proposed technique is designed for automatic IP protection using HLS tools. The essence of this new approach is the set of mathematical sub-marks on the design output ports that encode the IP watermark. The mathematical sub-marks are selected and inserted automatically during the synthesis process. This is done in such a way that they result in

the minimal hardware overhead while embedding the watermark. Finally, the watermark is difficult to detect and to remove. IP protection is a hot issue and is taking up more and more room in the industry. In this context, the proposed method promises to be a very attractive low-cost solution.

References

1. Pecht M, Tiku S (2006) Bogus! Electronic manufacturing and consumers confront a rising tide of counterfeit electronics. IEEE Spectr 43(5):37–46. Available from: http://www.spectrum.ieee.org/print/3423
2. http://www.agmaglobal.org (2012)
3. Gajski D et al (1992) High-level synthesis: introduction to chip and system design. Kluwer Academic, Dordrecht
4. Coussy P, Morawiec A (eds) (2008) High-level synthesis from algorithm to digital circuit. Springer, Berlin. ISBN: 978-1-4020-8587-1
5. Aditya S, Kathai V (2008) Algorithmic synthesis using PICO, high-level synthesis from algorithm to digital circuit. In: High-level synthesis from algorithm to digital circuit, vol XVI, pp 53–74
6. Meredith M (2008) High-level SystemC synthesis with forte's cynthesizer. In: High-level synthesis from algorithm to digital circuit, vol XVI, pp 75–97
7. Gupta S, Dutt N, Gupta R, Nicolau A (2003) Spark: a high-level synthesis framework for applying parallelizing compiler transformations. In: International conference on VLSI design, pp 461–466
8. Le Gal B, Casseau E (2009) Automated multimode system design for high performance DSP applications. In: Proceedings of the 17th EURASIP European conference on signal processing (EUSIPCO), Glasgow, Scotland, pp 1289–1293
9. Coussy P et al (2008) GAUT—a high-level synthesis tool for DSP applications. In: High-level synthesis from algorithm to digital circuit, vol XVI. Springer, Berlin, pp 147–169
10. Abdel-Hamid T, Tahar S, Aboulhamid EM (2004) A survey on ip watermarking techniques. Des Autom Embed Syst 9(3):211–227
11. Wolfe G, Wong JL, Potkonjak M (2002) Watermarking graph partitioning solutions. IEEE Trans Comput-Aided Des Integr Circuits Syst 21(10):1196–1204
12. Kirovski D, Potkonjak M (2003) Local watermarks: methodology and application to behavioral synthesis. IEEE Trans Comput-Aided Des Integr Circuits Syst 22(9):1277–1283
13. Intellectual Property Protection Development Working Group (2001) Intellectual property protection: schemes, alternatives and discussion (white paper). Virtual Socket Interface Alliance
14. Chapman R, Durrani T (2000) IP Protection of DSP algorithms for system on chip implementation. IEEE Trans Signal Process 48(3):854–861
15. Rashid A, Asher J, Mangione-Smith W, Potkonjak M (1999) Hierarchical watermarking for protection of dsp filter cores. In: Proceedings of the IEEE custom integrated circuits conference, pp 39–42
16. Torunoglu I, Charbon E (2000) Watermarking-based copyright protection of sequential functions. IEEE J Solid-State Circuits 35(3):434–440
17. Oliveira AL (2001) Techniques for the creation of digital watermarks in sequential circuits designs. IEEE Trans Comput-Aided Des Integr Circuits Syst 20(9):1101–1117
18. Kirovski D, Hwang YY, Potkonjak M, Cong J (1998) Intellectual property protection by watermarking combinational logic synthesis solutions. In: Proceedings of the IEEE/ACM international conference on computer-aided design (ICCAD), vol 98, pp 194–198
19. Koushanfar F, Hong I, Potkonjak M (2005) Behavioral synthesis techniques for intellectual property protection. ACM Trans Des Autom Electron Syst 10(3):523–545
20. Lach J, Mangione-Smith WH, Potkonjak M (1999) Robust FPGA intellectual property protection through multiple small watermarks. In: Proceedings of the 36th annual ACM/IEEE design automation conference (DAC'99). ACM Press, New York, pp 831–836
21. Jain A, Yuan L, Pari P, Qu G (2003) Zero overhead watermarking technique for FPGA designs. In: Proceedings of the 13th ACM Great Lakes symposium on VLSI (GLS-VLSI), pp 147–152
22. Sun G, Gao Z, Xu Y (2006) A watermarking system for ip protection by buffer insertion technique. In: Proceedings of the 7th international symposium on quality electronic design (ISQED'06). IEEE Computer Society, Washington, pp 671–675
23. Fan YC, Tsao HW (2003) Watermarking for intellectual property protection. Electron Lett 39(18):1316–1318

24. Arvind R, Nikhil S, Rosenband DL, Dave N (2004) High-level synthesis: an essential ingredient for designing complex ASICs. In: Proceedings of the IEEE/ACM international conference on computer-aided design (ICCAD'04), DC, USA, pp 775–782
25. Le Gal B, Andriamisaina C, Casseau E (2006) Bit-width aware high-level synthesis for digital signal processing systems. In: Proceeding of IEEE system-on-chip conference (SoC), Austin, Texas, pp 175–178
26. Sllame M, Drabek V (2002) An efficient list-based scheduling algorithm for high-level synthesis. In: Proceedings of the Euromicro symposium on digital systems design (DSD'02). IEEE Computer Society, Washington, p 316
27. Lim P, Kim T (2006) Thermal-aware high-level synthesis based on network flow method. In: Proceedings of the 4th international conference on Hardware/software codesign and system synthesis (CODES+ISSS'06). ACM Press, New York, pp 124–129
28. Ciesielski M, Askar S, Gomez-Prado D, Guillot J, Boutillon E (2007) Data-flow transformations using Taylor expansion diagrams. In: Proceedings of the conference on design, automation and test in Europe (DATE'07). EDA Consortium, San Jose, CA, USA, pp 455–460
29. Casseau E, Khan S, Le Gal B, Aubry W (2007) Multimode architecture design. In: Proceeding of design and architectures for signal and image processing workshop (DASIP), Grenoble, France
30. Le Gal B, Casseau E, Huet S (2008) Dynamic memory access management for high-performance DSP applications using high-level synthesis. IEEE Trans Very Large Scale Integr (VLSI) Syst 16(11):454–1464
31. Bossuet L, Gogniat G, Burleson W (2006) Dynamically configurable security for SRAM FPGA bitstreams. Int J Embed Syst, 2(1/2):73–85
32. Petitcolas F (2000) Watermarking schemes evaluation. IEEE Signal Process Mag 17(5):58–64