Forensic Detective Control using Hardware Steganography for IP Core Protection

Dr Anirban Sengupta
Professor, CSE,
IIT Indore, India

Outline

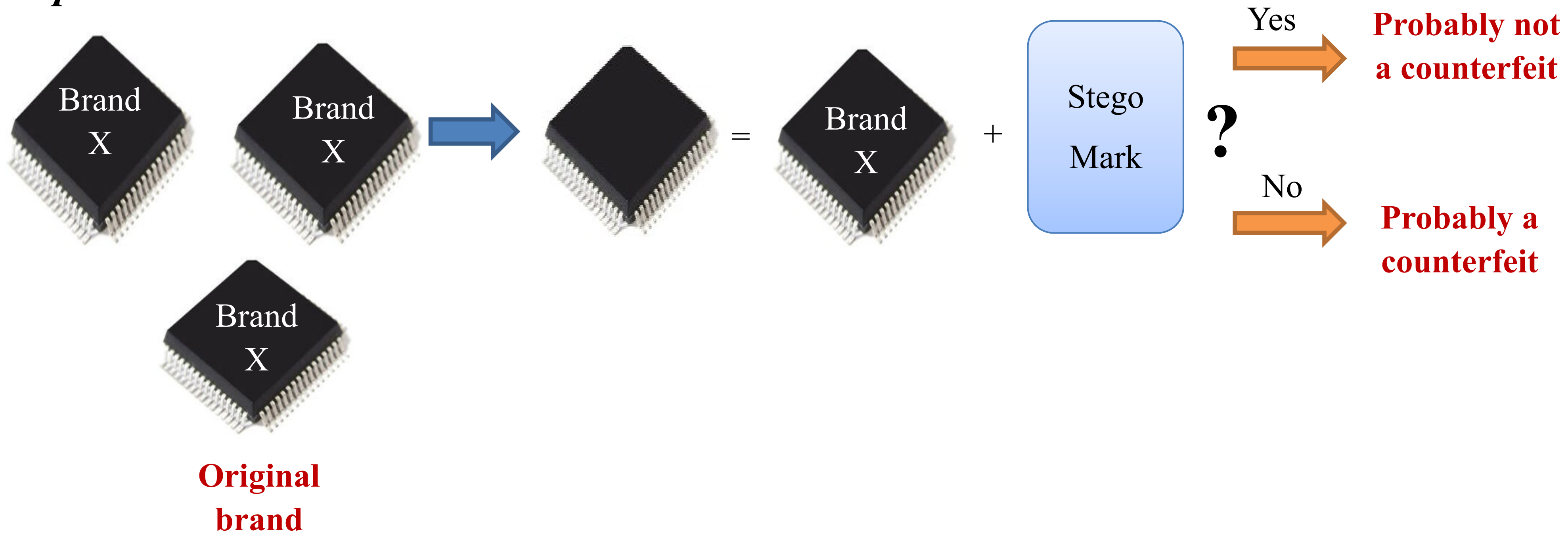
Introduction

- A number of IP cores are integrated into a system on chip (SoC) design to realize a complex electronic system to meet objectives such as reduction in design time and cost savings.
- The IP cores could be sold to SoC integrator or IC manufacturer to meet the aforesaid objectives.
- However, in the design cycle, these manufacturers or integrators may not be trustworthy and may invite some potential threats such as piracy, counterfeiting or cloning of IP core and also claim ownership.
- Therefore, forensic detective approaches can be employed to detect the abovementioned threats.
- The hardware steganography of IP core is employed to implant concealed information into the design in the form of stego-constraints.
- Thus, hardware steganography provides forensic detective control to resolve the conflict of ownership and to detect piracy, counterfeiting and cloning.

Introduction

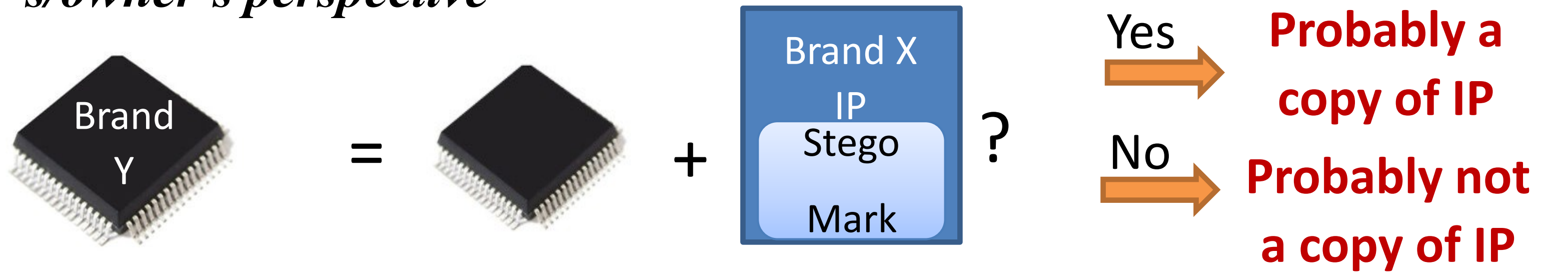
- ❑ Three different ways of using hardware steganography for forensic detective control of IP cores are depicted below:

1. ***From the end user's or the system integrator's perspective***



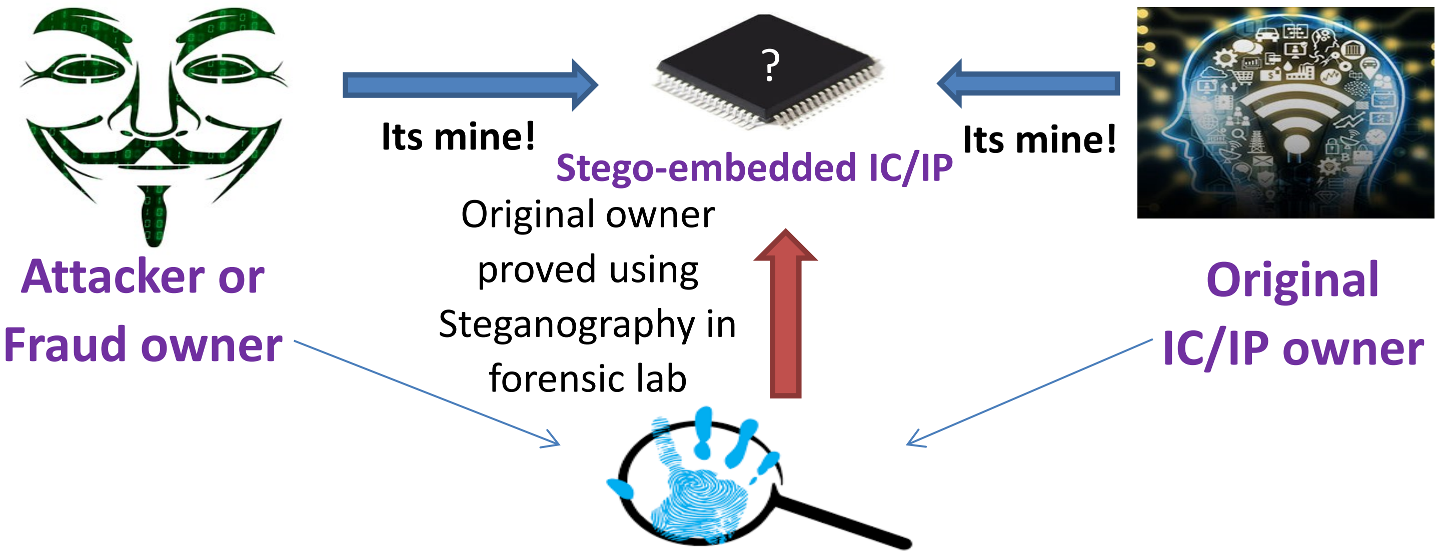
Introduction

2. From the designer's/owner's perspective



Detection of IC or IP cloning (Colombier and Bossuet, 2015)

3. IP/IC ownership conflict



Proving fraud claim of IC/IP ownership (Sengupta and Rathor, 2019a)

Fig. *Thematic representation of protecting IP cores using hardware steganography*

Threat Model

- The threat model recognizes the potential vulnerabilities and most relevant attacks to an IP core design from an attacker’s point of view.

❑ Potential threats against IP core designs are:

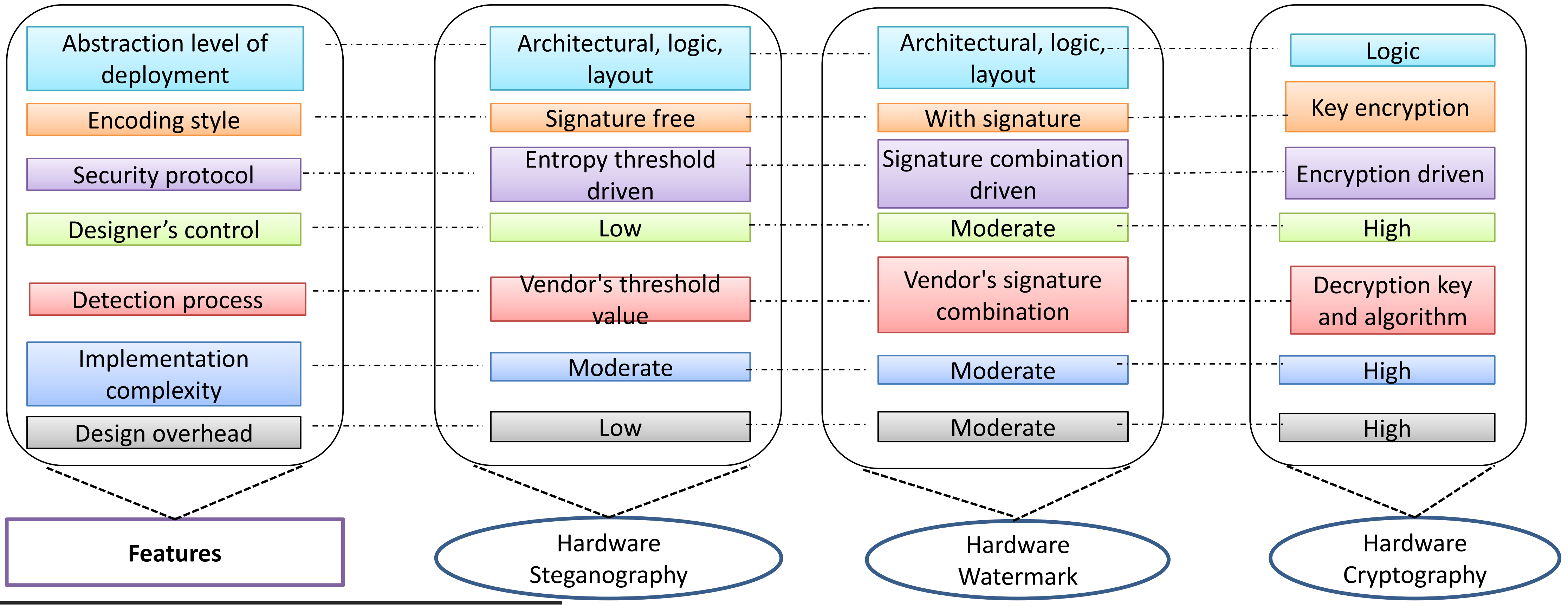
(i) IP/IC piracy

(ii) IP/IC counterfeiting and cloning

(iii) Fraudulent claim of ownership

Selected Contemporary Approaches

- Hardware watermarking and cryptography are some of the contemporary approaches of more recent hardware steganography.



Features	Hardware Steganography	Hardware Watermark	Hardware Cryptography
Abstraction level of deployment	Architectural, logic, layout	Architectural, logic, layout	Logic
Encoding style	Signature free	With signature	Key encryption
Security protocol	Entropy threshold driven	Signature combination driven	Encryption driven
Designer's control	Low	Moderate	High
Detection process	Vendor's threshold value	Vendor's signature combination	Decryption key and algorithm
Implementation complexity	Moderate	Moderate	High
Design overhead	Low	Moderate	High

Fig. *Comparison of hardware steganography with its contemporary approaches (Sengupta and Rathor, 2019c).*

Limitation of Hardware Watermarking

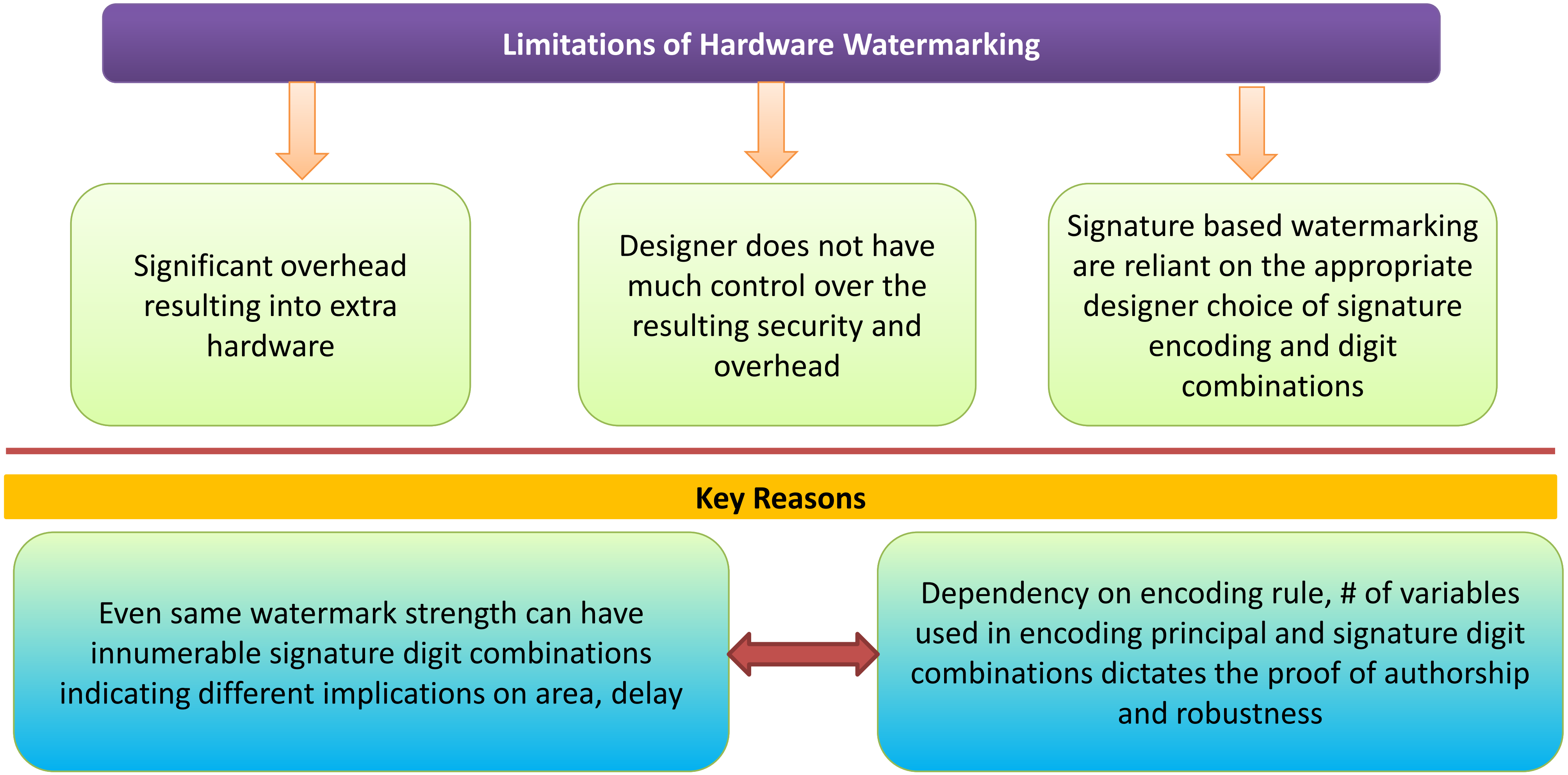
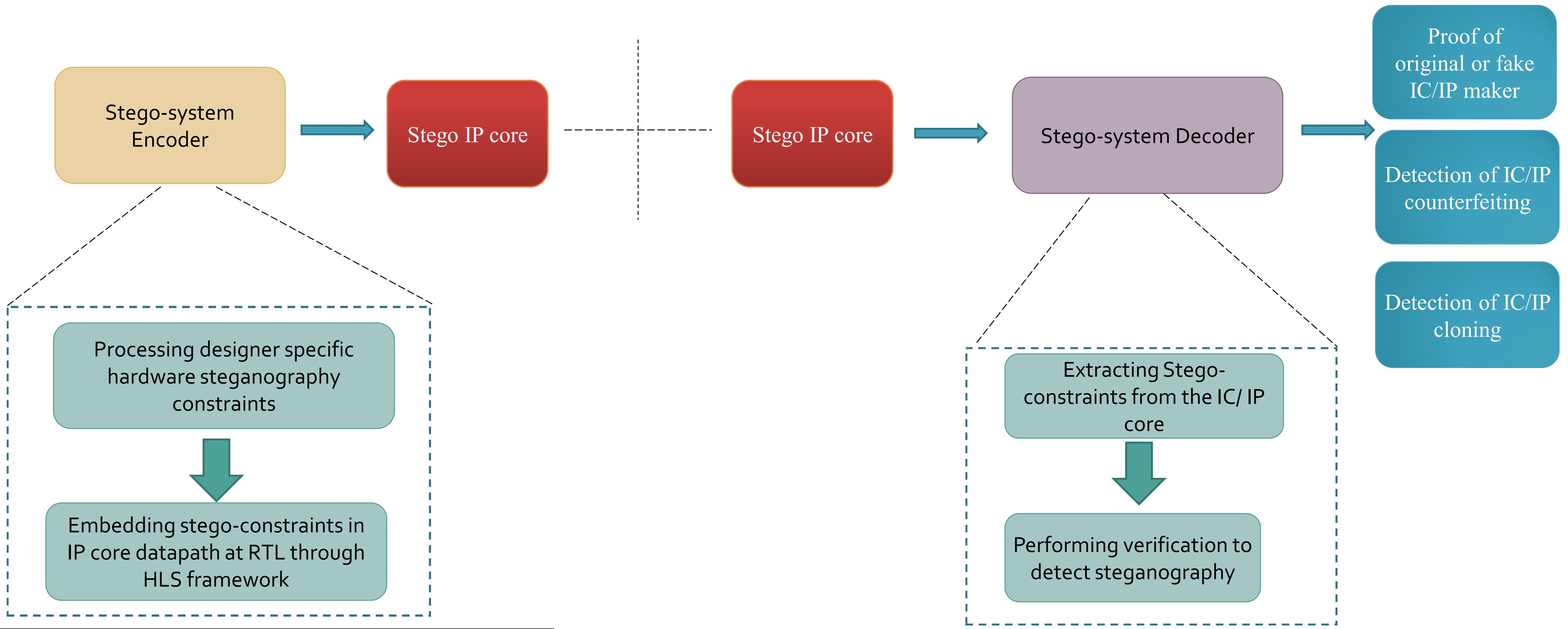

Limitations of Hardware Watermarking
Significant overhead resulting into extra hardware
Designer does not have much control over the resulting security and overhead
Signature based watermarking are reliant on the appropriate designer choice of signature encoding and digit combinations
Key Reasons
Even same watermark strength can have innumerable signature digit combinations indicating different implications on area, delay
Dependency on encoding rule, # of variables used in encoding principal and signature digit combinations dictates the proof of authorship and robustness

Fig. *Limitations of Hardware Watermarking (Sengupta and Rathor, 2019a).*

IP Core Steganography Model

❑ The IP core steganography model can be partitioned into following two processes as shown below:
***1) Hardware steganography encoding process*: *2) Hardware steganography decoding process*:**

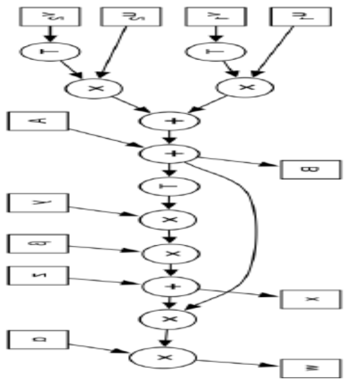


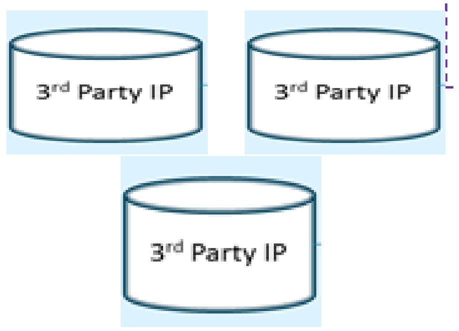
Stego-system Encoder
Stego IP core
Stego IP core
Stego-system Decoder
Proof of original or fake IC/IP maker
Detection of IC/IP counterfeiting
Detection of IC/IP cloning
Processing designer specific hardware steganography constraints
Embedding stego-constraints in IP core datapath at RTL through HLS framework
Extracting Stego-constraints from the IC/ IP core
Performing verification to detect steganography

A. Sengupta and M. Rathor, *IEEE Transactions on Consumer Electronics*, 2019.

Forensic Detective Control using Hardware (IP Core) Steganography

Input: DFG of the DSP design





Input Block

Data Flow Graph (DFG) of DSP

Resource Configuration

High Level Synthesis Framework

Storage assignment phase in Scheduling (Module 1)

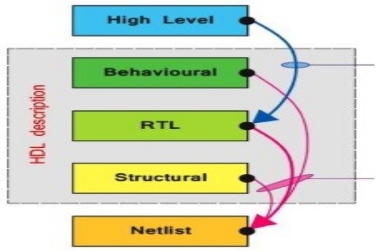
Edge sets determination phase in CIG (Module 2)

Swapping pair determination phase (Module 3)

Maximum Entropy determination phase (Module 4)

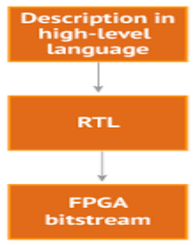
Shortlisted Edge Sets determination phase (Module 5)

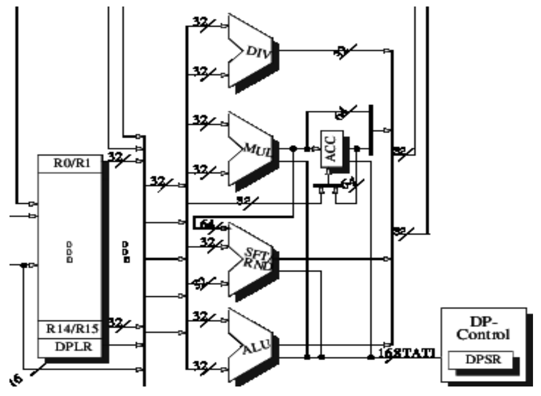
Stego-constraints embedding phase in IP core at RTL (Module 6)



Stego DSP IP Core

Output Block





Output: steganography embedded IP core (Stego IP core).

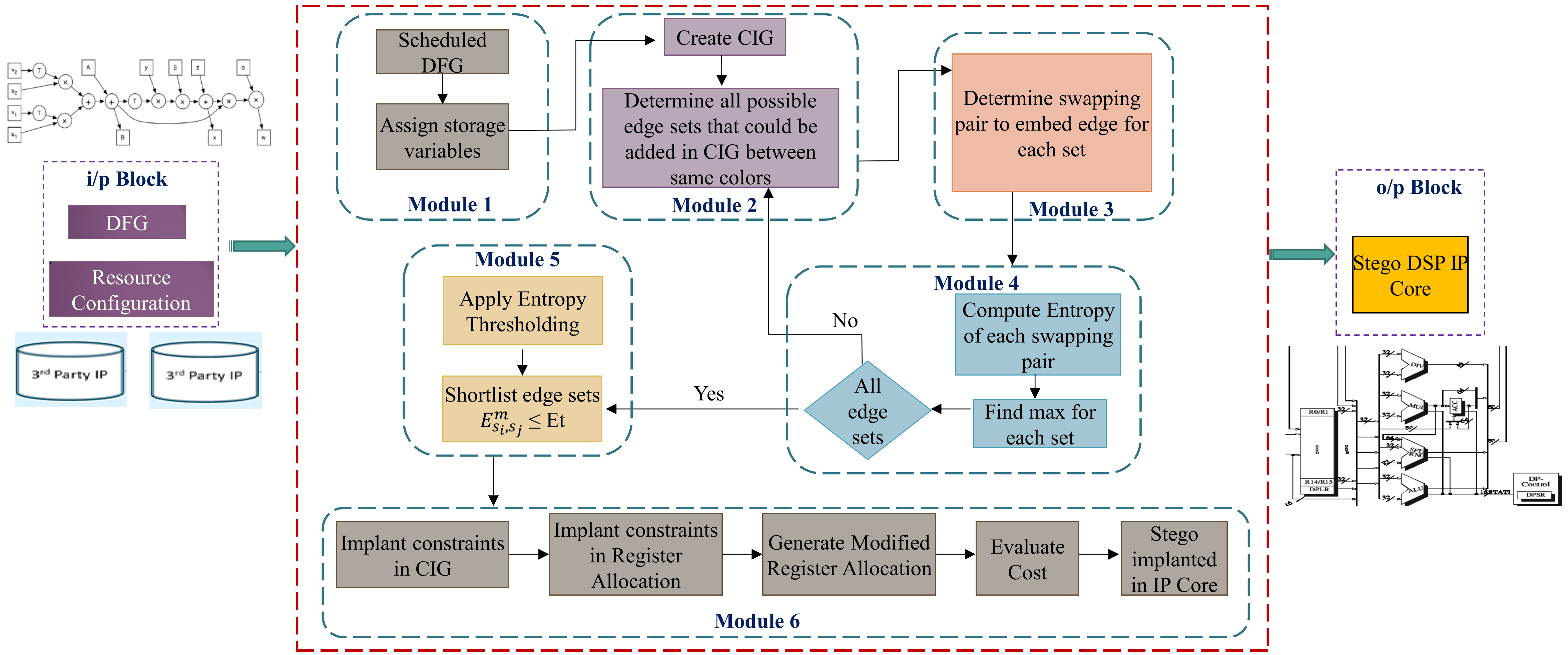
Fig. *Overview of the implanting Hardware Steganography in IP cores (Sengupta and Rathor, 2019b)*

Problem Definition

- Using vendor-specified resource configuration Ni(Rj), a hardware steganography- based IP core design is generated.
- A data flow graph (DFG) of the IP core design is fed as a primary input along with the resource configuration.
- The resource configuration Ni(Rj) represents the number of resources of each resource type Rj.

Cont.

- ❑ The details of these six modules during the process of embedding steganography.



i/p Block
DFG
Resource Configuration
3rd Party IP
3rd Party IP
Scheduled DFG
Assign storage variables
Module 1
Create CIG
Determine all possible edge sets that could be added in CIG between same colors
Module 2
Determine swapping pair to embed edge for each set
Module 3
Module 4
Compute Entropy of each swapping pair
Find max for each set
All edge sets
No
Yes
Module 5
Apply Entropy Thresholding
Shortlist edge sets $E^m_{s_i,s_j} \leq$ Et
Implant constraints in CIG
Implant constraints in Register Allocation
Generate Modified Register Allocation
Evaluate Cost
Stego implanted in IP Core
Module 6
o/p Block
Stego DSP IP Core

Fig. *Embedding process of Hardware Steganography for an IP Core (Sengupta and Rathor, 2019b)*

Detection (extraction) of Steganography

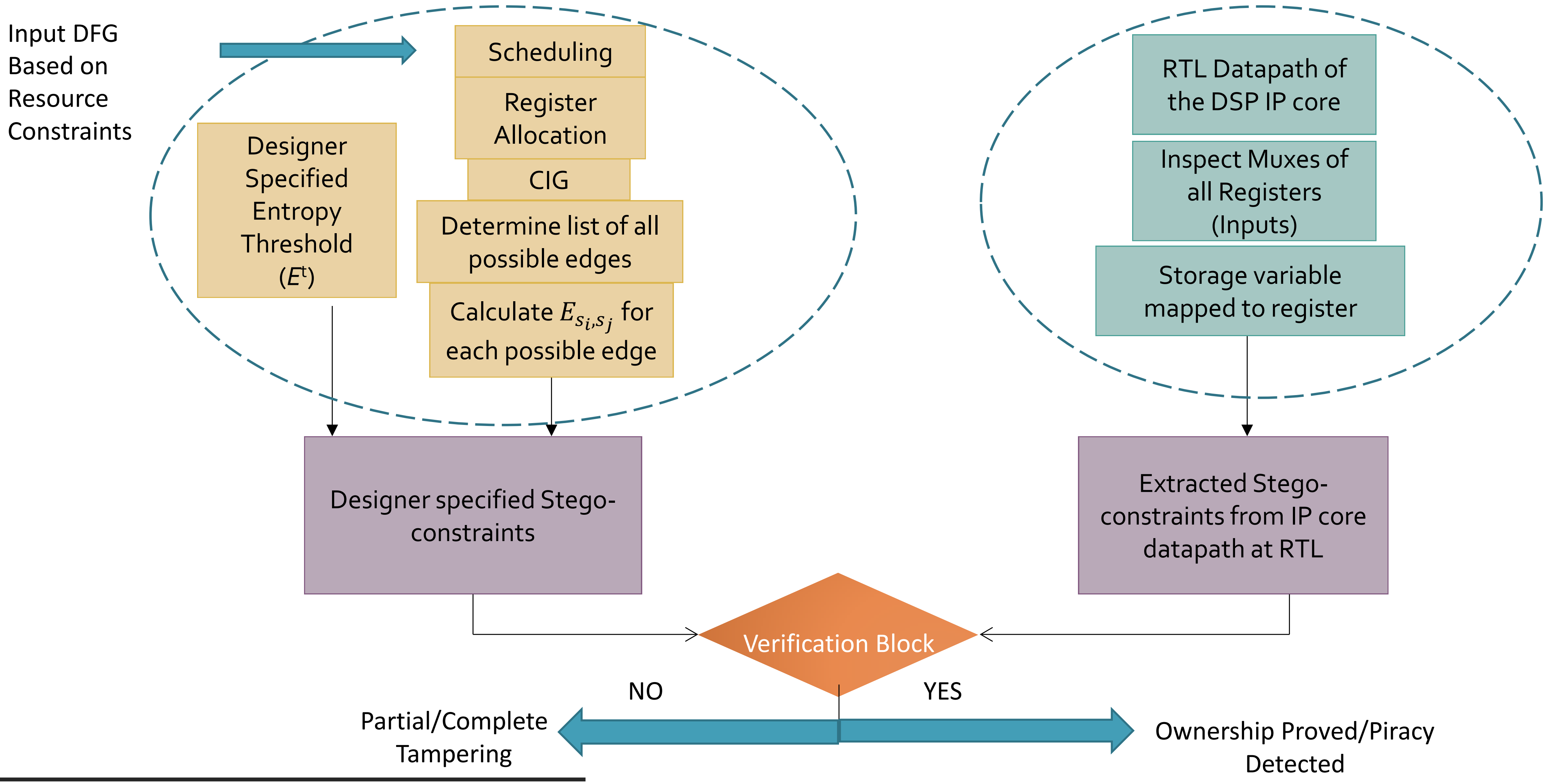

Input DFG Based on Resource Constraints
Scheduling
Register Allocation
CIG
Determine list of all possible edges
Calculate E_{s_i,s_j} for each possible edge
Designer Specified Entropy Threshold (E^t)
Designer specified Stego-constraints
RTL Datapath of the DSP IP core
Inspect Muxes of all Registers (Inputs)
Storage variable mapped to register
Extracted Stego-constraints from IP core datapath at RTL
Verification Block
NO
YES
Partial/Complete Tampering
Ownership Proved/Piracy Detected

Fig. *Hardware Steganography Detection Process in IP cores (Sengupta and Rathor, 2019a)*

Demonstration Example

- **The process of implanting hardware steganography is demonstrated with a sample application**.
- The CIG of a sample application and corresponding register allocation table is shown below:

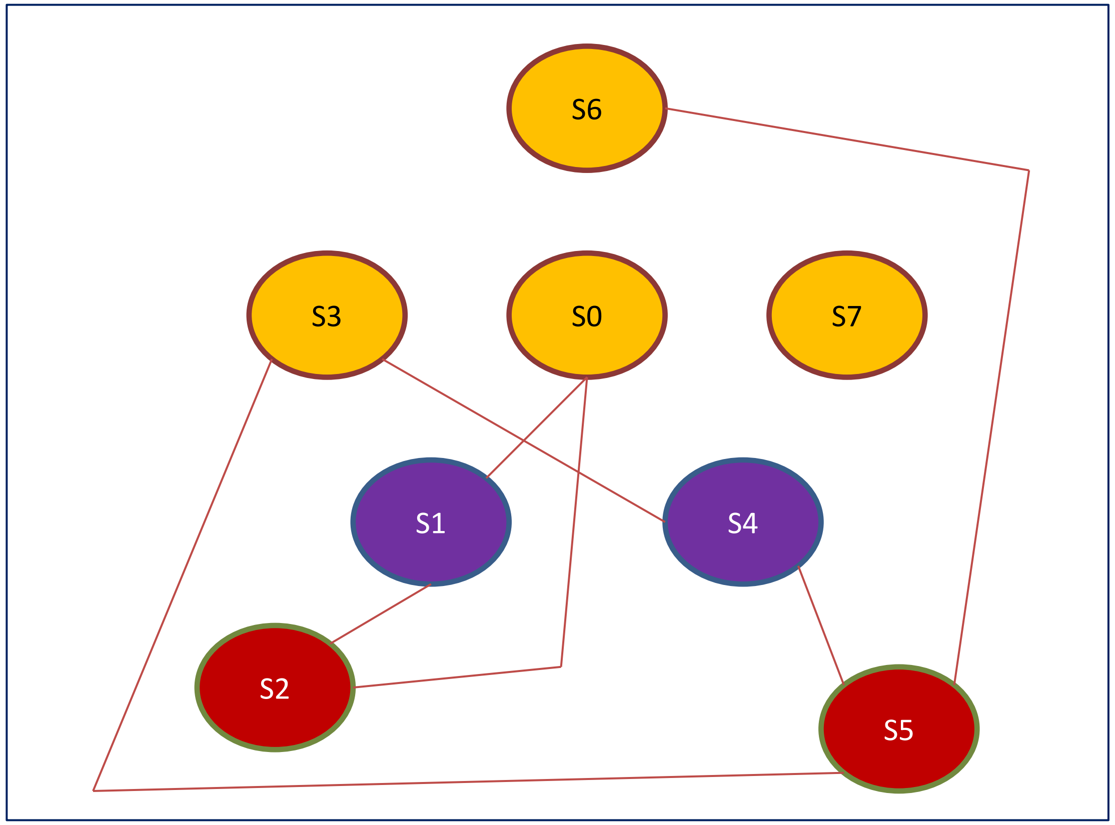


Fig. *CIG of a sample application*

TABLE 2.1 Register allocation of sample application

Clock cycle	**Yellow (Y)**	**Violet (V)**	**Red (R)**
T0	S0	S1	S2
T1	S3	S4	S5
T2	S6	--	S5
T3	S7	--	--

Cont.

- An edge set 'S' containing all possible (potential) edges between nodes of same colours can be determined by observing the CIG.

S= {(S0, S3), (S0, S6), (S0, S7), (S1, S4), (S2, S5), (S3, S6), (S3, S7), (S6, S7)}

❑ Embedding an edge between node pair $(S0, S3)$, (S0, S6) is possible through following solutions presented

Swapping pairs and corresponding entropies for the edge (S0, S3)

Edge	Control step	Swapping pair	Swapping colors (Registers)	Hardware Entropy (E_{s_i,s_j})
(S0, S3)	T0	(S0 ⇔ S1)	(Y ⇔ V)	2
	T0	(S0 ⇔ S2)	(Y ⇔ R)	2
	T1	(S3 ⇔ S4)	(Y ⇔ V)	2
	T1	(S3 ⇔ S5)	(Y ⇔ R) + (Y ⇔ R)	3
	T2	(S6 ⇔ S5)		

Swapping pairs and corresponding entropies for the edge (S0, S6)

Edge	Control step	Swapping pair	Swapping colors (Registers)	Hardware Entropy (E_{s_i,s_j})
(S0, S6)	T0	(S0 ⇔ S1)	(Y ⇔ V)	2
	T0	(S0 ⇔ S2)	(Y ⇔ R)	2
	T2	(S6 ⇒ --)	(Y ⇒ V)	1
	T2	(S6 ⇔ S5) + (S3 ⇔ S5)	(Y ⇔ R)	3

Cont.

- Embedding an edge between node pair $(S0, S7)$, (S1, S4) is possible through following solutions presented

Swapping pairs and corresponding entropies for the edge (S0, S7)

Edge	Control step	Swapping pair	Swapping colors (Registers)	Hardware Entropy (E_{s_i,s_j})
(S0, S7)	T0	(S0 ⇔ S1)	(Y ⇔ V)	2
	T0	(S0 ⇔ S2)	(Y ⇔ R)	2
	T3	(S7 ⇒ --)	(Y ⇒ V)	1
	T3	(S7 ⇒ --)	(Y ⇒ R)	1

Swapping pairs and corresponding entropies for the edge (S1, S4)

Edge	Control step	Swapping pair	Swapping colors (Registers)	Hardware Entropy (E_{s_i,s_j})
(S1, S4)	T0	(S1 ⇔ S0)	(V ⇔ Y)	2
	T0	(S1 ⇔ S2)	(V ⇔ R)	2
	T1	(S4 ⇔ S3)	(V ⇔ Y)	2
	T1 T2	(S4 ⇔ S5) + (S5 ⇒ --)	(V ⇔ R) + (R ⇒ V)	2

Cont.

- Embedding an edge between node pair $(S2, S5)$, (S3, S6) is possible through following solutions presented

Swapping pairs and corresponding entropies for the edge (S2, S5)

Edge	Control step	Swapping pair	Swapping colors (Registers)	Hardware Entropy (E_{s_i,s_j})
(S2, S5)	T0	(S2 ⇔ S0)	(Y ⇔ R)	2
	T0	(S2 ⇔ S1)	(R ⇔ V)	2
	T1	(S5 ⇔ S4)	(R ⇔ V)	2
	T1 T2	(S5 ⇔ S3) + (S5 ⇔ S6)	(R ⇔ Y) + (R ⇔ Y)	3

Swapping pairs and corresponding entropies for the edge (S3, S6)

Edge	CS	Swapping pair	Swapping colors (Registers)	Hardware Entropy (E_{s_i,s_j})
(S3, S6)	T1	(S3 ⇔ S4)	(Y ⇔ V)	2
	T2	(S6 ⇒ --)	(Y ⇒ R)	1

Cont.

- Embedding an edge between node pair $(S3, S7)$, (S6, S7) is possible through following solutions presented

Swapping pairs and corresponding entropies for the edge (S3, S7)

Edge	Control step	Swapping pair	Swapping colors (Registers)	Hardware Entropy (E_{s_i,s_j})
(S3, S7)	T1	(S3 ⇔ S4)	(Y ⇔ V)	2
	T1	(S3 ⇔ S5) + (S5 ⇔ S6)	(Y ⇔ R)	3
	T3	(S7 ⇒ --)	(Y ⇒ V)	1
	T3	(S7 ⇒ --)	(Y ⇒ R)	1

Swapping pairs and corresponding entropies for the edge (S6, S7)

Edge	Control step	Swapping pair	Swapping colors (Registers)	Hardware Entropy (E_{s_i,s_j})
(S6, S7)	T2	(S6 ⇒ --)	(Y⇒ V)	1
	T2	(S6 ⇔ S5) + (S5 ⇔ S3)	(Y ⇔ R)	3
	T3	(S7 ⇒ --)	(Y ⇒ V)	1
	T3	(S7 ⇒ --)	(Y ⇒ R)	1

Cont.

- These tables list all possible swapping pairs and their corresponding entropies for each edge between node pair (Si, Sj).
- For the designer (vendor) specified entropy thresholding value E^t =2, the edges which qualify for the insertion into the CIG (of the IP core) are represented using a set S*, where S*⊂ S.
- The generated S*constitutes as: S*= {(S0, S7), (S1, S4), (S3, S6)}, only the edges in the set S*are eligible to be embedded into the CIG because maximum entropy $(E^{m}_{s_i,s_j})$ of each of these edges is less than or equal to 2.

Cont.

- The modified register allocation after embedding first edge between node pair (S0, S7), (S1, S4), (S3,S6) is shown below:

Register allocation after implanting (S0, S7)			
Clock cycle	**Yellow**	**Violet**	**Red**
T0	S0	S1	S2
T1	S3	S4	S5
T2	S6	---	S5
T3	---	---	**S7**

Register allocation after implanting (S1, S4)			
Clock cycle	**Yellow**	**Violet**	**Red**
T0	S0	S1	S2
T1	S3	**S5**	**S4**
T2	S6	**S5**	---
T3	---	---	S7

Register allocation after implanting (S3, S6)			
Clock cycle	**Yellow**	**Violet**	**Red**
T0	S0	S1	S2
T1	S3	S5	S4
T2	---	S5	**S6**
T3	---	---	S7

Cont.

- Embedding these edges in the form of concealed Sego information results into modified CIG of the IP core is shown below:

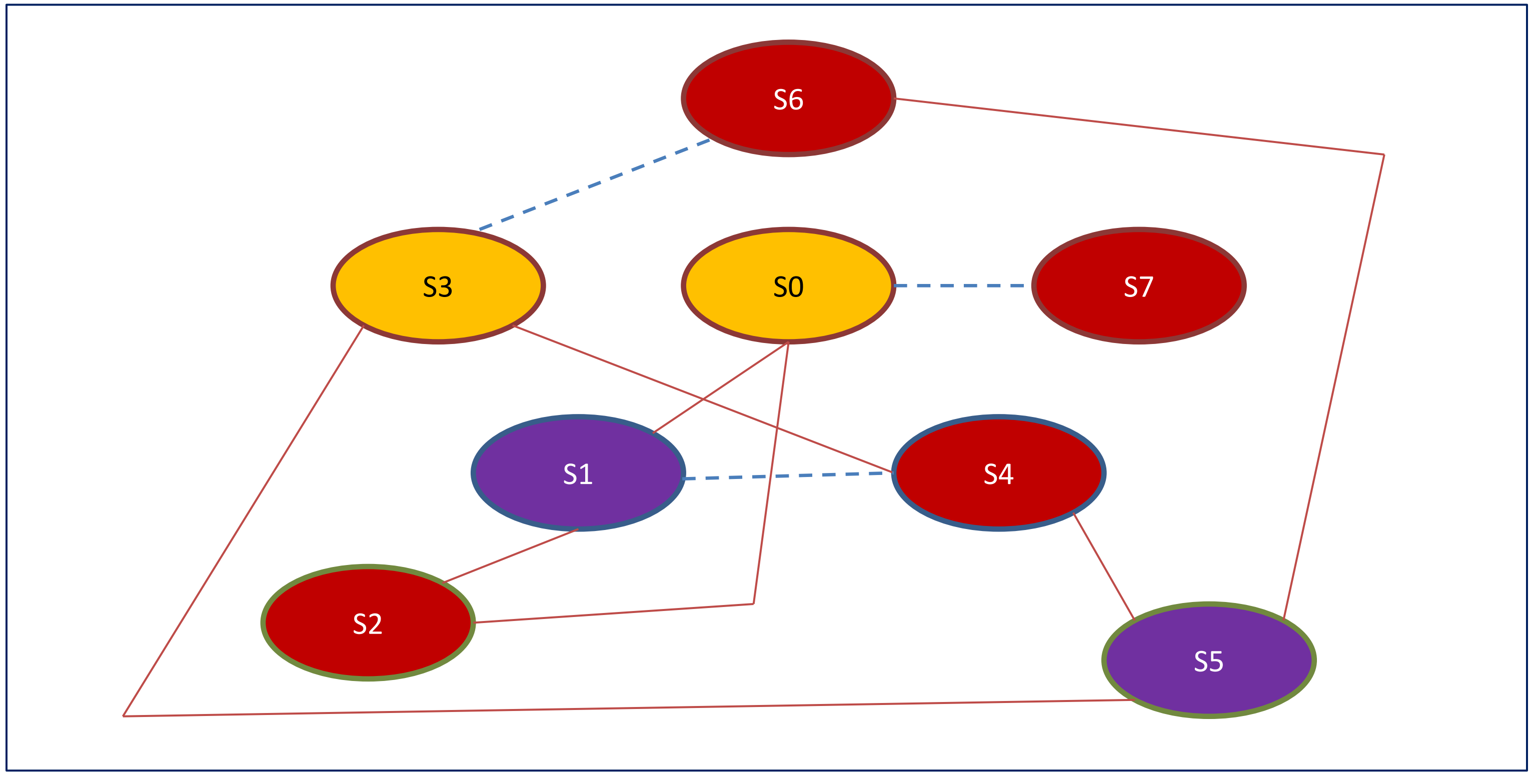


Fig. *Final CIG after implanting all the edges of hardware steganography*

Design Process of Steganography Embedded IP Core: A case study on 8-point DCT

- Highly complex designs such as DSP kernels are easier to describe at higher abstraction level such as architecture level or ESL.
- Such complex designs are transformed from their architecture description to RTL description using HLS framework.
- Therefore during HLS, a protection mechanism can easily be deployed in such complex designs to secure them against various threats.
- Therefore, more complex designs such as DSP kernels which are initially described at architecture level can be protected using hardware steganography during their high level synthesis.

❑ The design process is accomplished in following two phases:

(i) ***pre-processing phase before steganography*** (ii) ***Stego-constraints implantation phase***.

(i)Pre-processing phase before steganography

- ❑ This phase translates the 8-point DCT design into a suitable form to employ the steganography approach.

- ➢ The DFG representing 8-point DCT design along with vendor specified resource configuration is fed to the HLS framework.

- ➢ Based on the resource configuration; four multipliers (M1, M2, M3, and M4) and one adder (A1), the DFG is scheduled as shown here:

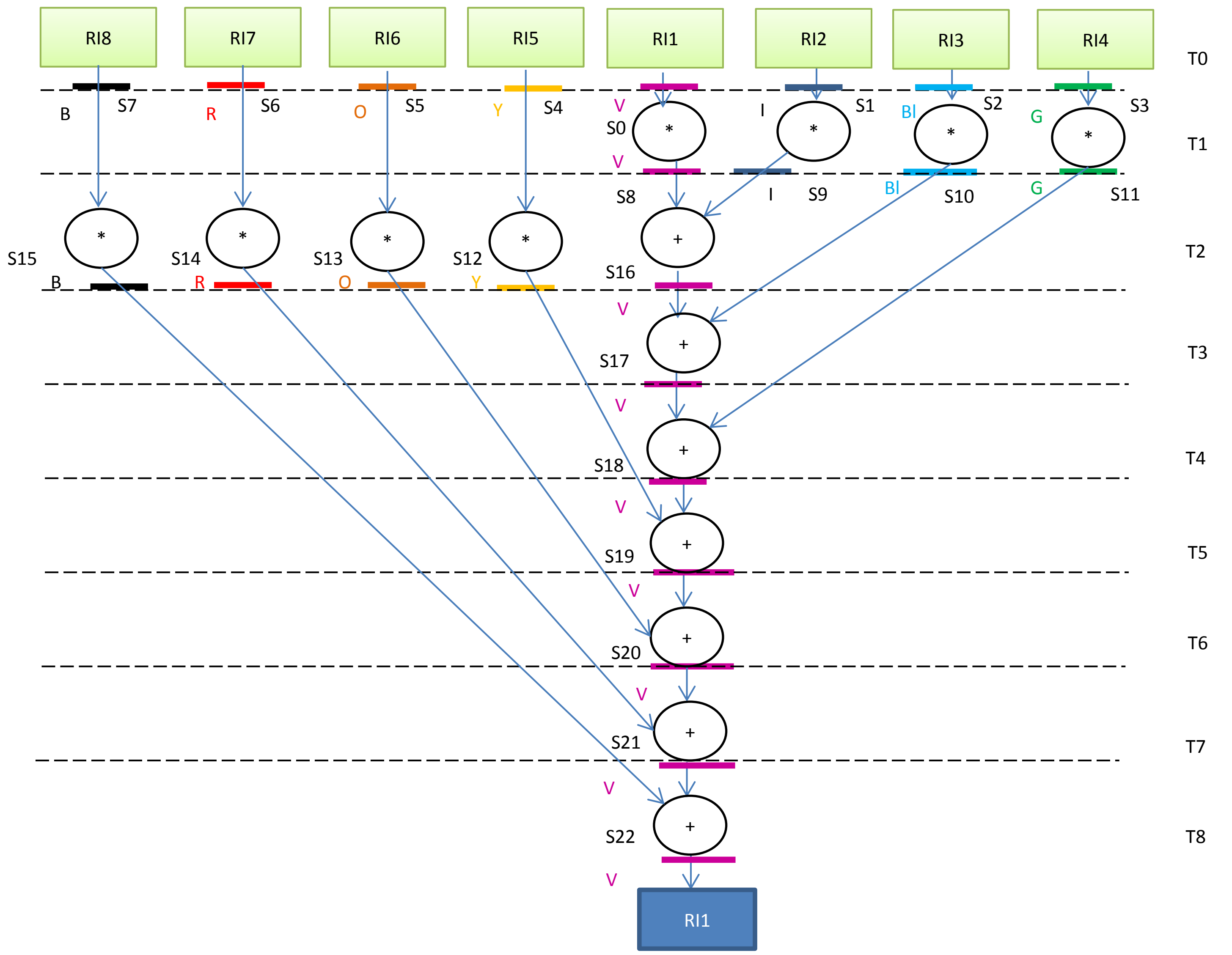


Fig. *Scheduled and hardware allocated 8-point DCT using 1 (+) and 4 (*) BEFORE implanting steganography*

Cont.

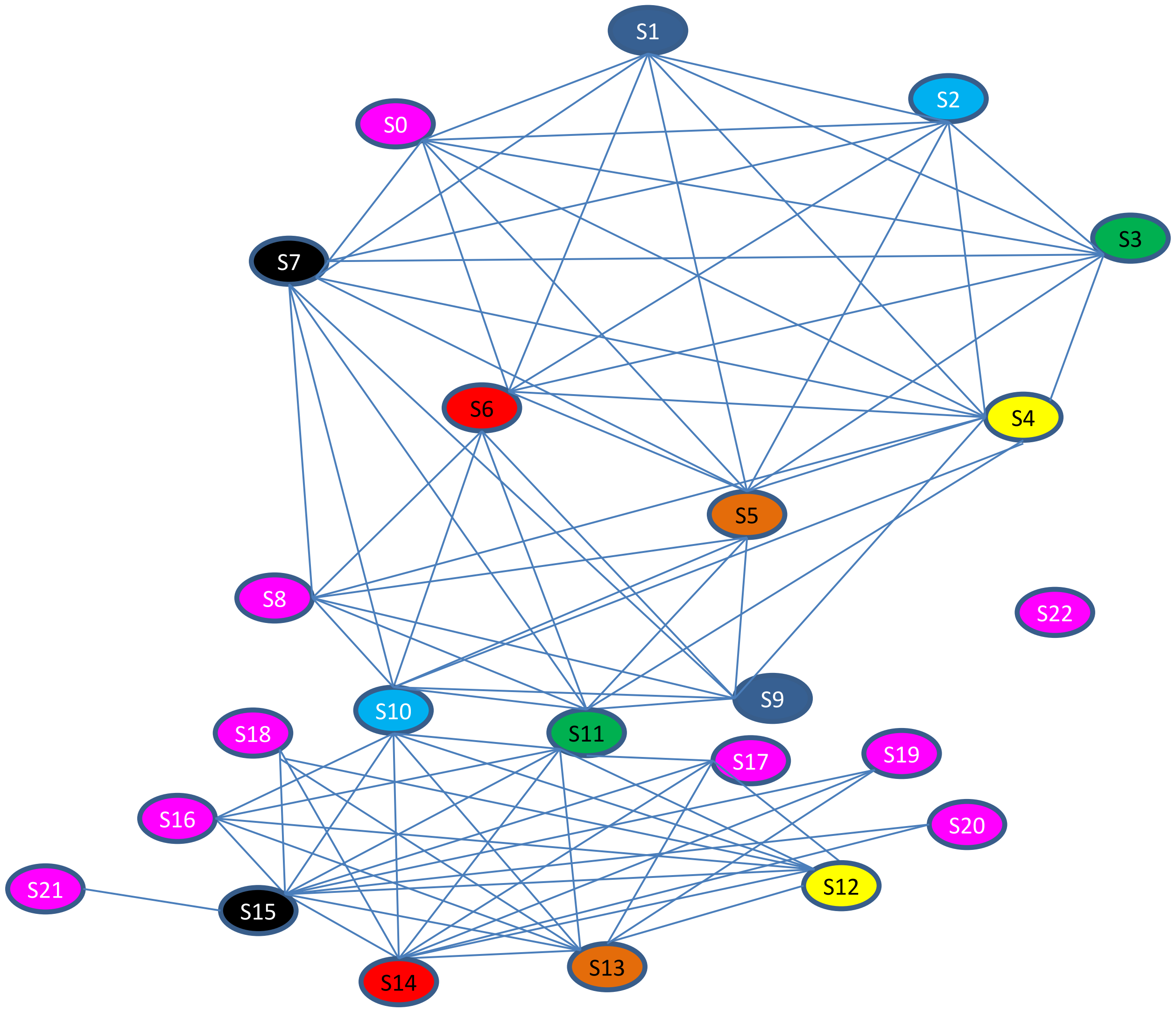


Fig. *A CIG of 8-point DCT before steganography*

The allocation of all storage variables to their corresponding registers

CS	Violet	Indigo	Blue	Green	Yellow	Orange	Red	Black
0	S0	S1	S2	S3	S4	S5	S6	S7
1	S8	S9	S10	S11	S4	S5	S6	S7
2	S16	--	S10	S11	S12	S13	S14	S15
3	S17	--	--	S11	S12	S13	S14	S15
4	S18	--	--	--	S12	S13	S14	S15
5	S19	--	--	--	--	S13	S14	S15
6	S20	--	--	--	--	--	S14	S15
7	S21	--	--	--	--	--	--	S15
8	S22	--	--	--	--	--	--	--

Cont.

The RTL datapath of the 8-point DCT design before implanting the steganography.

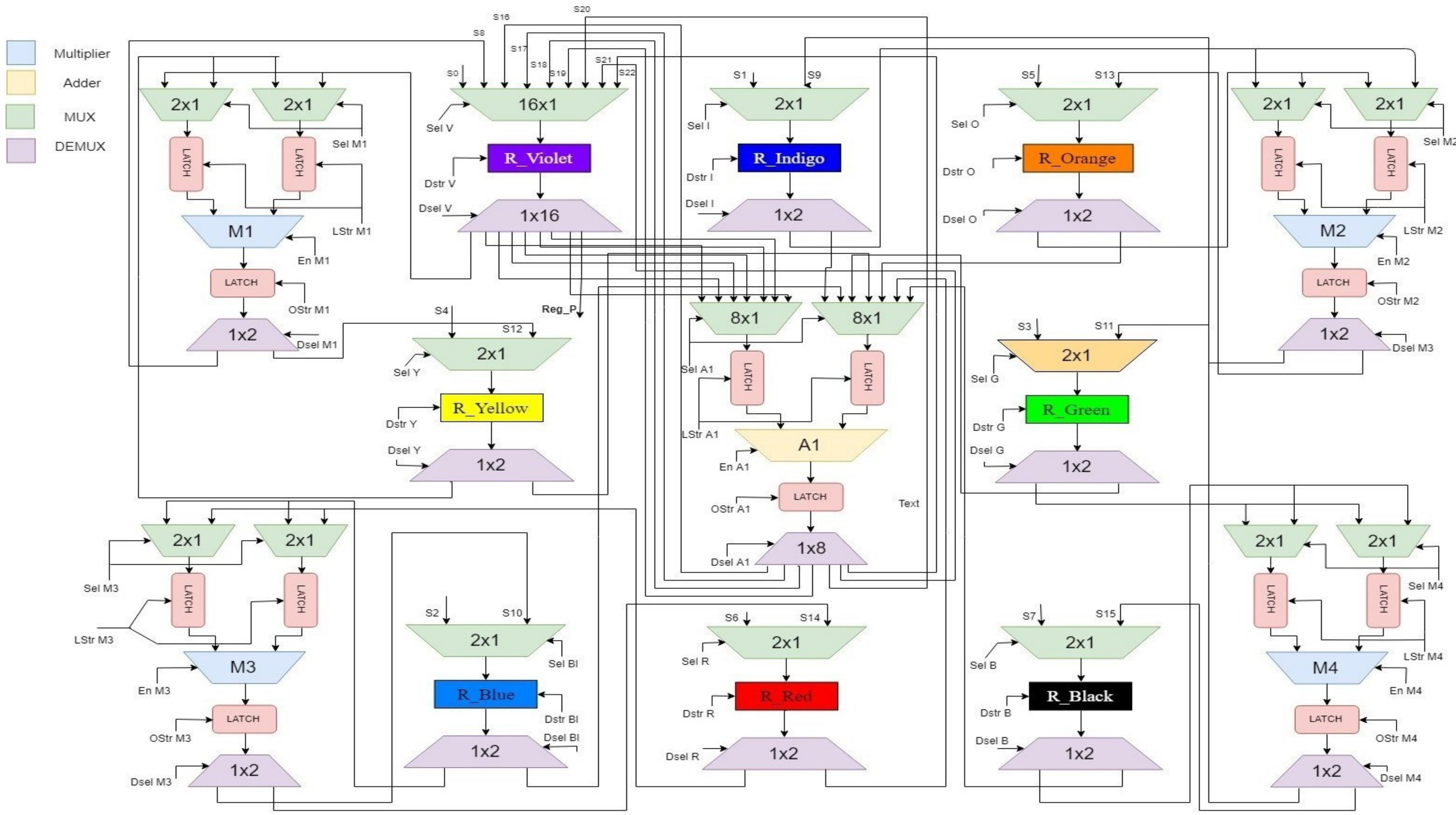


Fig. *RTL datapath of 8-point DCT before steganography*

(ii)Stego-constraints implantation phase

- Steps of implanting stego-constraints to perform hardware steganography are as follows:

Step-1: The CIG is inspected to obtain all the possible edges between nodes of same color. Hence the set. ‘S’ containing all possible edges comprises of:

S= {(S0, S8), (S0, S16), (S0, S17), (S0, S18), (S0, S19), (S0, S20), (S0, S21), (S0, S22), (S1, S9), (S2, S10), (S3, S11), (S4, S12), (S5, S13), (S6, S14), (S7, S15), (S8, S16), (S8, S17), (S8, S18), (S8, S19), (S8, S20), (S8, S21), (S8, S22), (S16, S17), (S16, S18), (S16, S19), (S16, S20), (S16, S21), (S16, S22), (S17, S18), (S17, S19), (S17, S20), (S17, S21), (S17, S22), (S18, S19), (S18, S20), (S18, S21), (S18, S22), (S19, S20), (S19, S21), (S19, S22), (S20, S21), (S20, S22), (S21, S22)}

Cont.

Step-2: All possible swapping pairs corresponding to each edge in the edge set S are determined to resolve the conflict of embedding that edge.

Step-3: Entropy corresponding to each swapping pair of each edge is determined.

Step-4: The maximum entropy value corresponding to each edge in the set is determined and has been listed in Table.

Note*: the pairs in bold indicate entropy less than threshold entropy.*

Sorted list of all possible edges in the set S and their respective maximum entropies

Possible Edge	Maximum Entropy to insert an Edge	Possible Edge	Maximum Entropy to insert an Edge
(S0, S8)	**4**	**(S8, S22)**	**4**
(S0, S16)	7	**(S16, S17)**	**3**
(S0, S17)	7	**(S16, S18)**	**4**
(S0, S18)	7	**(S16, S19)**	**4**
(S0, S19)	7	**(S16, S20)**	**5**
(S0, S20)	7	(S16, S21)	6
(S0, S21)	7	(S16, S22)	7
(S0, S22)	**3**	**(S17, S18)**	**4**
(S1, S9)	**2**	**(S17, S19)**	**4**
(S2, S10)	**3**	**(S17, S20)**	**5**
(S3, S11)	**4**	(S17, S21)	6
(S4, S12)	**4**	(S17, S22)	7
(S5, S13)	**5**	**(S18, S19)**	**4**
(S6, S14)	6	**(S18, S20)**	**5**
(S7, S15)	7	(S18, S21)	6
(S8, S16)	7	(S18, S22)	7
(S8, S17)	7	**(S19, S20)**	**5**
(S8, S18)	7	(S19, S21)	6
(S8, S19)	7	(S19, S22)	7

(ii)Stego-constraints implantation phase

- ***Step-5***: An appropriate value of the entropy threshold is chosen by the IP vendor or designer.
- ***Step-6***: The set of shortlisted edges S* to be embedded into the CIG as stego-constraints is obtained based on an entropy threshold value (E^t). For Et =5, only the following edges in the set S* qualify for the insertion into the CIG:

S*= {(S0, S8), (S0, S22), (S1, S9), (S2, S10), (S3, S11), (S4, S12), (S5, S13), (S8, S22), (S16, S17), (S16, S18), (S16, S19), (S16, S20), (S17, S18), (S17, S19), (S17, S20), (S18, S19), (S18, S20), (S19, S20)}

- ***Step-7***: All eligible edges in the set S* are embedded into the CIG as stego-constraints and design cost is evaluated. If the design cost and robustness (# of Stego-constraints) is not obtained as per expected value then go to step 5 and choose another value of E^t.
- ***Step-8***: For a designer selected E^t , secret information in the form of artificial constraints edges is concealed into the IP core design while achieving expected design cost and security and hence the process of implanting steganography into the 8-point DCT design is accomplished.

Cont.

- After implanting stego-constraints for Et equal to 5, the effect of steganography is reflected (at ESL) in the form of modified CIG and register allocation shown.

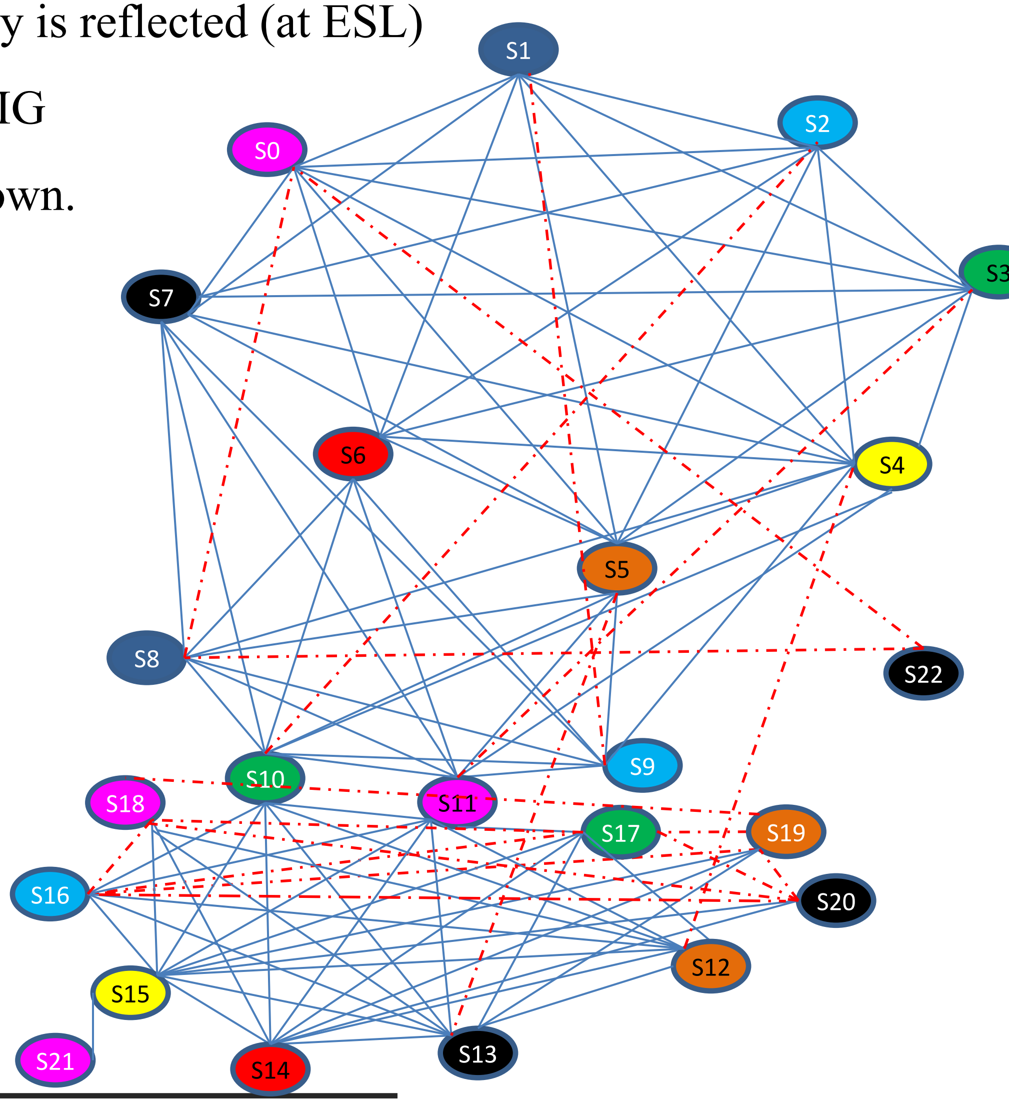


Fig. *A CIG of 8-point DCT after steganography*

Register allocation of 8-point DCT **after** implanting steganography

CS	Violet	Indigo	Blue	Green	Yellow	Orang e	Red	Black
0	S0	S1	S2	S3	S4	S5	S6	S7
1	S11	S8	S9	S10	S4	S5	S6	S7
2	S11	--	S16	S10	S15	S12	S14	S13
3	S11	--	--	S17	S15	S12	S14	S13
4	S18	--	--	--	S15	S12	S14	S13
5	--	--	--	--	S15	S19	S14	S13
6	--	--	--	--	S15	--	S14	S20
7	S21	--	--	--	S15	--	--	--
8	--	--	--	--	--	--	--	S22

Cont.

- The modified scheduling of the 8-point DCT post steganography implant.

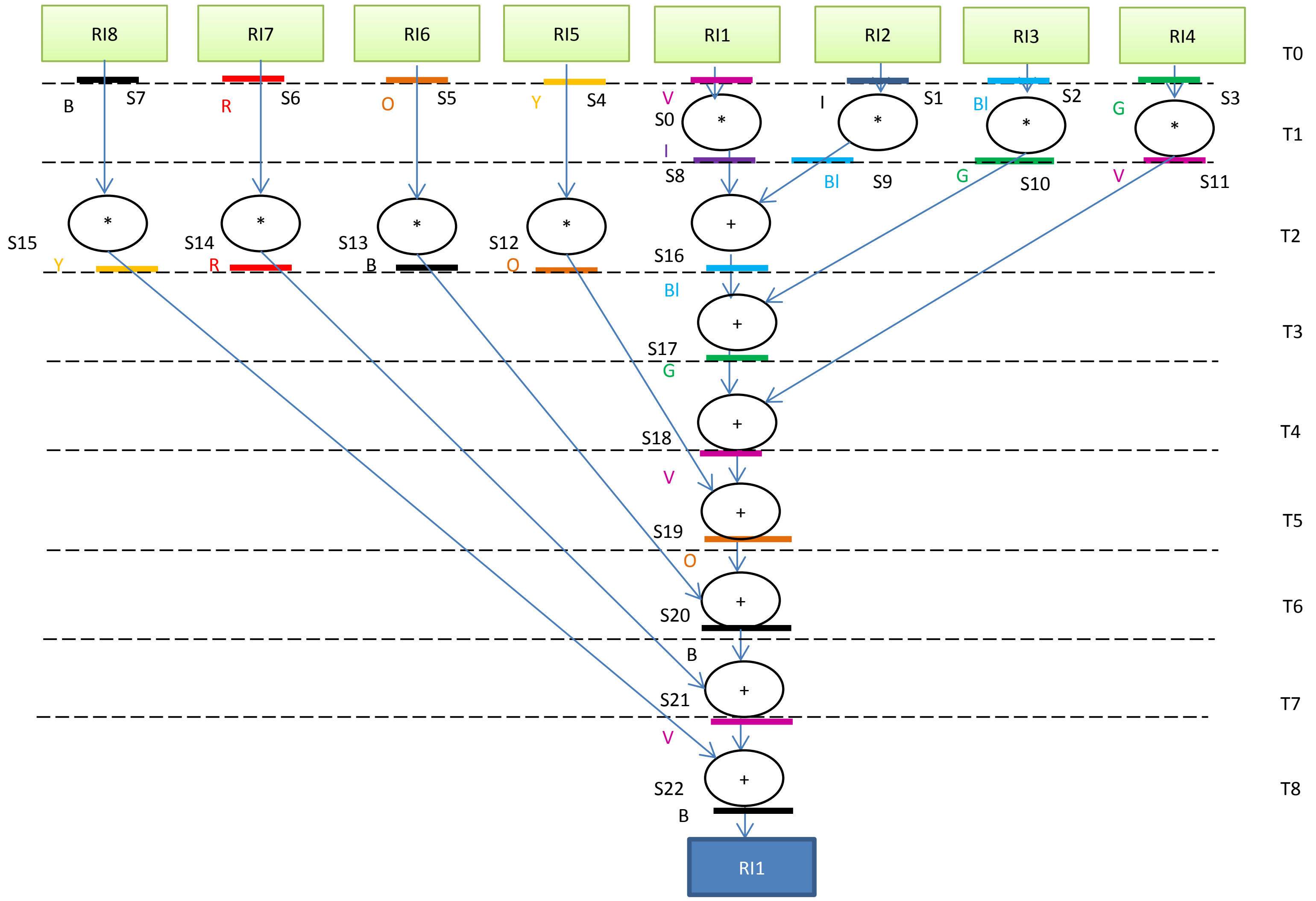


Fig. *Scheduled and hardware allocated 8-point DCT using 1 (+) and 4 (*) AFTER implanting steganography*

Pre-processing phase before steganography

- The effect of hardware steganography at RTL datapath of the IP core design is shown here.

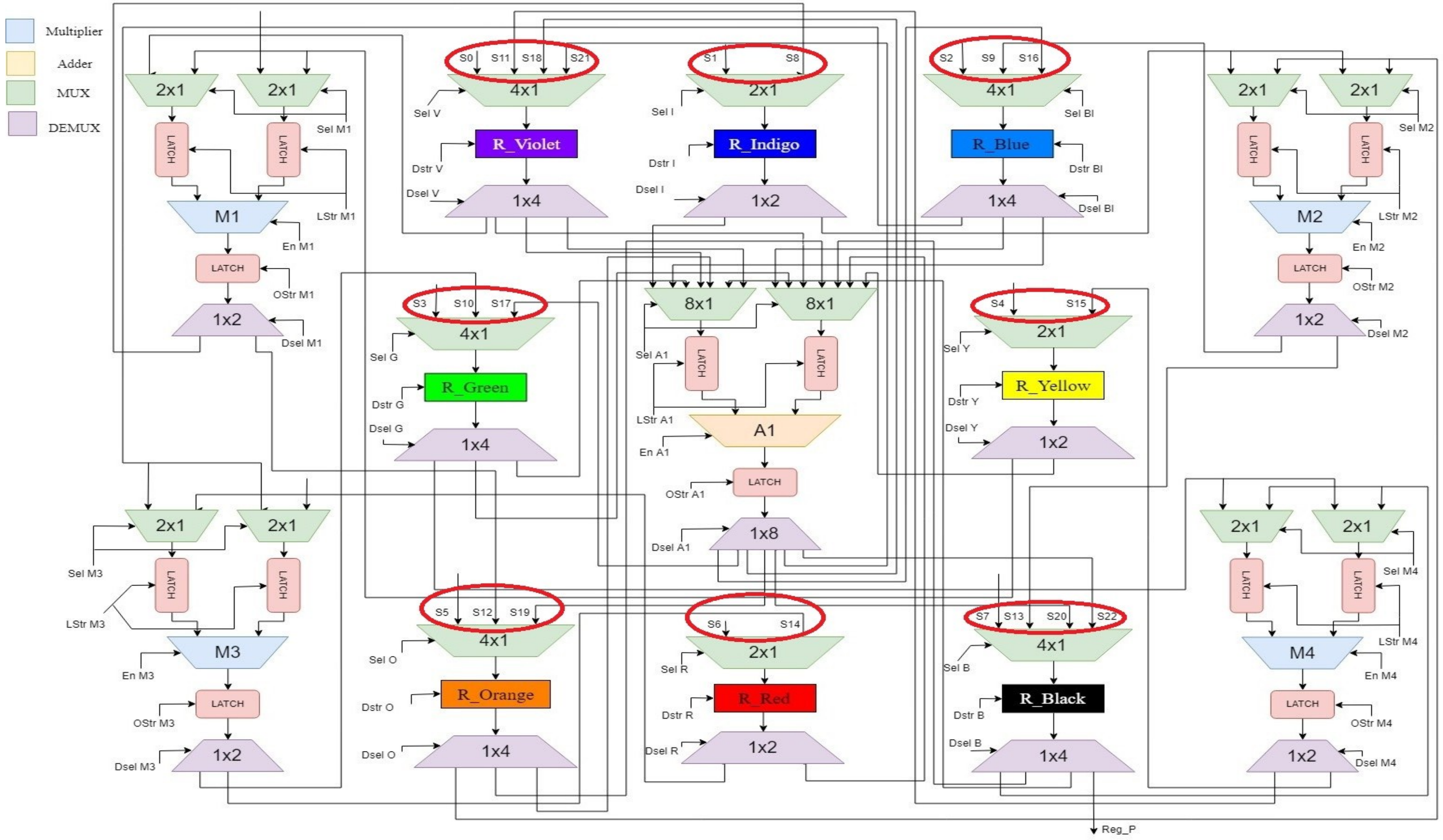


Fig. RTL datapath of 8-point DCT after Steganography

Security Features of Hardware Steganography

- Security features of hardware steganography are discussed in terms of following two aspects:

- ***Security Properties***

- *Security and reliability*
- *Robustness*
- *Tamper tolerance*

- ***How Does Steganography handle Threats and Attacks***

The hardware steganography handles different attacks as follows:

- *Unauthorized stego-constraints insertion*
- *Original stego- information tampering*
- *Unintended stego-constraints extraction*

Analysis on Case Studies

- ❑ Different DSP applications along with the 8-point DCT discussed in previous section have been analysed to evaluate performance of the hardware steganography.
- ❑ The impact of different values of E^t over the number of effective (non-default) constraints added and register overhead are analysed to assess the effectiveness of hardware steganography.

❖ The following criteria are used to analyse the performance of the hardware steganography in terms of the **security** and **design cost**:

(a) Probability of coincidence (P^c) metric to assess the strength of proof of ownership.

(b) Effect on Pc for different values of entropy threshold.

(c) Effect on register overhead for different values of entropy threshold.

(d) Effect on post-stego embedded design cost for different values of entropy threshold.

Security Analysis

❑ Security achieved through hardware steganography is assessed in terms of the probability of coincidence (P^c) metric given as follows:

$$P^c = \left(1 - \frac{1}{n}\right)^k \qquad (1)$$

- Where, ‘P^c’ denotes the probability of coincidence,
- ‘n’ denotes the number of colours in the CIG before implanting steganography and
- ‘k’ denotes the number of effective constraints (hidden edges) embedded into the CIG to process steganography.

Security Analysis

- The number of effective stego constraints-edges increases with the increasing entropy threshold E^t as shown in Table below:

Effect of entropy threshold value 'E^t=4' on probability of coincidence P^c of the steganography embedded IP cores

DSP application	# of storage variables	# of registers before steganography	# of constraints	P^c
FIR	30	8	20	6.9E-02
MPEG	41	14	21	2.1E-01
DCT	22	8	13	1.8E-01
EWF	35	7	12	1.6E-01
JPEG_IDCT	135	29	50	1.7E-01

Effect of entropy threshold value 'E^t=10' on probability of coincidence P^c of the steganography embedded IP cores

DSP application	# of storage variables	# of registers before steganography	# of constraints	P^c
FIR	30	8	57	4.9E-04
MPEG	41	14	59	1.3E-02
DCT	22	8	43	3.2E-03
EWF	35	7	86	1.8E-06
JPEG_IDCT	135	29	355	3.9E-06

Security Analysis

- Comparison with the contemporary approach (hardware watermarking) for DSP cores:

Comparison of steganography with watermarking in terms of the number of effective constraints imposed on DSP core design for ‘E^t= 3’

DSP application	# # of storage variables	number of effective constraints	# of effective constraints imposed	
			Hardware Steganography	Hardware Watermarking
FIR	30	17	17	1
MPEG	41	5	5	0
DCT	22	4	4	0
EWF	35	4	4	1
JPEG_IDCT	135	18	18	0

Comparison of steganography with hardware watermarking in terms of the number of effective constraints imposed on DSP core design for ‘E^t= 6’

DSP application	# # of storage variables	number of effective constraints	# of effective constraints imposed	
			Hardware Steganography	Hardware Watermarking
FIR	30	57	57	9
MPEG	41	52	52	6
DCT	22	24	24	2
EWF	35	34	34	7
JPEG_IDCT	135	203	203	11

Design Cost Analysis

- Design cost of the steganography embedded IP core is evaluated using following cost function:

$$C_d(R_i) = w_1 \frac{L_d}{L_{max}} + w_2 \frac{A_d}{A_{max}} \qquad (2)$$

- Where, $C_d(R_i)$ denotes the stego-embedded design cost with vendor specified resource configuration R_i.
- Here, design cost is evaluated in terms of the design latency and hardware area denoted by L_d and A_d respectively.
- Further, L_{max} and A_{max} denote the maximum execution latency and hardware area respectively and
- w_1, w_2 represent the user specified weights both fixed at 0.5 to assign equal preference.
- Both the area and latency of an IP core design are estimated using NanGate library based on 15 nm technology scale (http://www.nangate.com/?pageid=2328).

Design cost of the hardware steganography for various resource configurations

DSP core	Resource configurations	L_d (ps)	A_d (µm)	Design Cost
FIR	4A 4M	994	383.78	0.44437
MPEG	3A 7M	795	596.11	0.36169
DCT	1A 4M	928	327.156	0.47212
EWF	2A 1M	2716	118.75	0.66231
JPEG_IDCT	12A 12M	1988	1155.27	0.33271

Design Cost Analysis

- ❑ Further, the effect on register overhead for increase in magnitude of entropy threshold Et is shown here.

Effect of hardware steganography on register overhead with increase in threshold entropy

DSP application	# of storage variables	# of registers before steganography	Value of Threshold Entropy Threshold resulting into Re more registers				
			R^e=0	R^e =1	R^e =2	R^e =3	R^e =4
FIR	30	8	2-5	NA	NA	NA	NA
MPEG	41	14	2-7	NA	NA	NA	NA
DCT	22	8	2-6	7	NA	NA	NA
EWF	35	7	2-7	8	NA	9,10	11
JPEG_IDCT	135	29	2-9	NA	NA	NA	NA

Design Cost Analysis

- The effect on the design cost of IP cores before and after hardware steganography for $E^t = 4$ is presented. As shown in the tables, design cost remains same after the steganography is embedded for $E^t = 4$.

Design cost of the IP core **pre-steganography**

DSP application	# of Registers	Cost
FIR	8	0.44437
MPEG	14	0.36169
DCT	8	0.47212
EWF	7	0.66231
JPEG_IDCT	29	0.33271

Design cost of the IP core **post-steganography**

DSP application	Entropy Threshold (E^t)	# of Registers	Cost
FIR	4	8	0.44437
MPEG	5	14	0.36169
DCT	5	8	0.47212
EWF	7	7	0.66231
JPEG_IDCT	6	29	0.33271

Comparison with the contemporary approach (hardware watermarking) for DSP cores:

Comparison of Steganography with hardware watermarking in terms of the storage hardware for equivalent number of effective constraints for ‘E^t= 5’

DSP application	# of registers before steganography	# of effective constraints	# of storage hardware required after implanting equivalent number of constraints		% reduction in registers
			Steganography)	Hardware watermarking	
FIR	8	57	8	10	**20%**
MPEG	14	46	14	15	**6.6%**
DCT	8	18	8	10	**20%**
EWF	7	30	7	8	**12.5%**
JPEG_IDCT	29	124	29	30	**3.3%**

Comparison of Steganography with hardware watermarking in terms of the storage hardware for equivalent number of effective constraints for ‘E^t= 6’

DSP application	# of registers before steganography	number of effective constraints	# of storage hardware required after implanting equivalent number of constraints		% reduction in registers
			Steganography)	Hardware watermarking	
FIR	8	57	8	10	**20%**
MPEG	14	52	14	15	**6.6%**
DCT	8	24	8	10	**20%**
EWF	7	34	7	8	**12.5%**
JPEG_IDCT	29	203	29	30	**3.3%**

Conclusion

- ✓ The proposed approach highlights the necessity of IP core protection against various threats and presents a solution to achieve that using forensic detective control using hardware steganography.
- ✓ Hardware steganography is a promising alternative to hardware watermarking for forensic detective control.
- ✓ However, unlike watermarking, it is signature free and provides more designer control.
- ✓ That's why this approach provides full control to the designer over the amount of secret stego-constraints embedded into the design.
- ✓ Further, the steganography approach results into comprehensive forensic detective control while incurring nominal design overhead.

References

1. E. Castillo, U. Meyer-Baese, A. Garcia, L. Parilla, A. Lloris (May 2007), ‘IPP@HDL: Efficient intellectual property protection scheme for IP cores,’ *IEEE Trans. Very Large Scale Integr. (VLSI) Syst*., vol. 15, no. 5, pp. 578–590.
2. M.C. McFarland, A.C. Parker, R. Camposano (1988), ‘Tutorial on high-level synthesis’, *DAC '88 Proceedings of the 25th ACM/IEEE Design Automation,* vol. 27 (1), pp. 330-336.
3. S. M. Plaza, I. L. Markov (2015), ‘Solving the Third-Shift Problem in IC Piracy With Test-Aware Logic Locking,’ *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.,* vol. 34(6), pp. 961-971.
4. A. Sengupta (2017), ‘Hardware Security of CE Devices [Hardware Matters],’ *IEEE Consumer Electronics Mag,* vol. 6(1), pp. 130-133.
5. A. Sengupta (2016), ‘Intellectual Property Cores: Protection designs for CE products,’ *IEEE Consumer Electronics Mag*, vol. 5, no. 1, pp. 83-88. A. Sengupta, S. Bhadauria (2016), ‘Exploring Low Cost Optimal Watermark for Reusable IP Cores During High Level Synthesis,’ *IEEE Access*, vol. 4, pp. 2198-2215. A. Sengupta, D. Roy (2017), ‘Antipiracy-Aware IP Chipset Design for CE Devices: A Robust Watermarking Approach [Hardware Matters],’ *IEEE Consumer Electronics Mag*, vol. 6(2), pp. 118-124.
6. A. Sengupta, D. Roy (2018), ‘Multi-Phase Watermark for IP Core Protection,’ *Proc. 36th IEEE International Conference on Consumer Electronics (ICCE)*, pp. 1-3.
7. A. Sengupta, D. Roy, S. P. Mohanty (2018), ‘Triple-Phase Watermarking for Reusable IP Core Protection During Architecture Synthesis,’ *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst*, vol. 37(4), pp. 742-755.
8. A. Sengupta, M. Rathor (2019), IP Core Steganography for Protecting DSP Kernels used in CE Systems, *IEEE Transactions on Consumer Electronics (TCE)* , Volume: 65 , Issue: 4 , Nov. 2019, pp. 506 - 515.
9. B. Colombier, L. Bossuet (2015), ‘Survey of hardware protection of design data for integrated circuits and intellectual properties,’ *IET Computers & Digital Techniques*, vol. 8(6), pp. 274-287.
10. A. Obukhov, A. Kharlamov (2008) ‘Discrete Cosine Transform for 8x8 Blocks with CUDA’ Nvidia whitepaper document.

Thank You