B. TECH. PROJECT REPORT

On

Crypto Based Dual Phase Hardware Steganography for Securing DSP Circuits

BY

Rahul Kumar (160001047)

DISCIPLINE OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY INDORE

December 2019

Crypto Based Dual Phase Hardware Steganography for Securing DSP Circuits

A PROJECT REPORT

Submitted in partial fulfillment of the

requirements for the award of the degrees

of

BACHELOR OF TECHNOLOGY

in

COMPUER SCIENCE AND ENGINEERING

Submitted by:

Rahul Kumar (160001047)

Guided by:

Dr. Anirban Sengupta

INDIAN INSTITUTE OF TECHNOLOGY INDORE

December 2019

CANDIDATE’S DECLARATION

I hereby declare that the project entitled **“Crypto based dual phase hardware steganography for securing DSP circuits”** submitted in partial fulfillment for the award of the degree of Bachelor of Technology in ‘Computer Science and Engineering’ completed under the supervision of **Dr Anirban Sengupta, Computer Science and Engineering, Associate Professor,** IIT Indore is an authentic work.

Further, I declare that I have not submitted this work for the award of any other degree elsewhere.

Rahul Kumar (160001047)

CERTIFICATE by BTP Guide(s)

It is certified that the above statement made by the students is correct to the best of my knowledge.

Dr. Anirban Sengupta

Associate Professor

Computer Science and Engineering

IIT Indore

Preface

This report on “Crypto Based Dual Phase Hardware Steganography for Securing DSP Circuits" is prepared under the guidance of Dr. Anirban Sengupta.

I have tried to present the detailed concept of the process involved in implanting design constraint in a DFG. I have also added all the concepts which are required to generate the constraints which is to be embedded into the design. For better understanding of the process, I have also used a standard benchmark to create the design constraint using different keys.

Rahul Kumar

B.Tech. IV Year

Discipline of computer science and engineering

IIT Indore

Acknowledgements

I wish to thank Dr. Anirban Sengupta for his kind support and valuable guidance.

It is his help and support, due to which I became able to complete the design and technical report.

Without his support this report would not have been possible. It was only possible because of his enthusiasm, beforehand schedule, knowledge and sincerity towards me and my work to produce the better results. I wouldn’t have achieved my goals without his encouragement at each step.

Rahul Kumar

B.Tech. IV Year

Discipline of Computer Science and Engineering

IIT Indore

Abstract

An intellectual property (IP) core is susceptible to piracy. An adversary can deceitfully claim the ownership of a pirated IP core. In such cases of ownership conflict, the true ownership of an IP core should be provable. A novel approach of securing IP cores against piracy/ false claim of ownership using crypto-based dual phase steganography has been discussed in this text. By detecting the embedded robust stego-mark in the design, the ownership can be awarded to the genuine IP owner. This presents a novel security algorithm that leverages crypto-modules and security properties generate stego-constraints and embeds them into a hardware IP design during two distinct phases of behavioural synthesis. Because of using large size stego-keys and embedding steganography at two distinct phases, the proposed approach achieves robust security and high reliability than existing recent approaches.

Table of Contents

List of Figures

List of Tables

1

Introduction

With the mounting popularity of reusable intellectual property (IP) cores, security threats such as reverse engineering, piracy and hardware trojan insertion have become a serious problem for electronics design due to globalization of supply chain. The global semiconductor supply chain for SoC/IC design is highly susceptible to hardware attacks, such as trojan and IP piracy. However, globalization in the semiconductor supply chain is inevitable due to requirements of maximizing design productivity and minimizing design cycle time. Further, keeping the entire design and manufacturing process of IC design in house increases the nonrecurring cost significantly, especially for mass production commercial developments. It is estimated that 10% of the globally sold electronics products are counterfeited that leads to ~$100 billion of revenue loss. We need some mechanism through which a genuine vendor can prove his ownership over an IP easily and effectively.

2

Prior work

Against piracy, watermarking is a one of existing solutions in which owner's signature is embedded into the design. During ownership conflict, the embedded signature of true owner can be extracted to nullify the false claim [2]. Watermarking approaches [1], [2], [3] have limitations. For example, an adversary can steal author's secret signature to defeat the goal of watermarking. Additionally, selection of an appropriate signature that can provide higher security at lower design cost is challenging. Further, watermarking approaches do not involve crypto-keys to mathematically generate the hidden constraints which indicate that if the watermark signature is compromised, there is no way for an authentic owner to scientifically/mathematically prove his/her IP ownership rights. Additionally, watermarking approaches embeds hidden constraints only in single phase which weakens the proof of ownership.

3

Proposed work

To overcome the limitations of watermarking, a novel method of securing ownership using crypto-based dual phase steganography is proposed. The proposed approach uses multi-layered stego-keys besides implanting hidden constraints in dual phase of high level synthesis for enhanced security and ownership proof. The proposed approach offers two major advantages (i) generated secret information (stego-constraints) to be embedded are mathematically obtained using secret design data and large size stego-keys (only known to the owner). Hence unlike IP watermarking, even if the secret constraints are compromised by an adversary, a genuine owner is still able to prove the generated hidden constraints mathematically, thus proving ownership (ii) the proposed steganography approach is signature free, hence provides more designer's control.

3.1 Overview

The proposed approach generates stego-constraint to be implanted based on the following security modules/properties:

a. Bit manipulation
b. Row and column diffusion
c. Multi-layered trifid cipher
d. Alphabet substitution
e. Matrix transposition
f. Mix-column diffusion
g. Byte concatenation
h. Bit-stream truncation
i. Bit encoding

The roles of the individual transformations on the overall robustness are highlighted in table 1. The proposed approach for generating a steganography embedded IP core is shown in Fig 1.

The primary input to the proposed approach are :

a. Data flow graph (DFG) representing IP core
b. Module library containing data about area and latency of operators
c. Resource constraint
d. Stego keys

Based on resource constraint and module library, the DFG is scheduled using LIST scheduling. Further a colored interval graph (CIG) is obtained from thr scheduled DFG. Using the CIG , secret design data is taken out and steganography constraint to be implanted in the IP core is generated through the steps stated above. Once the stego-constraints are generated, they are implanted into the CIG and scheduled DFG. Thus steganography embedded IP core is generated.

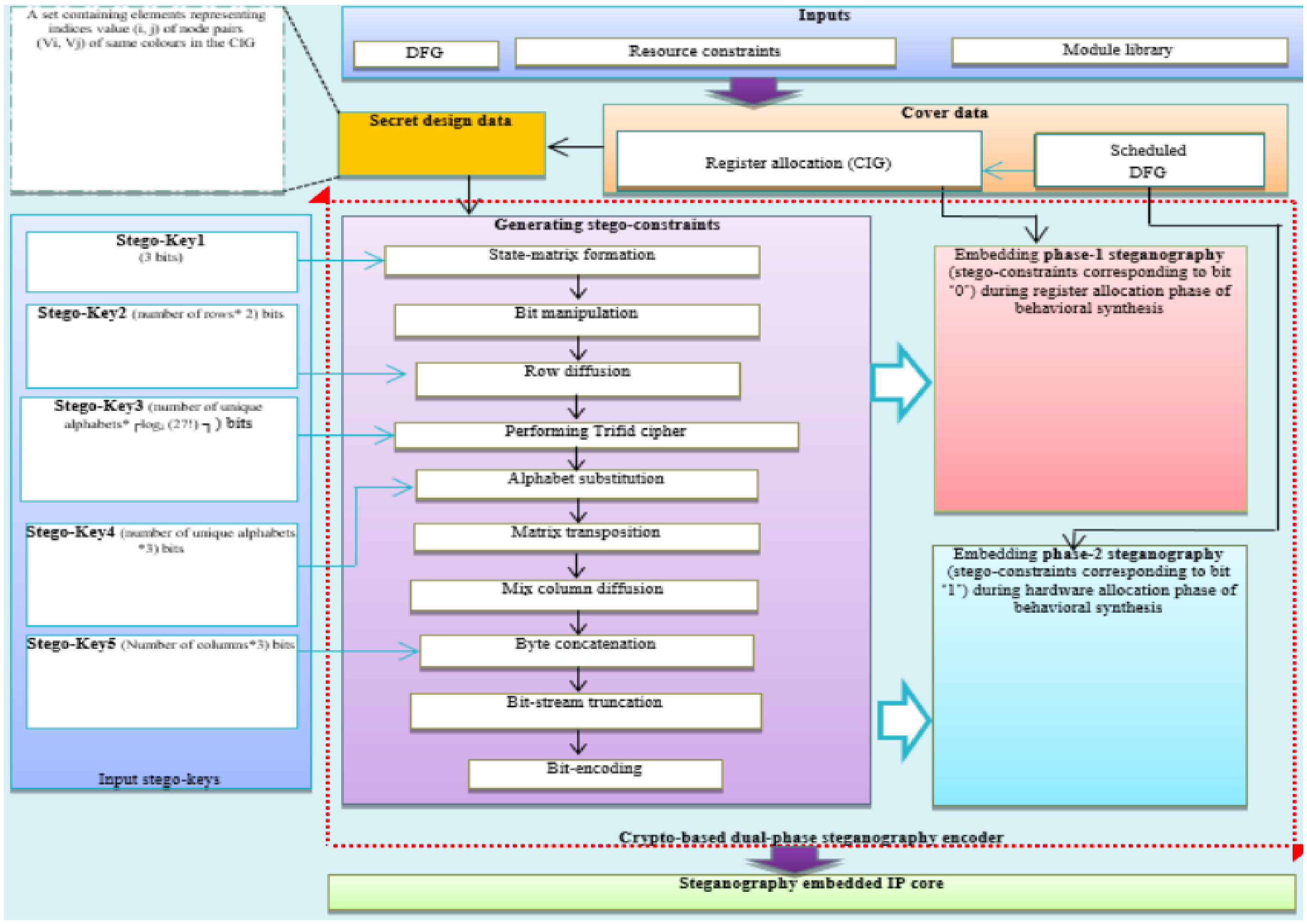

A set containing elements representing indices value (i, j) of node pairs (Vi, Vj) of same colours in the CIG
Inputs
DFG
Resource constraints
Module library
Secret design data
Cover data
Register allocation (CIG)
Scheduled DFG
Stego-Key1 (3 bits)
Stego-Key2 (number of rows* 2) bits
Stego-Key3 (number of unique alphabets* ⌈log₂ (27!)⌉) bits
Stego-Key4 (number of unique alphabets *3) bits
Stego-Key5 (Number of columns*3) bits
Input stego-keys
Generating stego-constraints
State-matrix formation
Bit manipulation
Row diffusion
Performing Trifid cipher
Alphabet substitution
Matrix transposition
Mix column diffusion
Byte concatenation
Bit-stream truncation
Bit-encoding
Embedding phase-1 steganography (stego-constraints corresponding to bit "0") during register allocation phase of behavioral synthesis
Embedding phase-2 steganography (stego-constraints corresponding to bit "1") during hardware allocation phase of behavioral synthesis
Crypto-based dual-phase steganography encoder
Steganography embedded IP core

Fig 1:- proposed crypto-based dual phase steganography approach

Bit manipulation	To obscure the relationship between the key and the generated stego-constraints (shannon's property of confusion)
Key based row diffusion	To obscure the relationship between input secret design data and generated stego-constraints (shannon's property of diffusion)
Key based trifid cipher and alphabet substitution	To provide following features: fractionation, transposition, substition. These features achieve certain amount of confusion and diffusion properties for obscurity.
Matrix transposition and mix column diffusion	To increase the obscurity between input secret design data and generated stego-constraints.
Key based byte concatenation	To further increase the Shannon's property of diffusion for enhanced onscurity.

Table 1:- Roles of individual transformations on overall robustness of proposed hardware steganography

Details of proposed approach

We use following inputs to generate stego-embedded IP core :

(a) *Cover design data.* The CIG and scheduled DFG act as cover data for proposed hardware steganography. This is because the proposed steganography information is embedded in both the above forms of cover design, more explicitly, phase-1 steganography is embedded into the CIG and phase-2 steganography is embedded into the scheduled DFG. Further, the secret design data is obtained from the CIG.

(b) *Secret design data.* A set containing elements representing indices value (i, j) of node pairs (V_i, V_j) of same colors in the CIG.

(c) *Stego-keys* . The proposed approach uses five different stego-keys which are fed into the following security modules: state matrix formation, row diffusion, trifid cipher, alphabet substitution, byte concatenation. Fig 1 shows the size of each stego-keys.

3.2 Secret design data extraction

The DFG representing IP core is scheduled and hardware allocated based on designer's specified resource constraints. After the CIG is obtained from the scheduled DFG, secret design data is extracted from the CIG. Secret design data is a set containing elements representing indices value (i, j) of node pairs (V_i, V_j) of same colors in the CIG. The value of node pairs (V_i, V_j) is converted such that value V_i and V_j is always less than 16 by taking modulus by 16 so that V_i and V_j can be represented in 4-bit binary.

3.3 State matrix formation

A subset of the secret design data is chosen based on stego-key 1 which is shown in table 2, we have to choose a 3-bit key. This subset is converted into a state matrix such that consecutive four elements of this set is present in each row of the matrix.

000	Choose 2 pairs and skip 2
001	Choose 4 pairs and skip 4
010	Choose 8 pairs and skip 8
011	Choose 16 pairs and skip 16
100	Choose 32 pairs and skip 32
101	Choose 64 pairs and skip 64

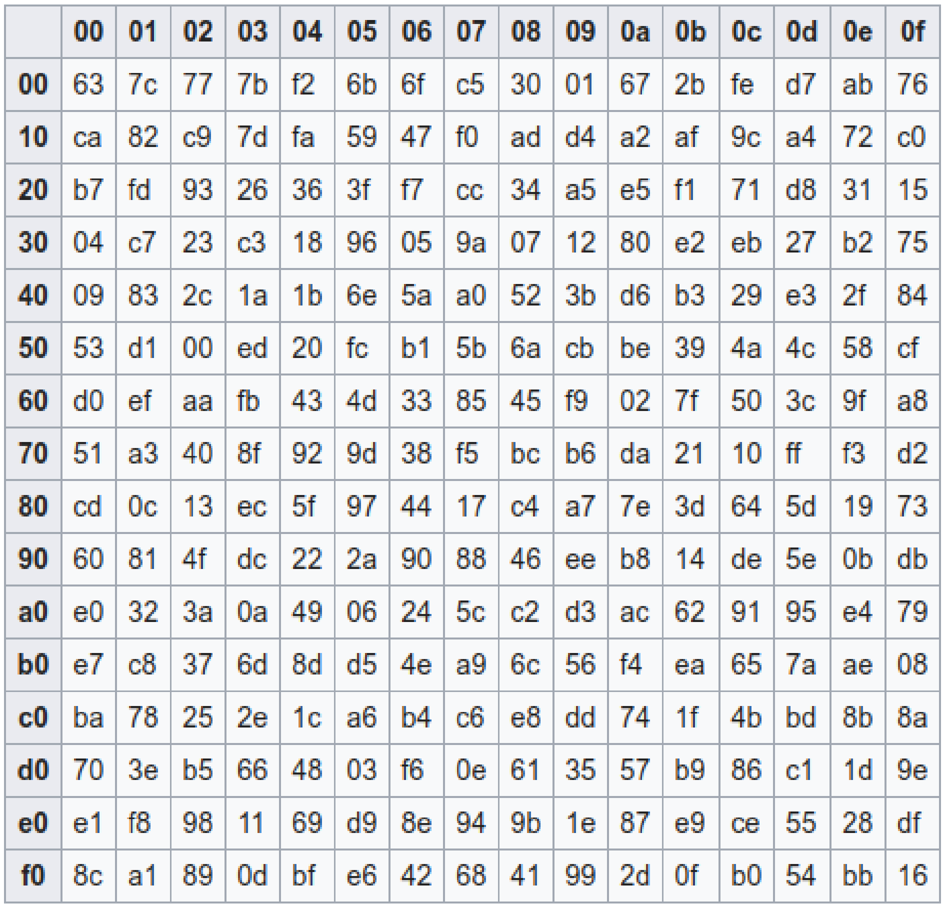
Table 2 : stego key 1

3.4 Bit manipulation using AES S-box

In cryptography, an S-box is a basic component of symmetric algorithm which performs substitution.In block ciphers, they are typically used to obscure the relationship between the key and the ciphertext to introduce shannon's property of confusion.

The AES S-box maps 8-bit input to an 8-bit output. AES S-box is shown in Fig 2.

AES bit manipulation is applied to the previous state matrix using the S-box. Now , we obtain the bit-manipulated matrix.



	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
10	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
20	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
30	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
40	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
50	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
60	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
70	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
80	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
90	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a0	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b0	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c0	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d0	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e0	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f0	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Fig 2 : AES S-box

3.5 Row Diffusion

In this step we will circularly right rotate each row of the state matrix obtained in previous step. Rotation is done in the matrix bases on designer's key (stego-key -2) . For each row of the matrix there is a 2-bit key as defined in table 3. This step further obscures the relationship between input secret design data and generated stego-constraint.

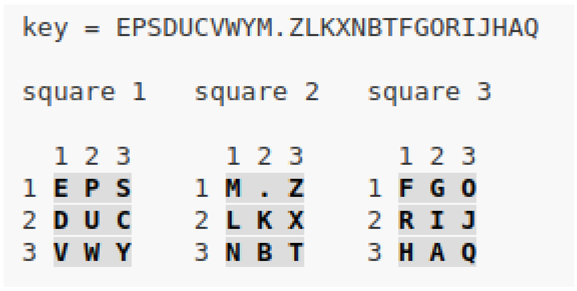
Size of stego-key 2 = 2*number of rows in matrix

00	1 step
01	2 step
10	3 step
11	4 step

Table 3 : stego-key 2

3.6 Trifid cipher

Trifid cipher combines substitution with transposition and fractionation. We need a 27 character long key for the trifid cipher. Arrange these keys into three 3×3 matrix and find the coordinate of the required character. Now, we will further process this coordinate.



```
key = EPSDUCVWYM.ZLKXNBTFGORIJHAQ

square 1   square 2   square 3

  1 2 3      1 2 3      1 2 3
1 E P S    1 M . Z    1 F G O
2 D U C    2 L K X    2 R I J
3 V W Y    3 N B T    3 H A Q
```

For example, coordinate of “D” in following trifid cipher is (2, 1, 1) and “W” is (3, 2, 1).

In previous state matrix, possible numbers in the matrix are 0 to F in hex. For each unique alphabet greater than or equal to “A” in state matrix, we will have a 27 character long key (stego-key 3). Stego-key 3 is used to find the encrypted values(co-ordinate) for the unique alphabet.

3.7 Alphabet substitution

The coordinate (x, y, z) corresponding to the alphabets are converted into a number less than 10 using key(stego key 4) based function shown in table 4.

The number obtained from the function is substituted with their corresponding alphabet, thus making all the numbers less than 10.

The state matrix is further transposed .

000	(X*Y*Z)
001	(X+Y+Z)
010	|X-Y-Z |
011	|X-Y+Z |
100	(Z+Y)/X
101	(Z+X)/Y

Table 4 : function corresponding to stego-key 4

3.8 Mix-column diffusion

Mix column diffusion is applied on the transposed matrix from the previous step.

Each column $[\ b_0\ \ b_1\ \ b_2\ \ b_3\ ]^T$ is transformed into $[\ d_0\ \ d_1\ \ d_2\ \ d_3\ ]^T$ based on following rule :-

$$
\begin{aligned}
d_0 &= 2 \bullet b_0 \oplus 3 \bullet b_1 \oplus 1 \bullet b_2 \oplus 1 \bullet b_3 \\
d_1 &= 1 \bullet b_0 \oplus 2 \bullet b_1 \oplus 3 \bullet b_2 \oplus 1 \bullet b_3 \\
d_2 &= 1 \bullet b_0 \oplus 1 \bullet b_1 \oplus 2 \bullet b_2 \oplus 3 \bullet b_3 \\
d_3 &= 3 \bullet b_0 \oplus 1 \bullet b_1 \oplus 1 \bullet b_2 \oplus 2 \bullet b_3
\end{aligned}
$$

Where A•B is found using below algorithm:-

```
mul( A, B){
        If(A==1)return B
        C=B;
        C<<1;
        If 8th bit is set in C then C=C xor 283
        If A==2    return C
        Return C xor B
}
```

3.9 Byte concatenation

Now , we will concatenate bytes of each column based on a key (stego key 5). Stego key 5 defines the order in which each byte of the column from the matrix will be concatenated. Stego key 5 is shown in table 5.

key	order
000	{0, 1, 2, 3}
001	{0, 1, 3, 2}
010	{0, 2, 1, 3}
011	{0, 2, 3, 1}
100	{0, 3, 1, 2}
101	{0, 3, 2, 1}

Table 5 :- stego key 5

Size of stego key 5 will be 3*number of column in matrix.

Now this byte string is converted to bit string and and a contraint size is selected by the designer to embed the constraint.

3.10 Constraint embedding

Constraints are embedded in two phases into the design. In phase-1, zero's in the selected bit string is embedded into the CIG of the DFG. In phase-2, one's in the bit string is embedded into the list scheduling of the DFG based on below encoding rule.

Encoding rule :-

'0' -   Embed an edge between node pair (even, even) into the CIG.

'1' - In the scheduled and hardware allocated DFG, odd operations are assigned to functional unit(FU) of vendor type 1 and even operations to vendor type 2.

3.11 Cost calculation

Formula for calculating cost is shown below

$$cost = \frac{1}{2}(\frac{area}{mx\_area} + \frac{latency}{mx\_latency})$$

Where,

- area = area calculated for current design
- mx_area = area calculated using maximum constraint
- latency = latency calculated for current design
- mx_latency = latency calculated for minimum constraint

3.12 Security Analysis

Security of proposed hardware steganography is analysed in terms of strength of ownership proof and crypto-key size indicating resilience against attack. The strength of ownership is measured using probability of coincidence (P_c).

Probability of coincidence is defined as the probability for an attacker to get the right constraint. Its formula is shown below

$$P_c = (1 - \frac{1}{c})^{f1} * (1 - \frac{1}{\prod_{d=1}^{D} N(R_d)})^{f2}$$

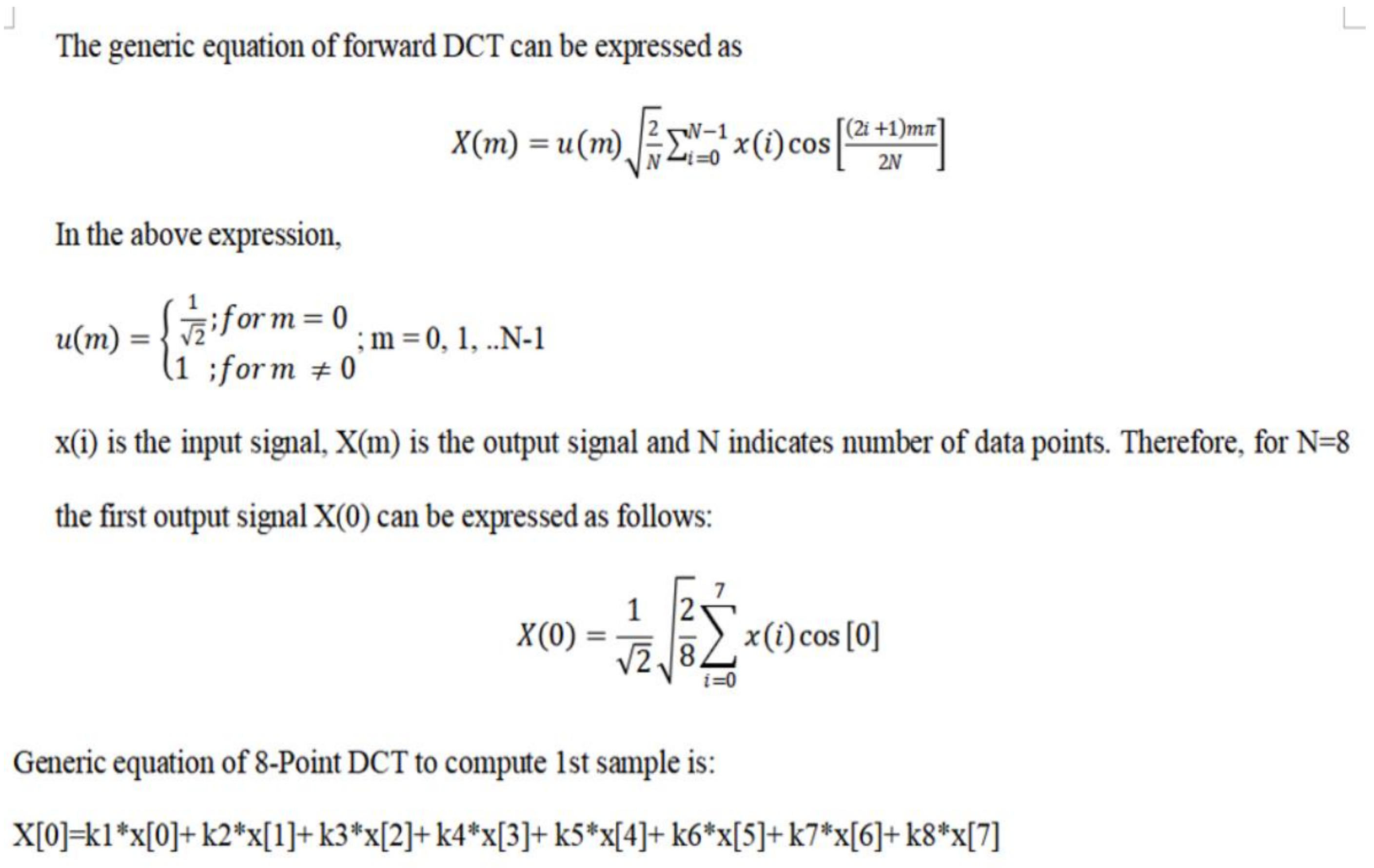
Where,

- C = number of register before steganography
- f1 = number of zero's
- f2 = number of effective one's implanted in design
- D = total number of resource type
- $N(R_d)$ = number of resource of type d

4

Demonstration using 8-point DCT

The proposed approach for embedding hardware steganography is demonstrated using 8-point discrete cosine transform (DCT) core.



The generic equation of forward DCT can be expressed as

$$X(m) = u(m)\sqrt{\frac{2}{N}}\sum_{i=0}^{N-1} x(i)\cos\left[\frac{(2i+1)m\pi}{2N}\right]$$

In the above expression,

$$u(m) = \begin{cases} \frac{1}{\sqrt{2}}; for\ m = 0 \\ 1\ \ ; for\ m\ \neq 0 \end{cases}; \text{m} = 0,\ 1,\ ..\text{N-1}$$

x(i) is the input signal, X(m) is the output signal and N indicates number of data points. Therefore, for N=8 the first output signal X(0) can be expressed as follows:

$$X(0) = \frac{1}{\sqrt{2}}\sqrt{\frac{2}{8}}\sum_{i=0}^{7} x(i)\cos[0]$$

Generic equation of 8-Point DCT to compute 1st sample is:

X[0]=k1*x[0]+ k2*x[1]+ k3*x[2]+ k4*x[3]+ k5*x[4]+ k6*x[5]+ k7*x[6]+ k8*x[7]

Scheduling and hardware allocation

The DFG representation of DCT core is scheduled and hardware allocated based on designer's specified constraints, say (1 +, 4*) as shown in fig 3. The register allocation table of the design is shown in table 6 and corresponding CIG is shown in fig 4. The scheduled DFG and CIG act as cover data.

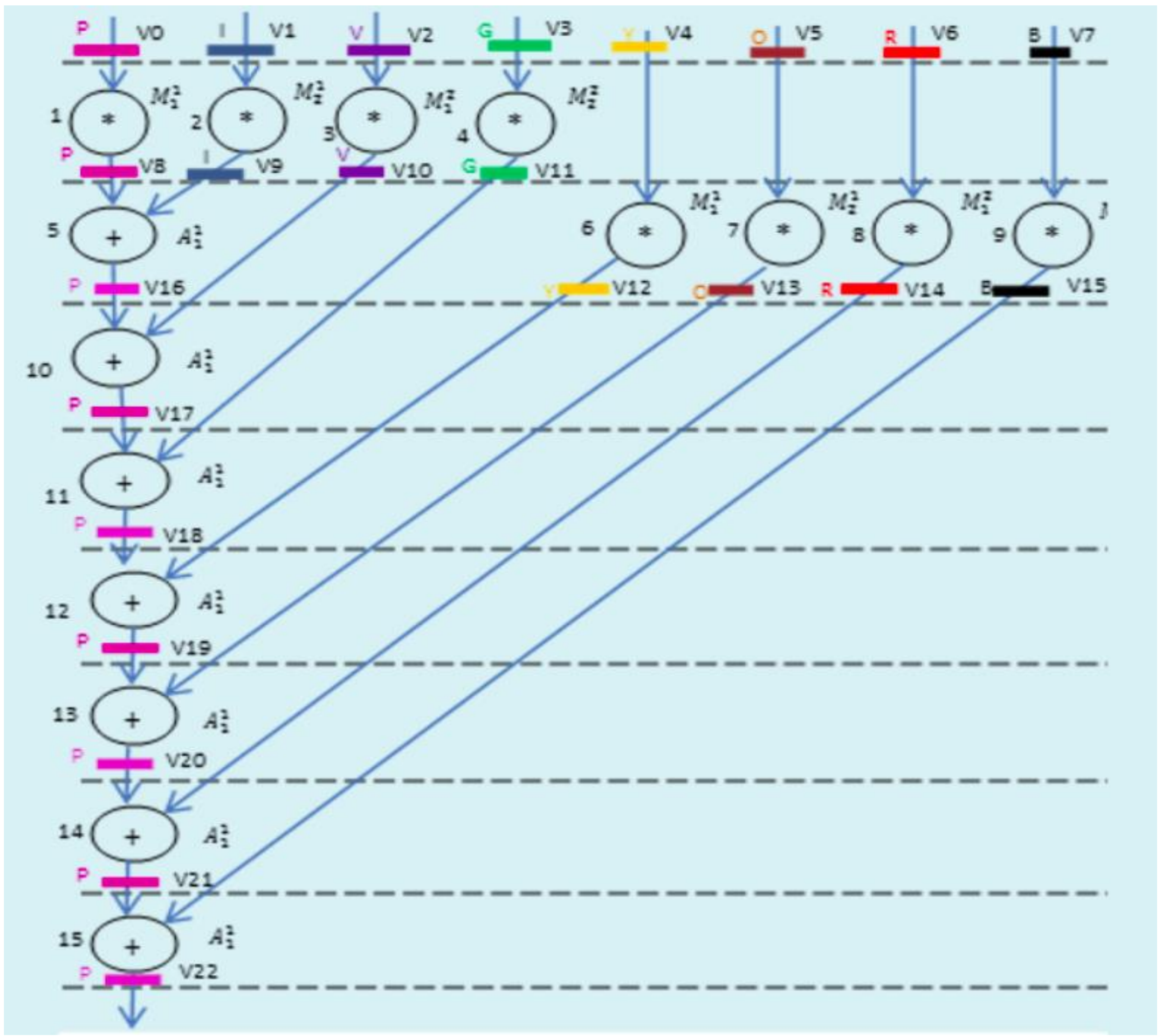


Fig 3: scheduled and hardware allocated 8-point DCT using 1(+) and 4(*) before implanting stego constraint

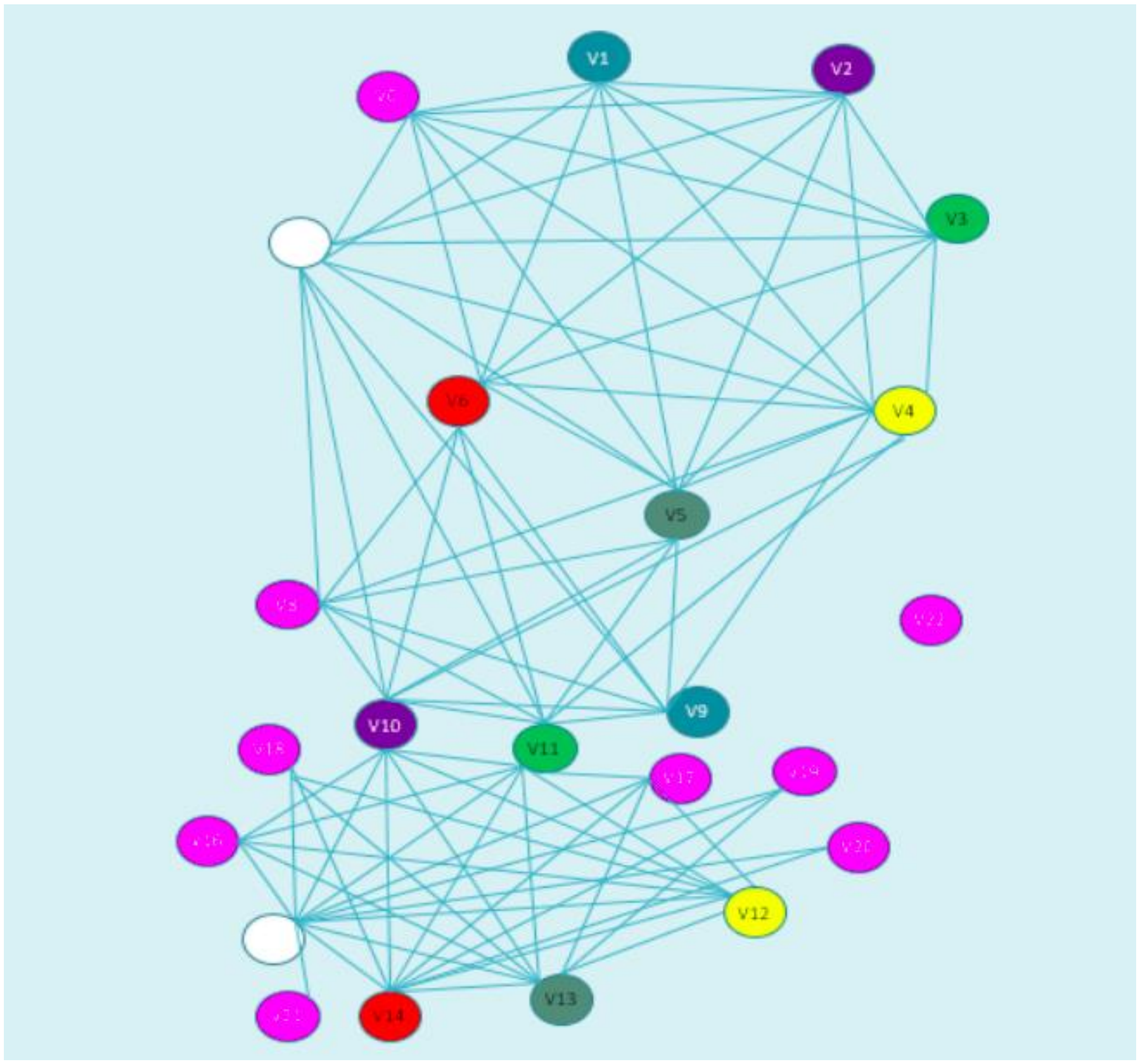


Fig 4:- A CIG of 8-point DCT before steganography

T	Pink	Indigo	violet	green	yellow	orange	red	black
0	v0	v1	v2	v3	v4	v5	v6	v7
1	v8	v9	v10	v11	v4	v5	v6	v7
2	v16		v10	v11	v12	v13	v14	v15
3	v17			v11	v12	v13	v14	v15
4	v18				v12	v13	v14	v15
5	v19					v13	v14	v15
6	v20						v14	v15
7	v21							v15
8	v22							

Table 6:- initial register allocation table for CIG

Extraction of secret design data

The secret design data extracted from the CIG is as follows:

A={(0,8), (0,16), (0,17), (0,18), (0,19), (0,20),(0,21), (0,22), (8,16), (8,17), (8,18), (8,19), (8,20), (8,21), (8,22), (16,17), (16,18), (16,19), (16,20), (16,21), (16,22), (17,18), (17,19), (17,20), (17,21), (17,22) , (18,19), (18,20), (18,21), (18,22), (19,20), (19,21), (19,22), (20,21), (20,22), (21,22), (1,9), (2,10), (3,11), (4,12), (5,13), (6,14), (7,15)}

In case of more than 16 nodes , we will reduce the value to 0-15 by taking modulus.

Design data after reducing node value:-

B= {(0,8), (0,1), (0,2), (0,3), (0,4), (0,5),(0,6), (0,7), (8,1), (8,2), (8,3), (8,4), (8,5), (8,6), (8,7), (1,2), (1,3), (1,4), (1,5), (1,6), (1,7), (2,3), (2,4), (2,5), (2,6), (2,7) , (3,4), (3,5), (3,6), (3,7), (4,5), (4,6), (4,7), (5,6), (5,7), (6,7), (1,9), (2,A), (3,B), (4,C), (5,D), (6,E), (7,F)}

Choose subset using stego-key 1

For stego-key 1=001 i.e. select consecutive 4 element and discard consecutive 4 element

The state matrix containing 4 elements in each row is formed using the set obtained using stego-key 1 is shown in fig 5.

08	01	02	03
81	82	83	84
13	14	15	16
26	27	34	35
47	56	57	67

Fig 5:- Initial state matrix

Bit manipulation

Bit manipulation is performed using forward S-box. Fig 6 shows matrix after bit manipulation.

30	7C	77	7B
0C	13	EC	5F
7D	FA	59	47
F7	CC	18	96
A0	B1	5B	85

Fig 6:- post bit manipulation

Row diffusion

Row diffusion is performed in each row based on stego-key 2.

Stego key2 = 01-00-10-00-11

Post row diffusion, matrix is shown in fig 7.

77	7B	30	7C
5F	0C	13	EC
FA	59	47	3D
96	F7	CC	18
A0	B1	5B	85

Fig 7:- post row diffusion

Trifid cipher

Trifid cipher is performed on each unique alphabet based on stego-key 3. Stego-key 3 for each alphabet is given below

A :-V$QAWSEDRFTGYHUJIKOLPZMXNCB

B :- QAWSEDRFTGYHUJIK$OLPZMXNCBV

C:- OLPZMXNCBV$QAWSEDRFTGYHUJIK

D:- GYHUJIK$OLPZMXNCBVQAWSEDRFT

E:- FTGYHUJIKOLPZMXNCBV$QAWSEDR

F:- LPZMXNCBVQAWSEDRFTGYHUJIK$O

The output of the trifid cipher showing the encrypted value for unique alphabet A, B, C, D, E, F are 211, 323, 321, 233, 313, 322 respectively.

The equivalent digit corresponding to the encrypted value of each unique alphabet is computed using stego-key 4.

Stego-key 4= 001-001-000-010-101-010 (A-F respectively 3 bit for each alphabet)

The computed equivalent digits for alphabets are :-

Alphabet	Function	Equivalent digit
A	(X+Y+Z)	4
B	(X+Y+Z)	8
C	(X*Y*Z)	6
D	|X-Y-Z|	4
E	(Z+X)/Y	6
F	|X-Y-Z|	1

The alphabets in the state matrix are substituted with their equivalent digits. The substituted state matrix is shown in fig 8.

77	78	30	76
51	06	13	66
14	59	47	74
96	17	66	18
40	81	58	85

Fig. 8:- post alphabet substitution

The matrix is now transposed. Transposed matrix is shown in fig 9.

77	51	14	96	40
78	06	59	17	81
30	13	47	66	58
76	66	74	18	85

Fig 9 :- post transposition

Mix column diffusion

Mix column diffusion is applied to each column of the state matrix. Fig 10 shows matrix after mix-column diffusion.

20	DD	F0	70	C5
A1	0E	1B	0A	34
F5	DB	5F	65	E5
3D	2A	CA	E0	08

Fig 10:- post mix-column diffusion

Byte concatenation

Stego-key 5 = 001-000-010-101-000

Byte string = 20A13DF5DD0EBD2AF05F1BCA70E0650AC534E508

This byte stream is converted into bit stream

Bit string =

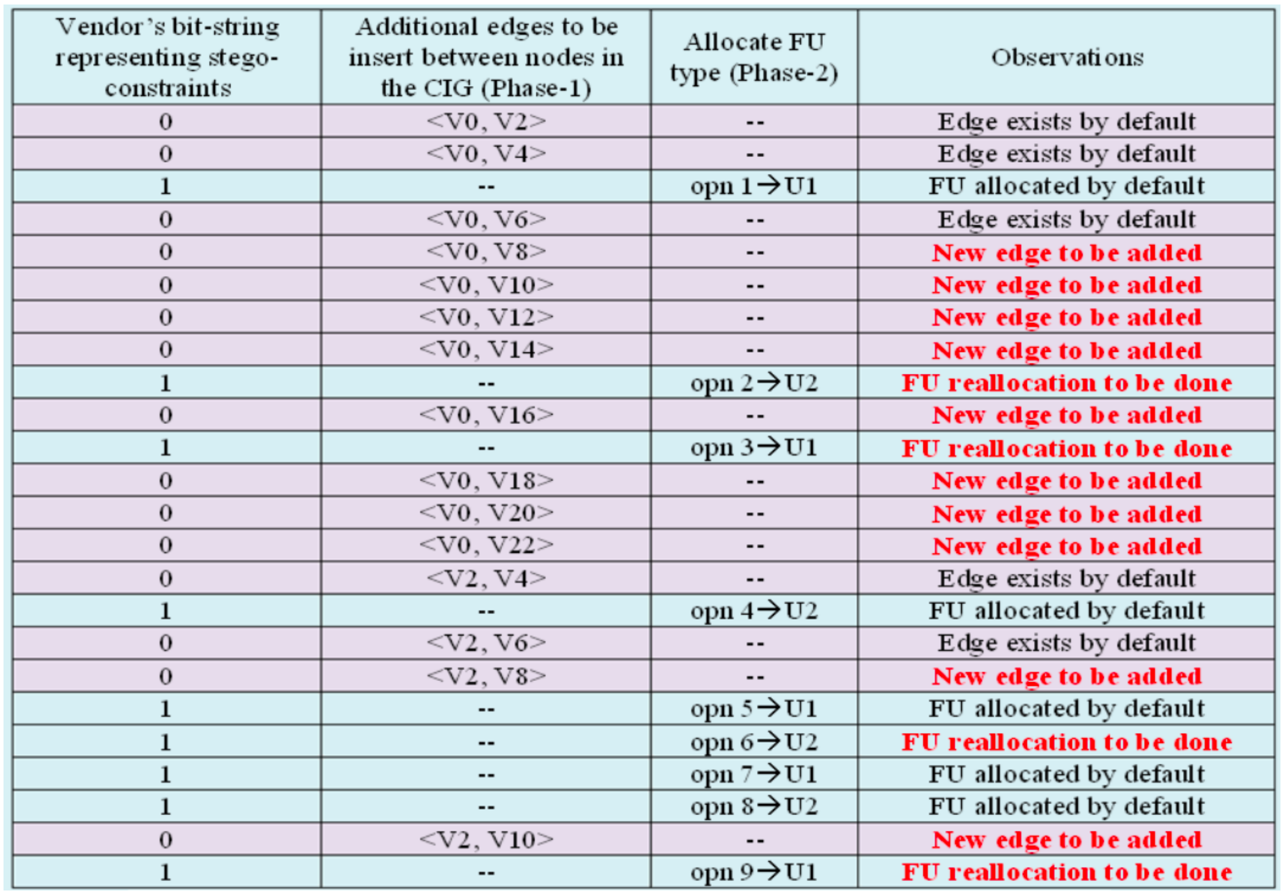
00100000101000010011110111110101110111010000111011011011001010101111000001011111000110111100101001110000111000000011

Let the constraint size which is to be implanted be 48.

Bit string to be implanted= 001000001010000100111101111101011101110100001110

Encoding

Zero's in the final bit string is implanted into CIG . Fig 11 shows the process of insertion of constraint. Fig-12 and table 7 shows CIG and register allocation table after implanting constraints respectively. New edges in the CIG is shown in red and the variables whose allocated register has changed has been shown in red in register allocation table.



Vendor's bit-string representing stego-constraints	Additional edges to be insert between nodes in the CIG (Phase-1)	Allocate FU type (Phase-2)	Observations
0	<V0, V2>	--	Edge exists by default
0	<V0, V4>	--	Edge exists by default
1	--	opn 1→U1	FU allocated by default
0	<V0, V6>	--	Edge exists by default
0	<V0, V8>	--	**New edge to be added**
0	<V0, V10>	--	**New edge to be added**
0	<V0, V12>	--	**New edge to be added**
0	<V0, V14>	--	**New edge to be added**
1	--	opn 2→U2	**FU reallocation to be done**
0	<V0, V16>	--	**New edge to be added**
1	--	opn 3→U1	**FU reallocation to be done**
0	<V0, V18>	--	**New edge to be added**
0	<V0, V20>	--	**New edge to be added**
0	<V0, V22>	--	**New edge to be added**
0	<V2, V4>	--	Edge exists by default
1	--	opn 4→U2	FU allocated by default
0	<V2, V6>	--	Edge exists by default
0	<V2, V8>	--	**New edge to be added**
1	--	opn 5→U1	FU allocated by default
1	--	opn 6→U2	**FU reallocation to be done**
1	--	opn 7→U1	FU allocated by default
1	--	opn 8→U2	FU allocated by default
0	<V2, V10>	--	**New edge to be added**
1	--	opn 9→U1	**FU reallocation to be done**

Fig 11:- Insertion process of constraint

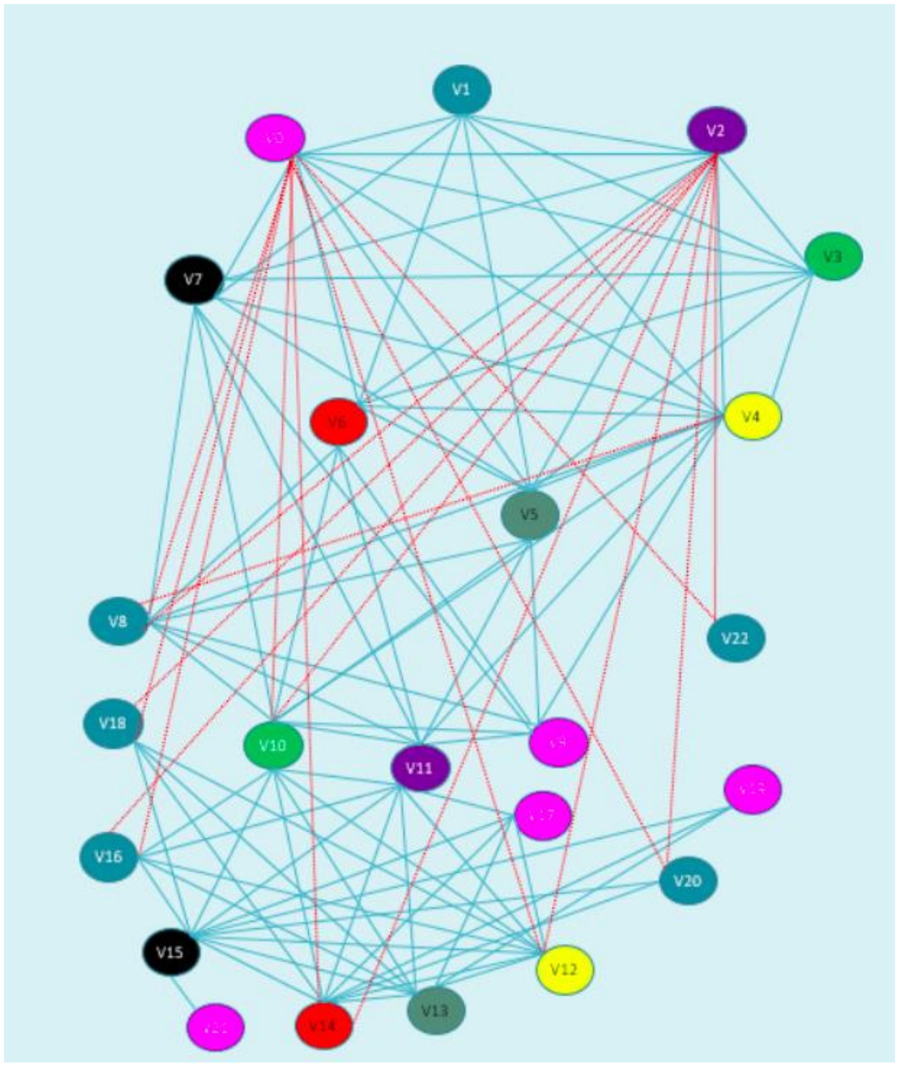


Fig 12:- CIG of 8-point DCT after implanting steganography constraint

T	Pink	Indigo	violet	green	yellow	orange	red	black
0	v0	v1	v2	v3	v4	v5	v6	v7
1	v9	v8	v11	v10	v4	v5	v6	v7
2		v16	v11	v10	v12	v13	v14	v15
3	v17		v11		v12	v13	v14	v15
4		v18			v12	v13	v14	v15
5	v19					v13	v14	v15
6		v20					v14	v15
7	v21							v15
8		v22						

Table 7:- register allocation table after implanting constraint in 8-point DCT

5. Results and Analysis

The proposed approach was implemented in CPP and run on 4GB DDR3 memory at 2.5 GHz.

Table 8 shows the effective number of constraints embedded in each phase of the constraint embedding process in different benchmark. Table 9 compares the value of P_c between related and proposed work. P_c of proposed work is lower than the related work.

Benchmarks	Proposed work	
	f1 (effective # of 0s)	f2 (effective # of 1s)
8-point DCT	24	12
FIR	20	23
JPEG_IDCT	203	109
MPEG	52	23
JPEG_sampl e	30	31
EWF	34	28

Table 8:- total stego-constraint (f1+ f2) embedded in proposed work

Benchmarks	Related work (Pc)	Proposed work (Pc)
8-point DCT	4.06e-2	1.29e-3
FIR	6.92e-2	1.57e-2
JPEG_IDCT	8.03e-4	3.78e-4
MPEG	2.12e-2	6.92e-3
JPEG_sample	7.36e-2	1.91e-3
EWF	5.29e-3	1.53e-4

Table 9 :- comparison of P_c between proposed work and related work [2]

Table 10 compares the design cost for different benchmark after phase-1 and phase-2 with the baseline. Design cost for most of the benchmark did not change after phase-1. This means that number of register remained same for those benchmark after constraint embedding. Fig 13 shows total key sizes for different benchmark.

Benchmarks	Design cost (Baseline)	Design cost post phase-1	Design cost post phase-2
8-point DCT	0.4535	0.4535	0.4535
FIR	0.4405	0.4405	0.4426
JPEG_IDCT	0.3034	0.3034	0.3034
MPEG	0.3736	0.3743	0.3748
JPEG_sample	0.4732	0.4732	0.4778
EWF	0.6610	0.6627	0.6632

Table 10:- comparison of design cost of proposed work with respect to baseline

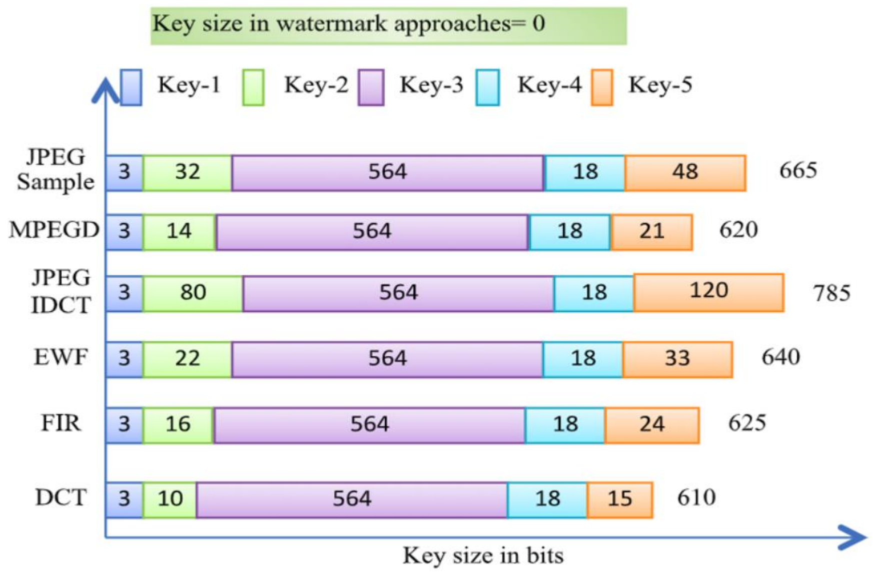


Fig 13:- key size for different DSP benchmark

6.Conclusion

Protection of IP core is of prominence these days because of globalization of design supply chain. A designer at times may have to compromise with other design metrics such as area and delay during incorporating protection feature within an IP. This has always been a challenge for a designer to generate a highly secured design while keeping design overhead as minimal as possible. This paper illustrated how proposed work obtain a secure IP with minimal overhead. Through our work we tried to bring new innovation in this field. This kind of work has never been proposed in the literature before. We have achieved very less overhead in terms of cost with results in the strong security.

References

[1] F. Koushanfar, I. Hong, and M. Potkonjak, “Behavioral synthesis techniques for intellectual property protection,” ACM Trans. Des. Autom. Electron. Syst., vol. 10, no. 3, pp. 523–545, 2005.

[2] A. Sengupta and S. Bhadauria, “Exploring low cost optimal watermark for reusable IP cores during high level synthesis,” IEEE Access, vol. 4, pp. 2198– 2215, 2016.

[3] A. Sengupta and D. Roy, “Anti-piracy aware IP chipset design for CE devices: Robust watermarking approach,” IEEE Consum. Electron. Mag., vol. 6, no. 2, pp. 118–124, Apr. 2017.

[4] B. Le Gal and L. Bossuet, “Automatic low-cost IP watermarking technique based on output mark insertions,” Des. Autom. Embedded Syst., vol. 16, no. 2, pp. 71–92, 2012.

[5] B. Colombier and L. Bossuet, “Survey of hardware protection of design data for integrated circuits and intellectual properties,” IET Comput. Digital Tech., vol. 8, no. 6, pp. 274–287, 2015.

[6] NanGate 15 nm open cell library: http://www.nangate.com/?pageid=2328, Jun. 2019

[7] A. Sengupta, Dipanjan Roy "Protecting an Intellectual Property Core during Architectural Synthesis using High-Level Transformation Based Obfuscation" IET Electronics Letters, Volume: 53, Issue: 13, June 2017, pp. 849 - 851.

[8] A. Sengupta "Hardware Vulnerabilities and its Effect on CE Devices: Design-for-Security against Trojan", IEEE Consumer Electronics, Volume: 6, Issue: 3, July 2017, pp. 126 – 133