Hardware Security, IP Core Protection and High Level Synthesis (HLS)

Hardware Security:

- Hardware security is one of the major aspects of cyber security.
- **Cyber Security**: Cyber security/ computer security ensures the protection computers, electronic systems, mobile devices, servers, networks, and data from malicious attacks.
- The current rapid development of modern technical society is heavily dependent on the usages of various types of electronic systems/ gadgets.
- **Examples of electronic systems used rapidly evolving modern technical society:** Computers, tablets, smartwatches, tablets, digital cameras, different types electronic sensors (thermal, pressure and temperature sensors, etc.), fingerprint biometric readers, etc.
- All of these electronic systems consists of different types of hardware, which are designed as per their usages and specifications.
- **Different areas where these electronic systems are being used:** (a) Character recognition (detection of number plates at tolls), (b) Biometric fingerprinting, (c) facial biometric systems, (d) advance medical imaginary (detection of critical diseases from medical images), (e) robotics vision, (f) advance military usages, (g) automation in car driving, etc. [1], [2].

Intellectual Property cores (IP cores):

- Chips, Integrated circuits, and other designs owned by a company, designer, or manufacturer used in the electronic systems.
- Processors, Co- Processors (Digital signal processors) and other Consumer Electronics (CE) hardware.
- **Importance of digital signal processors (DSP):** These co-processors performs various data-intensive and power-hungry applications involving massive computations like data compression-decompression, digital data filtering, and different complex mathematical calculations.
- The digital signal processors are design as a dedicated reusable intellectual property (IP) core using High Level Synthesis (HLS) framework.
- The digital signal processing (DSP) intellectual property (IP) cores are the integral part of consumer electronic systems, used to facilitate **applications such as image, audio and video processing etc.** with higher efficacy and low cost [3],[4].
- **Examples of DSP IP cores and their usages:** The DSP IP cores such as JPEG, MPEG, DCT and digital filters like FIR, IIR are widely used in several electronic gadgets such as digital camera, cellular phones, smart watches etc.

Abstraction levels in IP core(H/W) design:

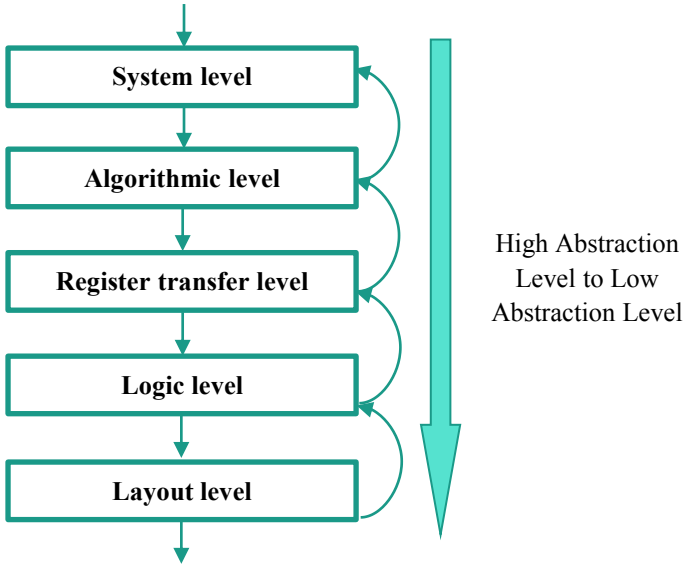


Fig. 1. Top to bottom representation of different abstraction level used in digital ICs design process

Abstraction levels:

1. System level

- Represent the design at the highest level of abstraction,
- Design (or application) is in the form of system specifications/input-output,
- At this level, functionality, space, speed and power requirement are considered.

2. Algorithmic level

- Design description in terms of behavior ,
- Control data flow graph is a popular intermediate representation of the design at the this level,
- Also known as electronic system level (ESL) or behavioural level.

3. Register transfer level

- Interconnection between different units such as arithmetic and logic unit (ALU), control unit, storage hardware.

4. Logic level

- Represents the design in terms of logic gates.

5. Layout level

- Physical/Layout representation of the design.

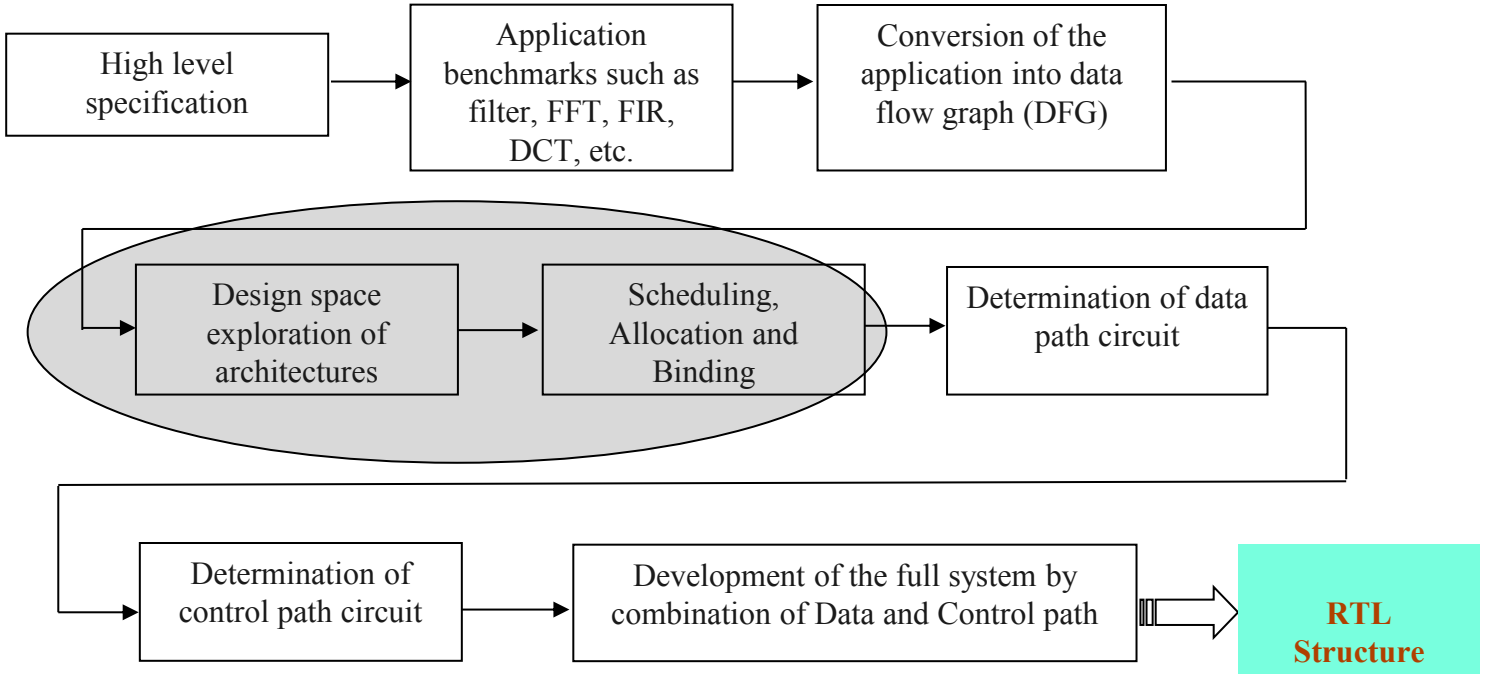
High Level Synthesis (HLS) and its importance:

- HLS framework accepts functional description of the design as input in the form of transfer function of respective computing algorithm and yields register transfer level (RTL) datapath of corresponding design including IP vendor specified design constraints [5].

❖ **Importance of HLS:**

- Shorter design cycle. Reduces the design cycle due to automation of design process.
- Easy error handling.
- Ability to search the design space (optimal resource constraints).
- Decisions made at higher levels has a great impact on lower levels.

High Level Synthesis procedure:



Different steps of HLS:

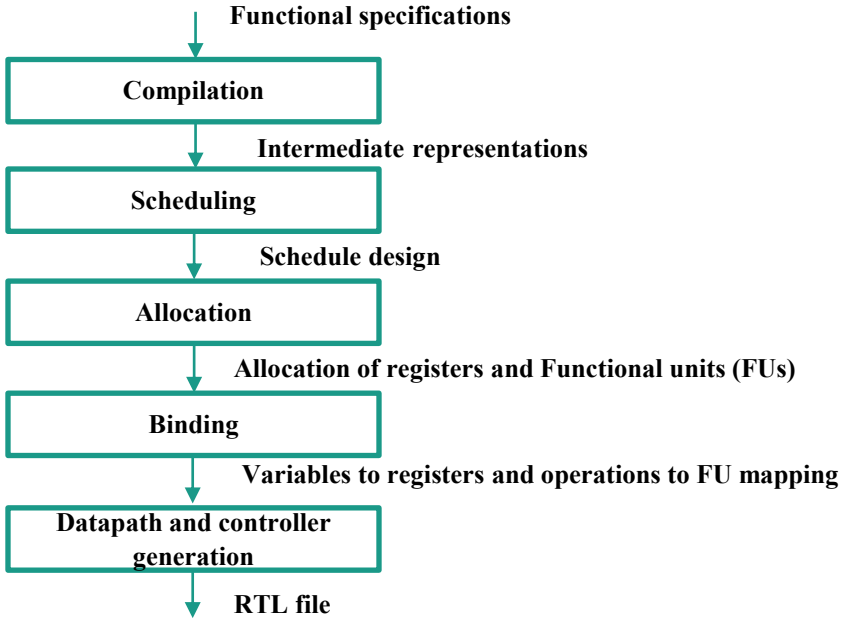


Fig. 2. Top to bottom representation of different steps of HLS

Scheduling, Allocation, and Binding:

1. Scheduling:

Scheduling involves assigning operations of the control data flow graph (CDFG) to control steps. A control step usually corresponds to a cycle of the system clock, the basic time unit of a synchronous digital systems. Some examples of scheduling algorithms are (a) As Soon As Possible (ASAP), (b) As Late As Possible (ALAP), (c) List scheduling, etc.

2. Allocation:

Here, allocation of storage variables and functional units (such as adders, multipliers, etc.) is performed.

3. Binding:

After the functional operations and storage operations are scheduled and components from the design library are selected for such operations (allocation), then comes the role of binding. Binding assigns operations to functional units, variables to storage variables and data transfers to wires or buses such that data cab be correctly computed and passed, according to the scheduling [6].

Finally, an RTL file, containing datapath and controller is generated.

Design space exploration:

- It is the process of generating optimal architectural solution (resource constraints) corresponding to a particular DSP IP core, considering IP vendor specified high level specifications such as power, area, energy, latency, etc. among different possible solutions in the design search space.
- The main objective of the design search exploration (DSE) is to perform optimization after considering different high level specifications.
- There are various heuristic and meta-heuristic techniques available to perform DSE such as (a) DSE using particle swarm optimization [7], DSE using genetic algorithm [8], DSE using bacterial foraging [9], etc.

Design process of IC/ IP core:

- Due to globalization of design supply chain, the designing of reusable IP cores or ICs involved multiple third party IP (3-PIP) vendors.
- Some countries can afford cheaper raw materials, while some have lower labor costs and a skilled workforce. The dependency on the various 3-PIP vendors is due to the time-to-market competition with rival companies. Maintaining a trade off between mass production and a lesser price is also necessary.
- The involvement of multiple 3-PIP vendors in the design process of an IC/ IP core make it prone to various **hardware threats** [10].

Fig. 3. IP/ IC design process

Security issues associated with IP Cores [11]:

IP Piracy

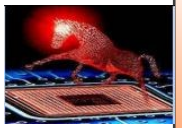
IP Counterfeiting: Selling different product under same brand name. Here, the counterfeited design (containing malicious logic) is sold under same brand name.

IP Cloning: Selling the cloned copy of the IP core under with different brand name.

Presence of malicious logic (Hardware Trojans) : Counterfeited IPs are not rigorously tested as genuine ones. Therefore, it may contain malicious logic which can cause safety hazard for both IP vendor and end consumer (such as leaking sensitive information, incorrect functional computation (wrong diagnosis of disease in case of medical imaging, loss of esteem for IP vendor, etc.).

Fraudulent claim of IP ownership: An adversary tries to fraudulently claim the ownership of the IP.

Security issues associated with IP Cores (Contd.) [12]:



Reverse Engineering (RE) attack : By means of reverse engineering (RE) during the design process, an adversary may attempt to analyze and identify the detailed interconnectivity of the register transfer level (RTL)/ gate structure to covertly insert hardware Trojans at safe places such that it goes undetected during testing process or seamlessly counterfeits/ clones the original design.

Overproduction: Exceeding the specified licensing limit (illegally) of manufactured IPs.

Therefore, it is essential to secure these image processing filter IP cores from these hardware threats.

Possible hardware threats and attacks in the design flow of hardware IC:

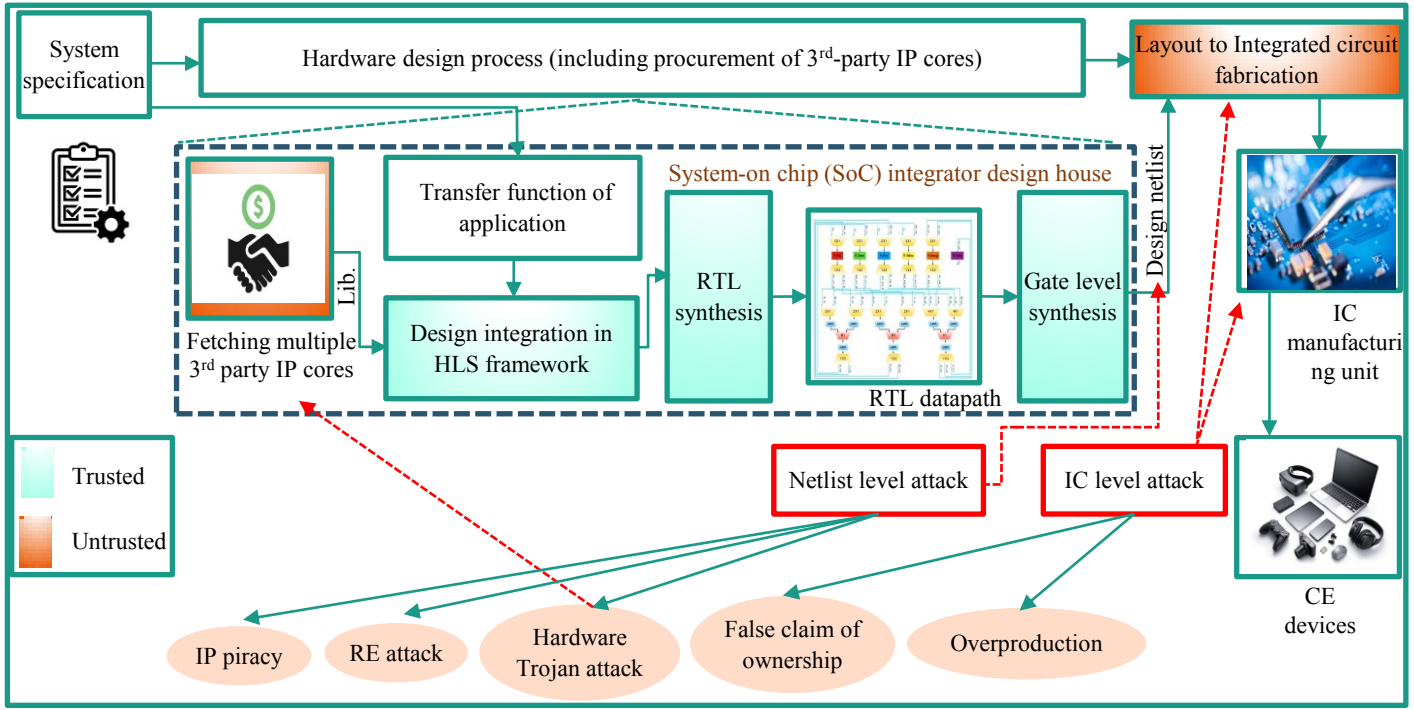
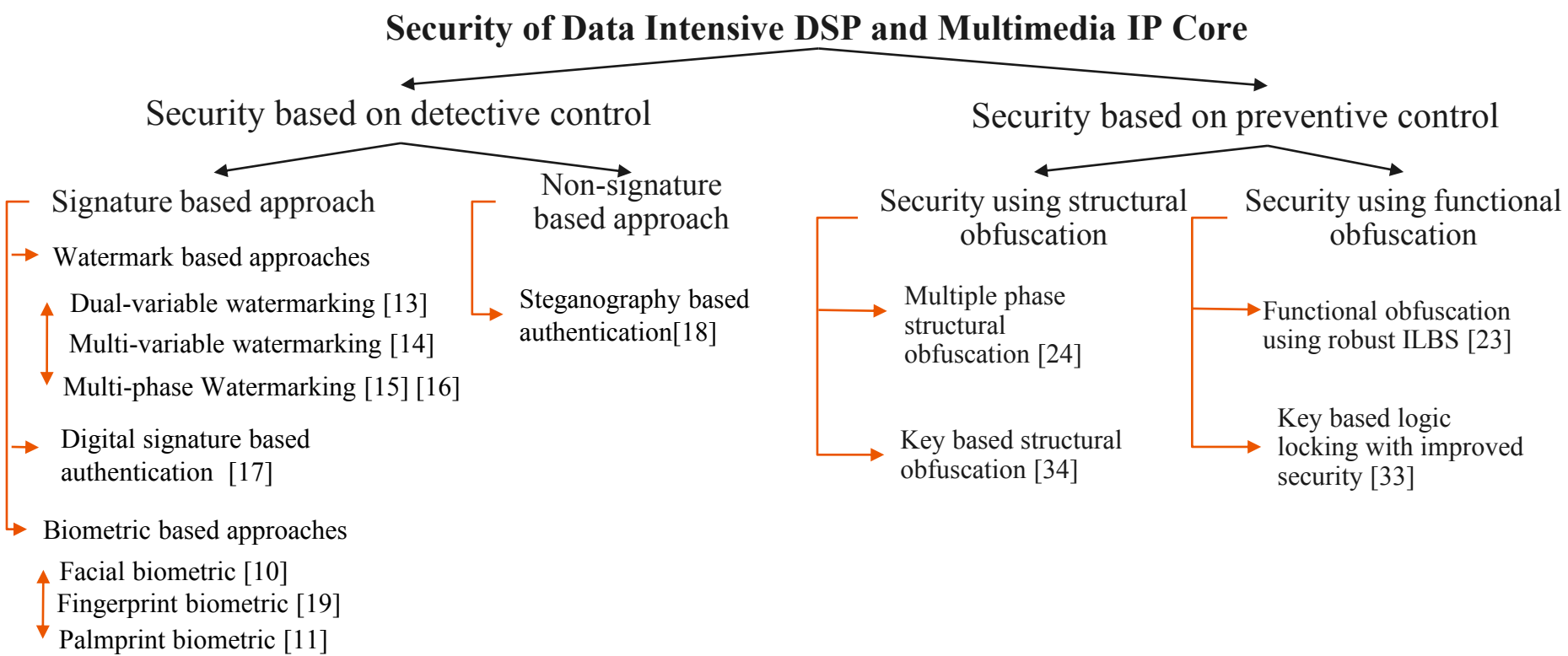


Fig. 4. Possible hardware threats and attacks in the design flow of hardware IC

DSP IP security classification tree:



Security need:

- **Protection against threat of IP ownership** is to authenticate genuine IP vendor/ designer in case of soc integrator falsely claiming the ownership of the IP core.

- **Protection against IP piracy** is to authenticate SoC integrator/ user from dishonest IP vendor selling extra copies of IPs and blaming the user.

- **Trojan** can be inserted at any stage of IP design and is not easily detectable during testing phase of IP design or remains dormant until the happening of some specific triggering/ timing event.

- **Counterfeited IPs** may cause leakage of credential information/ passwords, drowning energy resources, excessive heat dissipation of the IC components, and abnormal functioning or denial of service of the underlying computing device.

➢ Therefore, detective and preventive control of IP core from the SoC integrator's perspective must be mandatory.

Evaluation parameters :

- **Evaluation of Robustness Using Probability of Coincidence (Pc):**

$$P_c = \left(1 - \frac{1}{c}\right)^f$$

'c' denotes the number of registers used in the CIG and 'f' denotes the number of hardware constraints added.

A lower probability of coincidence indicates greater uniqueness (stronger digital evidence) in the generated signature, which helps in easier identification of counterfeited ones during the counterfeit detection process as well as robust proof against fraud IP ownership claims.

- **Tamper tolerance (TT):**

$$TT = (\mathrm{w})^f$$

'w' is the number of types of digits in the signature and 'f' is the signature size (or the number of corresponding hardware security constraints)

The higher the TT, the more robust is the proposed approach against tampering attack. Therefore, tampering an IP core design for evading the counterfeit detection process becomes complex, with a higher TT value for an adversary.

Evaluation parameters (Contd.):

- **Design cost:**

$$\text{Cost} = t1 * \frac{\text{Area}}{\text{Max area}} + t2 * \frac{\text{Latency}}{\text{Maximum latency}}$$

Where ‘area’ and ‘latency’ represents the total area and latency (delay) of the proposed methodology-based secured IP core design; ‘max area and max latency’ depict the maximum area and latency of the proposed secured design of IP core using maximum resource constraints possible. ‘t1 and t2’ are the weighing factors (weightage given to are and delay), which in the proposed approach is 0.5 each.

There are various high level specifications such area, latency, power, energy, etc. for designing an optimized DSP IP core. The IP vendor/ designer selects the parameter while designing.

Part 1 (Detective Control)

Exploring Low Cost Optimal Watermark for Reusable IP Cores During High Level Synthesis [14]:

- A novel approach based on watermarking technique has been used for protection of complex reusable IP Cores used in CE systems.
- **Watermarking:** The covert insertion of some secret information into the hardware IP core design is termed hardware watermarking.
- The proposed approch is signature-based and capable of generating hardware security constraints for securing a DSP Kernel application.
- It makes use of the IP vendor selected (defined) watermarking signature variables and their corresponding encoding rules to generate hardware security constraints.
- The generated hardware security constraints are then embedded in the IP Cores design (single phase) to authenticate genuine IP Maker.
- The watermark signature is encrypted using RSA mechanism before sending it to the genuine buyer (extra layer of defense).

[14]. 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.

Watermarking properties:

- A watermark signature must follow the following attributes:

1. It has capability to be wmbedded into an artifact (e.g. text, image, video, audio etc.) or into an IP design (hardware, software, algorithm etc.).
2. It provides an easier identification of owner/ creator of the design.
3. It should be embedded into such a way that it is difficult to remove and robust in nature.
4. It should show resiliency against Ghost signature attack and Tampering.
5. Embedding of the watermark should not consume a longer runtime. It becomes importatnt when there is a need to find an optimal architectural solution between various candidate solutions in a design space.
6. It should provide an easier/ smoother signature (watermark) detection at genuine receiver's end.

Design space exploration using PSO [14]:

- The integration of the PSO-DSE block with the watermarking methodology serves the objective of determining an optimized architectural solution.
- PSO prunes the design search space based on IP vendor specified high level specification such as area, delay, energy, power, etc. corresponding to secured DSP design to generate an optimized low-cost design.

Advantage of PSO-DSE over others such as genetic algorithm [8] and bacterial foraging [9] based DSE:

- PSO-DSE considers the magnitude of the previously computed velocity with the help of a parameter called inertia weight, while [8] and [9] do not consider the momentum of prior iterations, which increases the probability of getting stuck in the local minima during architecture exploration.
- PSO-DSE creates a balance between exploitation and exploration time with the help of linearly decreasing the value of inertia from 0.9 to 0.1. The algorithm takes more significant steps at the beginning and smaller steps on reaching higher fitness solutions, which is missing in [8]and [9]. This also enhances the chance of reaching global optimal solution.
- The inclusion of various other factors (hyperparameters), such as social and cognitive factors in PSO-DSE, helps achieve higher fitness solution within a very low exploration time

Inputs and basic steps [14]:

1. Transfer function of the target DSP application or CDFG,
2. User constraints (area and latency constraints),
3. IP vendor selected combination of multi-variable signature digits and their encoding rules,
4. module library (containing relevant information such as area, delay, etc. corresponding to DSP application),
5. PSO parameters: the maximum number of iterations, and
6. Control parameters (such as acceleration coefficients, swarm size, inertia weight, and iterations).

- First, a scheduled data flow graph (SDFG) is generated with the help of input CDFG of the target DSP application and resource constraints (here generated by PSO).
- Next, a register allocation table (RAT) is generated and a colored interval graph (CIG) is generated corresponding to it. CIG is used to embed the generated hardware security constraints.

Watermarking signature variables and their corresponding encoding mechanism [14]:

- The IP vendor selected four different signature digits and their corresponding encoding rules are as follows::

1. **'i'**= embed an artificial edge between <prime, prime> node pairs (storage variables) in colored interval graph (CIG) of DSP application,
2. **'I'**= embed an artificial edge between <even, even> node pairs (storage variables) in CIG of DSP application,
3. **'T'**= embed an artificial edge between <odd, even> node pairs (storage variables) in CIG of DSP application, and
4. **'!'** = embed an artificial edge between <0, any integer> node pairs (storage variables) in CIG of DSP application

Flow-chart of Low-Cost Watermarking-based security approach [14] :

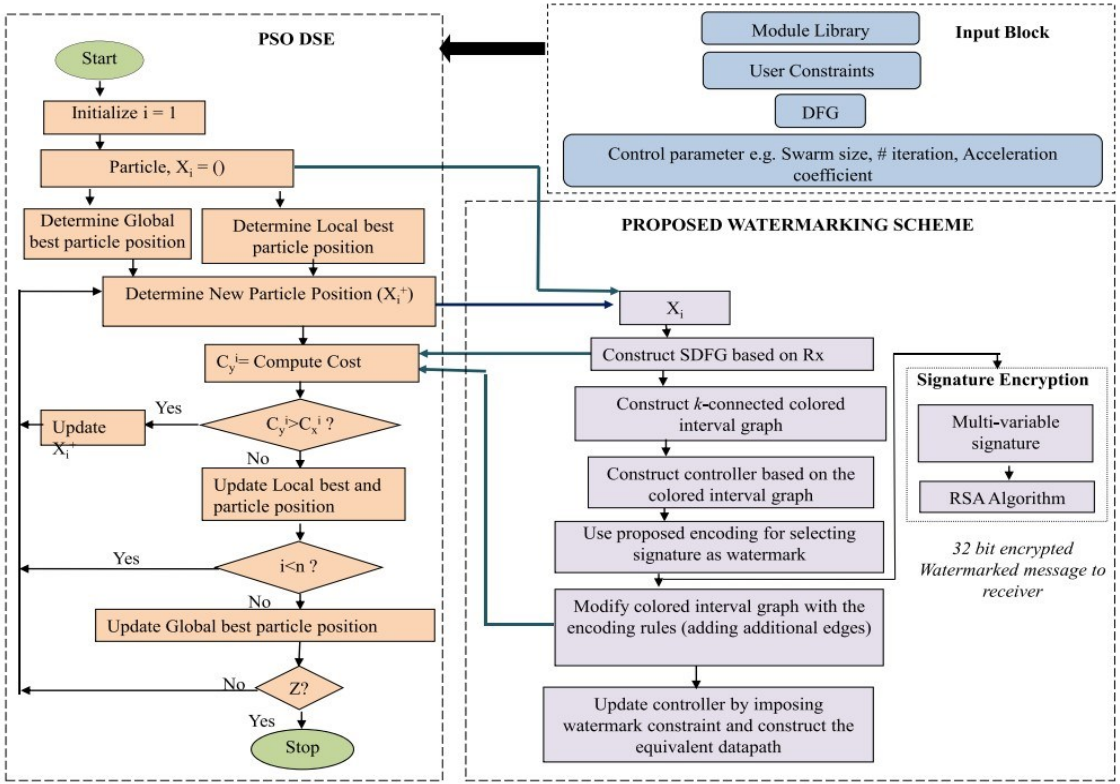


Fig. 5. Flow-chart of low-cost watermarking based approach [14]

Demonstration using a sample DFG [14] :

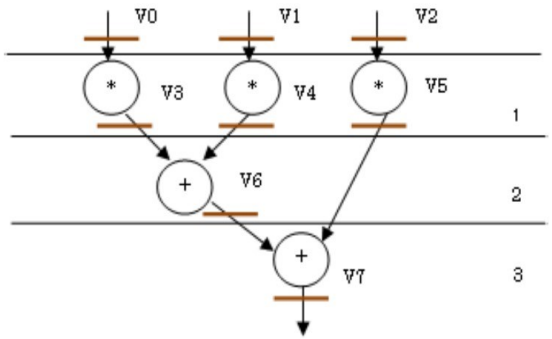


Fig. 6. Scheduling of a sample DFG with 3 multipliers and 1 adder.

Control Step (c.s)	Red (R)	Blue (B)	Green (G)
0	v0	v1	v2
1	v3	v5	v4
2	v6	v5	–
3	v7	–	–

Table 1. Controller for register allocation before embedding watermark.

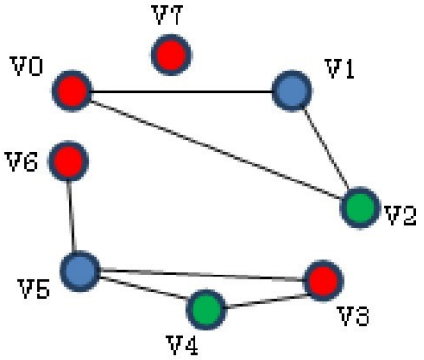


Fig. 7. Colored interval graph for the scheduling.

Flow-chart of Low-Cost Watermarking-based security approach [14] :

Desired signature (6-digit)	*i*	*i*	*I*	*!*	*T*	*I*
Corresponding additional edges to add in the colored interval graph	(2,3)	(2,5)	(2,4)	(0,1)	(1,2)	(2,6)

Table 2. Signature and its decoded meaning.

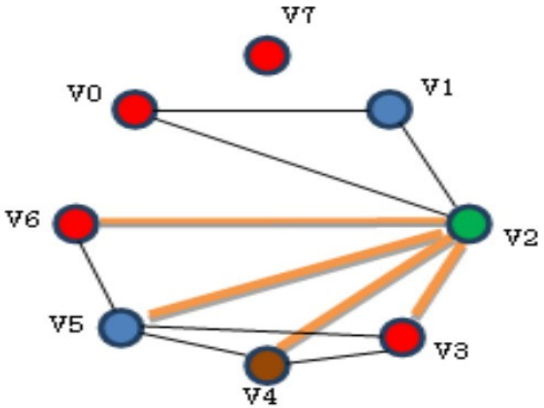


Fig. 8. Colored Interval Graph with additional edges (watermarking constraints) colored in orange.

Control Step (c.s)	Red (R)	Blue (B)	Green (G)	Maroon (M)
0	v0	v1	v2	–
1	v3	v5	–	v4
2	v6	v5	–	–
3	v7	–	–	–

Table 3. Modified controller after embedding watermark (includes authors' hidden signature).

Results compared to [13]:

Benchmarks	Proposed Watermark Solution		Watermarked Solution for Related		Cost of watermarked Solution	
	FU's	Registers	FU's	Registers	Proposed	Related
DWT	1(+), 3(*)	6	2(+), 3(*)	5	-0.064	-0.034
ARF	2(+), 4(*)	8	4(+), 2(*)	8	-0.210	0.022
MPEG	2(+), 5(*)	14	3(+), 7(*)	14	-0.448	-0.368
JPEG IDCT	5(+), 5(*)	30	12(+),12(*)	30	-0.311	-0.208
IDCT	4(+), 2(*)	8	4(+), 2(*)	8	-0.062	-0.041
MESA INTERPOLATE	3(+), 8(*)	48	9(+), 16(*)	48	-0.493	-0.382
IIR BUTTERWORTH	1(+), 3(*)	5	1(+), 3(*)	6	-0.352	-0.345
FIR	4(+), 4(*)	8	4(+), 4(*)	9	-0.257	-0.250
FFT	2(+), 2(*)	8	4(+), 2(*)	8	-0.139	-0.074

Table 4. Comparison of the proposed watermarking approach with [13] for # of watermark constraint=15.

Results (Contd.):

Benchmarks	# of variables used in signature encoding		Watermark Creation Time (ms)		# of watermarking constraints	
	Ours (I, i, T, !)	Related (0, 1)	Ours	Related	Ours	Related
DWT	4	2	1	10	15	15
ARF			1	17		
MPEG			5	14		
JPEG IDCT			6	61		
IDCT			3	14		
MESA INTERPOLATE			8	101		
IIR BUTTERWORTH			1	16		
FIR			2	31		
FFT			1	49		

Table 5. Comparison of the proposed watermarking approach with [13].

Triple-Phase Watermarking for Reusable IP Core Protection during Architecture Synthesis [15]:

- A novel approach based on Watermarking technique applied in three different phases of IP core design process has been used for protection of complex reusable IP cores.
- The proposed approch is signature-based and capable of generating hardware security constraints for securing a DSP Kernel application.
- It makes use of the IP vendor selected (defined) watermarking signature variables and their corresponding encoding rules to generate hardware security constraints.
- The generated hardware security constraints are then embedded in the IP Cores design (three different phases) to authenticate genuine IP Maker.
- The involvement of a 7-digit multi-variable signature and different IP vendor-selected encoding mechanisms for different phases of watermarking makes the proposed approach highly robust [15].

[15]. A. Sengupta, D. Roy and S. P. Mohanty, "Triple-Phase Watermarking for Reusable IP Core Protection During Architecture Synthesis," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 4, pp. 742-755, April 2018.

Inputs and basic steps [15]:

Inputs:

1. Transfer function of the target DSP application or CDFG,
2. Resource constraints,
3. IP vendor selected combination of multi-variable signature digits, and
4. IP vendor selected encoding rules.

- First, a scheduled data flow graph (SDFG) is generated with the help of input CDFG of the target DSP application and resource constraints.
- Subsequently, all the available functional units (FU), such as adders and multipliers, are allocated based on IP vendor-provided hardware type. A FU allocation table is generated for all operations with a non-critical operations timing table.
- **Non-critical operations** in an SDFG are operations that can be shifted in upward or downward control steps without violating data dependency and functionality.
- Next, all operations are sorted in increasing order based on their numbers in each CS.

Watermarking signature variables and their corresponding encoding mechanism [15]:

- The IP vendor selected seven different signature digits and their corresponding encoding rules are as follows::

1. 'α'= operations with odd numbers are allocated to hardware of vendor type 1 and operations with even numbers are allocated to hardware of vendor type 2 in odd control step (CS),
2. 'β'= operations with odd numbers are allocated to hardware of vendor type 2 and operations with even numbers are allocated to hardware of vendor type 1 in even control step,
3. 'γ'= a non-critical path operation with highest mobility is moved to immediate next control step,
4. 'i'= embed an artificial edge between <prime, prime> node pairs (storage variables) in colored interval graph (CIG) of DSP application,
5. 'I'= embed an artificial edge between <even, even> node pairs (storage variables) in CIG of DSP application,
6. 'T'= embed an artificial edge between <odd, even> node pairs (storage variables) in CIG of DSP application, and
7. '!' = embed an artificial edge between <0, any integer> node pairs (storage variables) in CIG of DSP application.

Flow-chart of Triple phase watermarking [15] :

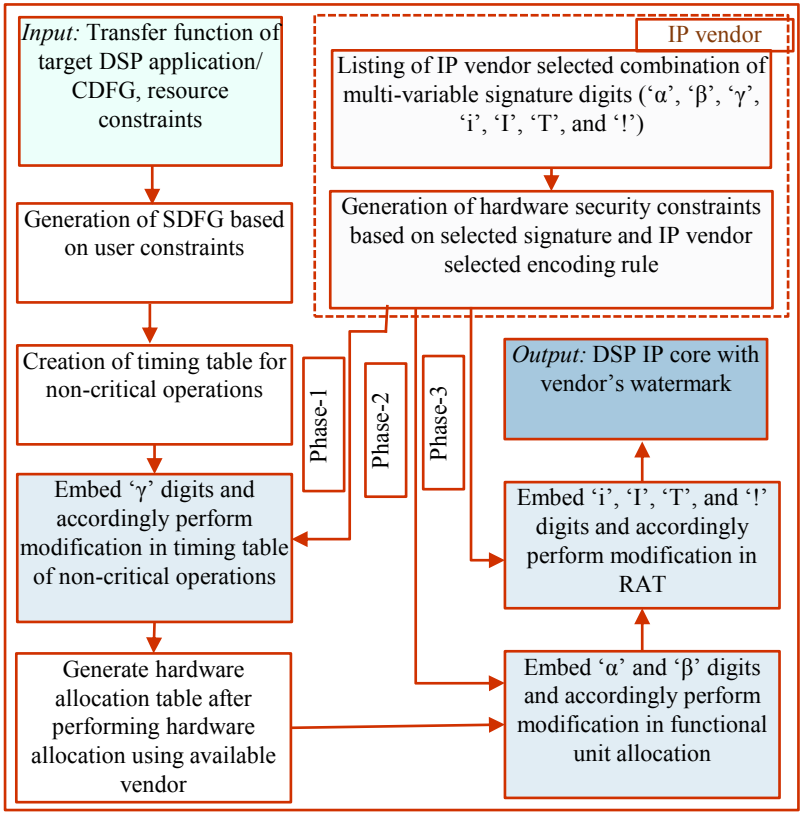


Fig. 9. Flow-chart of triple phase watermarking mechanism [15]

Steps used in triple phase watermarking [15]:

Phase 1 (Scheduling phase):

- In the first phase of watermarking, non-critical operations (starting from CS 1) are moved to the immediate next CS for each occurrence of 'γ' (shifting must not violate the data dependency and hardware constraints), and a modified timing table for non-critical operations is generated. A hardware allocation table is generated corresponding to different used functional units (hardware)..

Phase 2 (Hardware allocation phase):

- In the second watermarking phase, **FUs are re-allocated according to the IP vendor selected encoding rules' α' and 'β', and a modified hardware allocation table is generated**. After this, allocation of storage variables in the SDFG (double phased watermarked) is performed, and a CIG is created to find the minimum number of required registers for storage variables. Next, a register allocation table (RAT) is created from SDFG (assigned with storage variables).

Phase 3 (Register allocation phase):

- Then, the additional artificial edges (security constraints) are determined based on the IP vendor's selected 'i', 'I', 'T', and '!' digits. Further, these determined security constraints are embedded into the CIG of the design, followed by local alteration in register allocation if two adjacent register's colors are the same.

Evaluation parameter :

- **Probability of Coincidence (Pc):**

$$\mathrm{Pc} = \left(1 - \frac{1}{z \times \prod_{i=1}^{h} N(R_i)}\right)^{t}$$

Where ‘t’ denotes number digit used for performing watermarking, ‘z’ denotes total number registers (colors) used in the register allocation phase, ‘h’ represents the different types of hardware, and “N(Ri)” indicates the number of hardware of type ‘i’.

A lower probability of coincidence indicates greater uniqueness (stronger digital evidence) in the generated signature, which helps in easier identification of counterfeited ones during the counterfeit detection process as well as robust proof against fraud IP ownership claims.

Results compared to [13], [14]:

Benchmarks [13], [14]	# of register before watermark	P_c			# of times lower P_c of proposed approach compared to [13], [14]
		Proposed	[13]	[14]	
ARF	8	$3.3x10^{-27}$	$2.2x10^{-5}$	$2.2x10^{-5}$	$6.9x10^{21}$
DCT	8	$3.7x10^{-21}$	$2.2x10^{-5}$	$2.2x10^{-5}$	$6.1x10^{15}$
DWT	5	$8.3x10^{-35}$	$1.7x10^{-8}$	$1.7x10^{-8}$	$2.1x10^{26}$
EWF	4	$6.8x10^{-39}$	$1.0x10^{-10}$	$1.0x10^{-10}$	$1.5x10^{28}$
IDCT	8	$3.3x10^{-27}$	$2.2x10^{-5}$	$2.2x10^{-5}$	$6.9x10^{21}$
MPEG MV	14	$3.8x10^{-31}$	$2.6x10^{-3}$	$2.6x10^{-3}$	$6.9x10^{27}$
JPEG IDCT	12	$1.9x10^{-23}$	$9.4x10^{-4}$	$9.4x10^{-4}$	$5.0x10^{19}$

Table 6. Comparison of strength of watermark indicated through probability of coincidence (as proof of authorship) between proposed [13] and [14] for signature size (80digits).

Results (Contd.):

Signature Size (digits)	# of possible signature combination			# of times higher tamper-tolerance of proposed approach compared to [13], [14]	
	Proposed	[13]	[14]	[13]	[14]
15	$4.8*10^{12}$	32768	$10.7*10^{8}$	$14.5*10^{7}$	4421
30	$2.3*10^{25}$	$1.1*10^{9}$	$1.2*10^{18}$	$2.1*10^{16}$	$19.5*10^{6}$
45	$1.1*10^{38}$	$3.5*10^{13}$	$1.2*10^{27}$	$3.0*10^{24}$	$8.6*10^{10}$
60	$5.1*10^{50}$	$1.2*10^{18}$	$1.3*10^{36}$	$4.4*10^{32}$	$3.8*10^{14}$
80	$4.1*10^{67}$	$1.2*10^{24}$	$1.5*10^{48}$	$3.4*10^{43}$	$2.8*10^{19}$

Table 7. Comparison of tamper tolerance between proposed, [13] and [14] for different signature strength.

Quadruple phase watermarking during high level synthesis for securing reusable hardware intellectual property cores [16]:

- A Novel approach based on Quadruple phase Watermarking technique has been used for protection of complex reusable IP Cores.
- The proposed approch is signature-based and capable of generating hardware security constraints for securing a DSP Kernel application.
- It makes use of the IP vendor selected (defined) watermarking signature variables and their corresponding encoding rules to generate hardware security constraints.
- The generated hardware security constraints are then embedded in the IP Cores design (four different phases) to authenticate genuine IP Maker.
- The four different phases are **(a) scheduling, (b) register binding, (c) resource binding and (d) interconnect binding**, respectively. Further, The proposed approach introduces several novelties: **graph partitioning, encoding tree, and eightfold mapping** to generate a robust watermarking signature.

[16]. Mahendra Rathor, Aditya Anshul, K Bharath, Rahul Chaurasia, Anirban Sengupta, Quadruple phase watermarking during high level synthesis for securing reusable hardware intellectual property cores, *Computers and Electrical Engineering*, Volume 105, 2023, 108476,ISSN 0045-7906.

Flow-chart of Quadruple phase watermarking [16]:

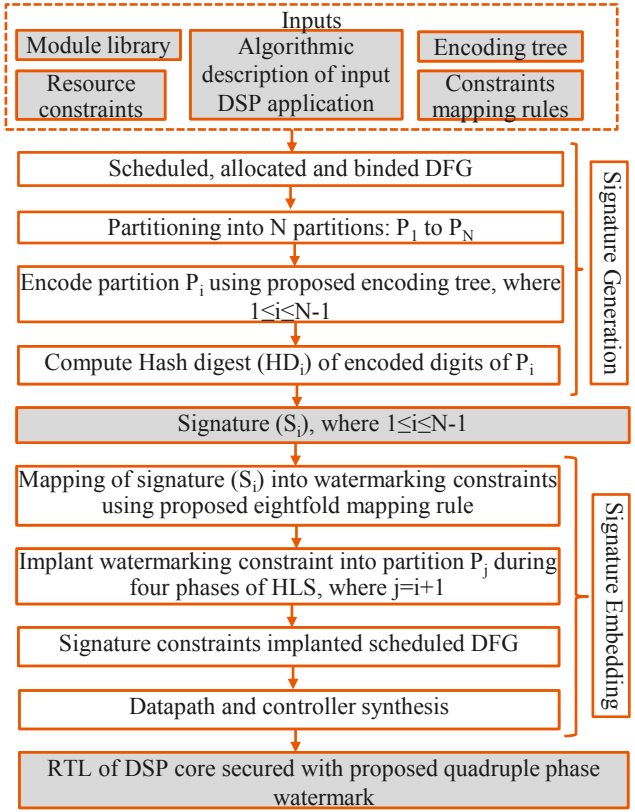


Fig. 10. Flow-chart of quadruple phase watermarking mechanism [16]

Step 1 (Graph partitioning):

- After scheduling, allocation and binding graph portioning is performed. A graph portioning is performing in following ways:

a) A small possible partition should contain at least two connected nodes of the graph to enable more meaningful encoding and embedding of constraints,

b) There should be at least two partitions of the graph for the applicability of the proposed approach,

c) The first partition P1 should be the smallest partition as the constraints are not embedded in this partition but it is used to derive signature for the subsequent partition,

d) The number of partitions should vary as per the size of the target application (in terms of number of operations) for effective watermarking.

- The proposed SDFG partitioning plays a significant role in enhancing the strength of the signature. Owing to partitioning, intricacy of deducing the exact signature is increased manifold for an attacker. This is because; the attacker needs to know the location of the partition and number of partitions of the SDFG along with the knowledge of partition encoding.

Graph partioning example (using 8-point DCT) [16]:

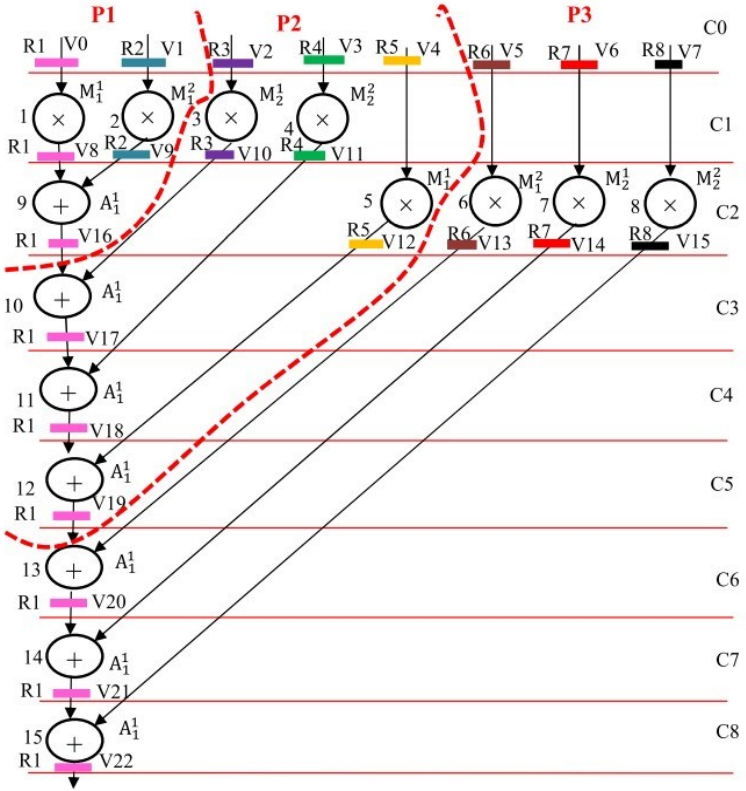


Fig. 11. Scheduled DFG of 8-point DCT core with partitions P1, P2 and P3. (note: dashed curves in red indicate the cuts applied for partitioning, different colored bars indicates registers to store primary and intermediate data or storage variables).

Encoding Tree [16]:

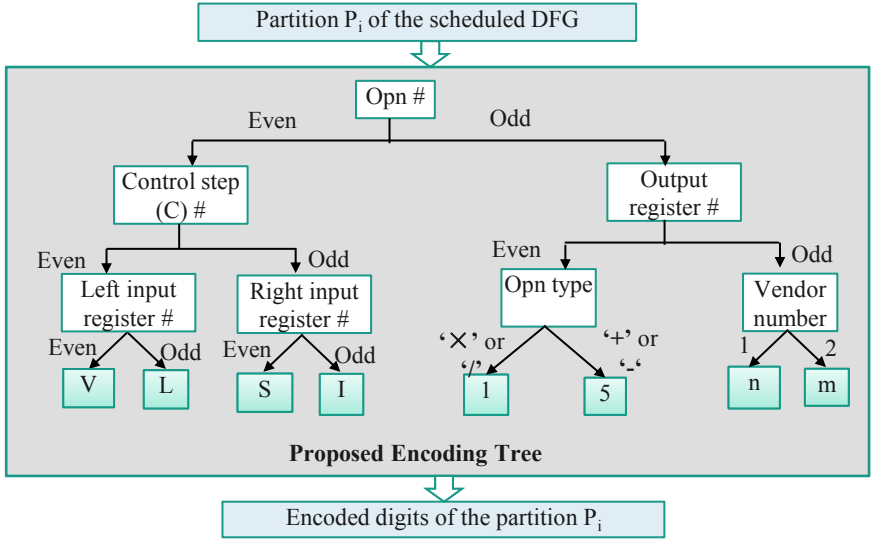


Fig. 12. Proposed encoding tree used to encode partitions of SDFG.

- The leaves of the encoding tree are the designer chosen alphanumeric digits. In order to generate encoded digits from a partition, each opn# is traversed through the encoding tree.

Encoding bitstream generation for each partition [16]:

- Let's see the encoding of partition-P1of SDFG (shown in Fig. 11). The partition-P1 is encoded into alphanumeric digits using proposed encoding tree. For example, opn #1 has odd parity, odd O/P register# (R1) and is assigned to vendor number-1. Hence it is encoded into 'n' through traversal of the encoding tree shown in Fig. 12. Thereby, the encoding of partition-P1 (3 opns) is: "nSn".
- Hash digest of encoded digits, generated from a partition (Pi), is calculated using SHA-512. The overall signature is a concatenation of different hashes generated from the encoding of different partitions.
- The alphanumeric digits "nSn" are first transformed into 512-bit hash digest. Further, the obtained hash-bitstream is truncated to 48 (=16×3) bits based on the designer chosen size 16 of signature S1. The signature S1 (size =16 triads) is: "101- 001-101-000-100-011-100-001-111-100-010-110-001-010-010-100″.
- The truncated bitstream is represented in the form of triads.

Eightfold mapping (mapping of signature into hardware security constraints [16]:

Table 8.

Mapping triads in the signature into security constraints.

Triads	Mapping into security (watermarking) constraints
"000"	Embed an edge between (even, even) node pair in CIG
"001"	Embed an edge between (odd, odd) node pair in CIG
"010"	Embed an edge between (odd, prime) node pair in CIG
"011"	Move an operation of non-critical path with highest mobility into immediate next control step (C)
"100"	Bind vendor-1 to even opn and vendor-2 to odd opn
"101"	Bind vendor-1 to odd opn and vendor-2 to even opn
"110"	Assign odd register to the 'right' input of FU and even register to the 'left' input of FU
"111"	Assign odd register to the 'left' input of FU and even register to the 'right' input of FU

Signature generation and embedding process [16]:

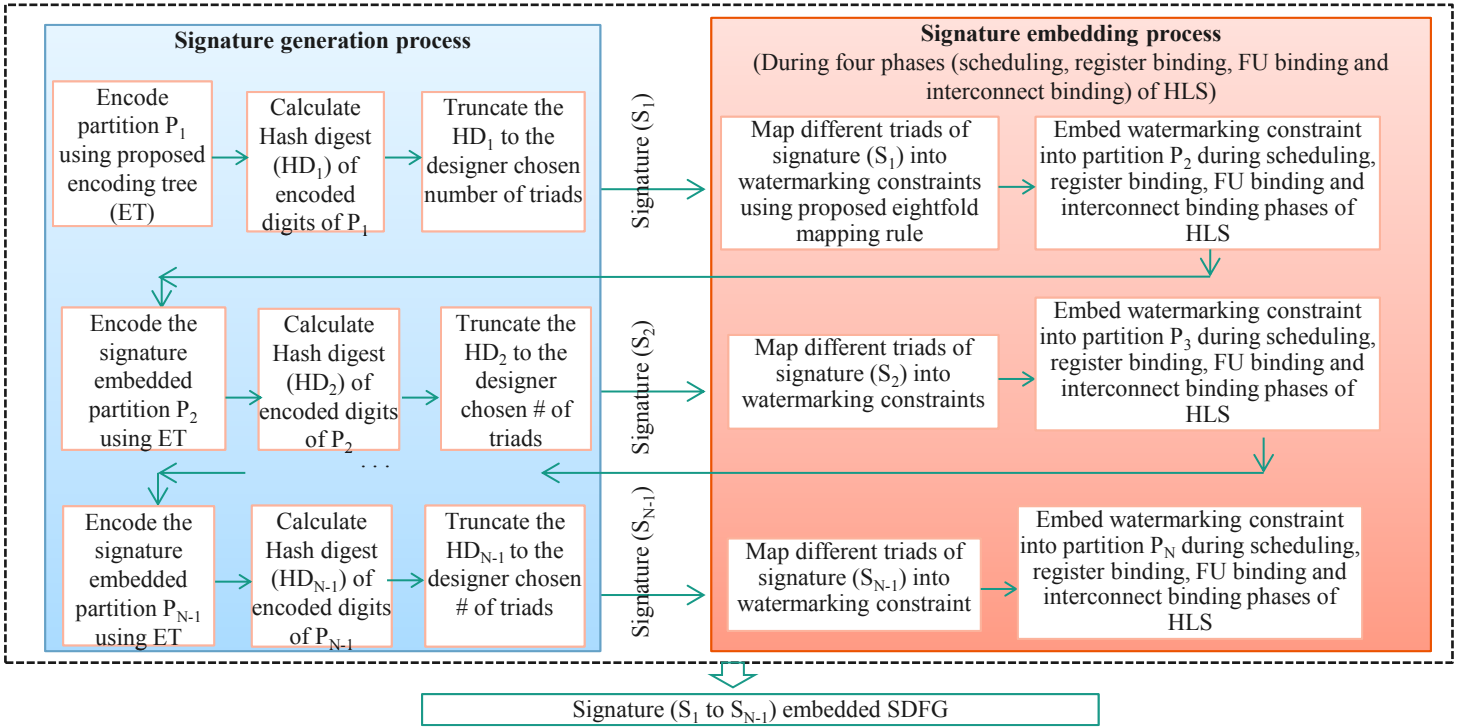


Fig. 13. Signature generation and embedding flow of proposed quadruple phase watermarking approach.

Eightfold mapping (mapping of signature into hardware security constraints [16]:

Table 9.

Watermarking constraints for embedding in partition P_2.

Embedding phase	Triads in signature	Corresponding watermarking (security) constraints based on mapping rules
Scheduling	011	Shift opn5 from C2 to C3
Register binding	001	Edge between node pair (V3, V11) in CIG
	000	Edge between node pair (V2, V4) in CIG
	001	Edge between node pair (V3, V17) in CIG
	010	Edge between node pair (V3, V19) in CIG
	001	Edge between node pair (V11, V17) in CIG
	010	Edge between node pair (V11, V19) in CIG
	010	Edge between node pair (V17, V19) in CIG
FU binding	101	bind opn3 to the FU of vendor1
	101	bind opn4 to the FU of vendor2
	100	bind opn5 to the FU of vendor2
	100	bind opn10 to the FU of vendor1
	100	bind opn11 to the FU of vendor2 (Embedding not possible)
	100	bind opn12 to the FU of vendor1
Interconnect binding	111	R3(storing variable V10) to left input of FU
	110	R4 (storing variable V11) to left input of FU

Signature generation and embedding process [16]:

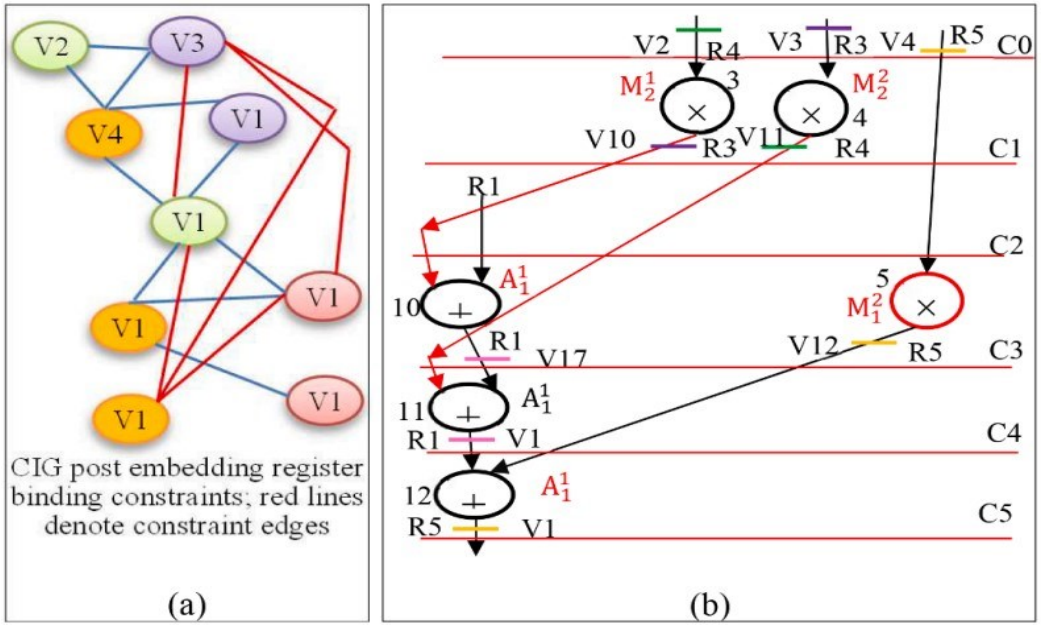


Fig. 14. (a) CIG post embedding register binding constraints (b) SDFG of partition-P2 post embedding signature S1..

Evaluation parameter :

- **Probability of Coincidence (Pc):**

$$Pc = \left(1 - \frac{1}{c}\right)^{f1} * \left(\frac{1}{\pi_{i=1}^{K} U(Zi)}\right)^{f2} * \left(\frac{1}{2}\right)^{f3} * \pi_{j=1}^{f4}\left(\frac{1}{\mu(x_j)}\right)$$

Where, first, second, third and fourth term here indicate Pc w.r.t. register binding, FU binding, interconnect binding and scheduling phases respectively. In the first term, c and f1 indicate the number of colors (registers) in the CIG pre-embedding register binding constraints and number of constraint edges respectively. In the second term, K, U(Zi) and f2 denote the number of types of FU resources, number of instances of FU type Zi and number of FU binding constraints respectively. In the third term, f3 denotes number of interconnect binding constraints. In the fourth term, f4 denotes number of scheduling constraints and $\mu(x_j)$ denote the mobility of opn 'x' which is subjected to imposing of scheduling constraint and x_j indicates the opn corresponding to j^{th} scheduling constraint.

Results compared to [13], [14], [15]:

Table 10
Probability of Coincidence (Pc) Analysis of Proposed Approach w.r.t. Related Approaches [13], [14], [15]

DSP Benchmarks	Signature size (# of triads)	Pc				#times lower Pc than [14]	#times lower Pc than [15]	#times lower Pc than [13]
		Proposed	[14]	[15]	[13]			
DCT	20	7.6e-7	6.9e-2	1.0e-5	6.9e-2	9.0e+4	1.3e+1	9.0e+4
	25	6.4e-8	3.5e-2	1.5e-6	3.5e-2	5.4e+5	2.3e+1	5.4e+5
	30	1.7e-10	1.8e-2	2.8e-6	1.8e-2	1.0e+8	1.6e+4	1.0e+8
FFT	20	2.3e-6	2.7e-1	5.4e-5	2.7e-1	1.1e+5	2.3e+1	1.1e+5
	26	1.1e-8	1.8e-1	5.1e-7	1.8e-1	1.6e+7	4.6e+1	1.6e+7
	32	6.0e-11	1.2e-1	4.9e-9	1.2e-1	2.0e+9	8.1e+1	2.0e+9
IIR	20	3.1e-4	2.2e-1	1.9e-2	2.2e-1	7.0e+2	6.1e+1	7.0e+2
	26	1.6e-5	1.4e-1	6.5e-3	1.4e-1	8.7e+3	4.0e+2	8.7e+3
	32	1.0e-5	9.3e-2	1.2e-3	9.3e-2	9.3e+3	1.2e+2	9.3e+3
FIR	22	1.3e-9	5.3e-2	1.3e-6	5.3e-2	4.0e+7	1.0e+3	4.0e+7
	34	3.8e-13	1.0e-2	1.9e-8	1.0e-2	2.6e+10	5.0e+4	2.6e+10
	46	7.4e-20	2.1e-3	1.0e-13	2.1e-3	2.8e+16	1.3e+6	2.8e+16
ARF	22	1.0e-9	2.4e-1	2.4e-8	2.4e-1	2.4e+8	2.4e+1	2.4e+8
	34	3.0e-15	1.1e-1	4.7e-13	1.1e-1	3.6e+13	1.5e+2	3.6e+13
	46	1.5e-17	5.1e-2	2.8e-14	5.1e-2	3.4e+15	1.8e+3	3.4e+15
1D-DWT	20	9.2e-6	2.6e-2	4.2e-5	2.6e-2	2.8e+3	4.5e+0	2.8e+3
	25	5.9e-7	1.0e-2	2.7e-6	1.0e-2	1.6e+4	4.5e+0	1.6e+4
	30	3.8e-8	4.2e-3	1.7e-7	4.2e-3	1.1e+5	4.4e+0	1.1e+5
MPEG	20	3.3e-11	2.2e-1	2.1e-10	2.2e-1	6.6e+9	6.3e+0	6.6e+9
	30	1.1e-15	1.0e-1	1.3e-14	1.0e-1	9.0e+13	1.1e+1	9.0e+13
	40	4.0e-20	5.1e-2	8.7e-19	5.1e-2	1.2e+18	2.1e+1	1.2e+18

Results (Contd.):

Table 11
Tamper Tolerance Analysis of Proposed Approach w.r.t. Related Approaches [13], [14], [15]

DSP Benchmarks	Signature size (# of triads)	Tamper tolerance (T^P)				# times higher T^P than [14]	# times higher T^P than [15]	# times higher T^P than [13]
		Proposed	[14]	[15]	[13]			
8-point DCT	20	9.9e+27	1.1e+12	7.9e+16	1.0e+6	9.0e+15	1.2e+11	9.9e+21
	25	3.2e+32	1.1e+15	1.3e+21	3.3e+7	2.9e+17	2.4e+11	9.7e+24
	30	1.4e+45	1.1e+18	2.2e+25	1.1e+9	1.2e+27	6.3e+19	1.2e+36
FFT	20	1.2e+27	1.1e+12	7.9e+16	1.0e+6	1.0e+15	1.5e+10	1.2e+21
	26	1.0e+37	4.5e+15	9.3e+21	6.7e+7	2.2e+21	1.0e+15	1.4e+29
	32	2.9e+51	1.8e+19	1.1e+27	4.2e+9	1.6e+32	2.6e+24	6.9e+41
IIR	20	4.7e+21	1.1e+12	7.9e+16	1.0e+6	4.2e+9	5.9e+4	4.7e+15
	26	6.3e+29	4.5e+15	9.3e+21	6.7e+7	1.4e+14	6.7e+7	9.4e+21
	32	5.8e+48	1.8e+19	1.1e+27	4.2e+9	3.2e+29	5.2e+21	1.3e+39
FIR	22	5.4e+39	1.7e+13	3.9e+18	4.2e+6	3.1e+26	1.3e+21	1.2e+33
	34	2.4e+52	2.9e+20	5.4e+28	1.7e+10	8.2e+31	4.4e+23	1.4e+42
	46	6.7e+66	4.9e+27	7.5e+38	7.0e+13	1.3e+39	8.9e+27	9.5e+52
ARF	22	5.4e+39	1.7e+13	3.9e+18	4.2e+6	3.1e+26	1.3e+21	1.2e+33
	34	3.7e+50	2.9e+20	5.4e+28	1.7e+10	1.2e+30	6.8e+21	2.1e+40
	46	1.0e+65	4.9e+27	7.5e+38	7.0e+13	2.0e+37	1.3e+26	1.4e+51
1D-DWT	20	7.9e+28	1.1e+12	7.9e+16	1.0e+6	7.1e+16	1.0e+12	7.9e+22
	25	2.6e+33	1.1e+15	1.3e+21	3.3e+7	2.3e+18	2.0e+12	7.8e+25
	30	8.5e+37	1.1e+18	2.2e+25	1.1e+9	7.7e+19	3.8e+12	7.7e+28
MPEG	20	1.0e+37	1.1e+12	7.9e+16	1.0e+6	9.0e+24	1.2e+20	1.0e+31
	30	1.1e+46	1.1e+18	2.2e+25	1.1e+9	1.0e+28	5.0e+20	1.0e+37
	40	1.2e+55	1.2e+24	6.3e+33	1.1e+12	1.0e+31	1.9e+21	1.0e+43

IP core steganography used for protecting DSP kernels used in CE systems [18]:

- A novel approach based on steganograpgy technique has been used for protection of complex reusable IP Cores used in CE systems.
- The proposed approch is signature-free and capable of generating hardware security constraints for securing a DSP Kernel application.
- It makes use of the register allocation table of DSP kernel application itself to generate hardware security constraints.
- The generated hardware security constraints then embedded in the IP Cores degine to authenticate genuine IP Maker.
- Threshold entropy option in the approach provides more control to designer as compared to signature based approach.

[18]. A. Sengupta and M. Rathor, “IP Core Steganography for Protecting DSP Kernels Used in CE Systems,” in *IEEE Transactions on Consumer Electronics*, vol. 65, no. 4, pp. 506-515, Nov. 2019.

Flow chart of Steganography-based security approach [18]:

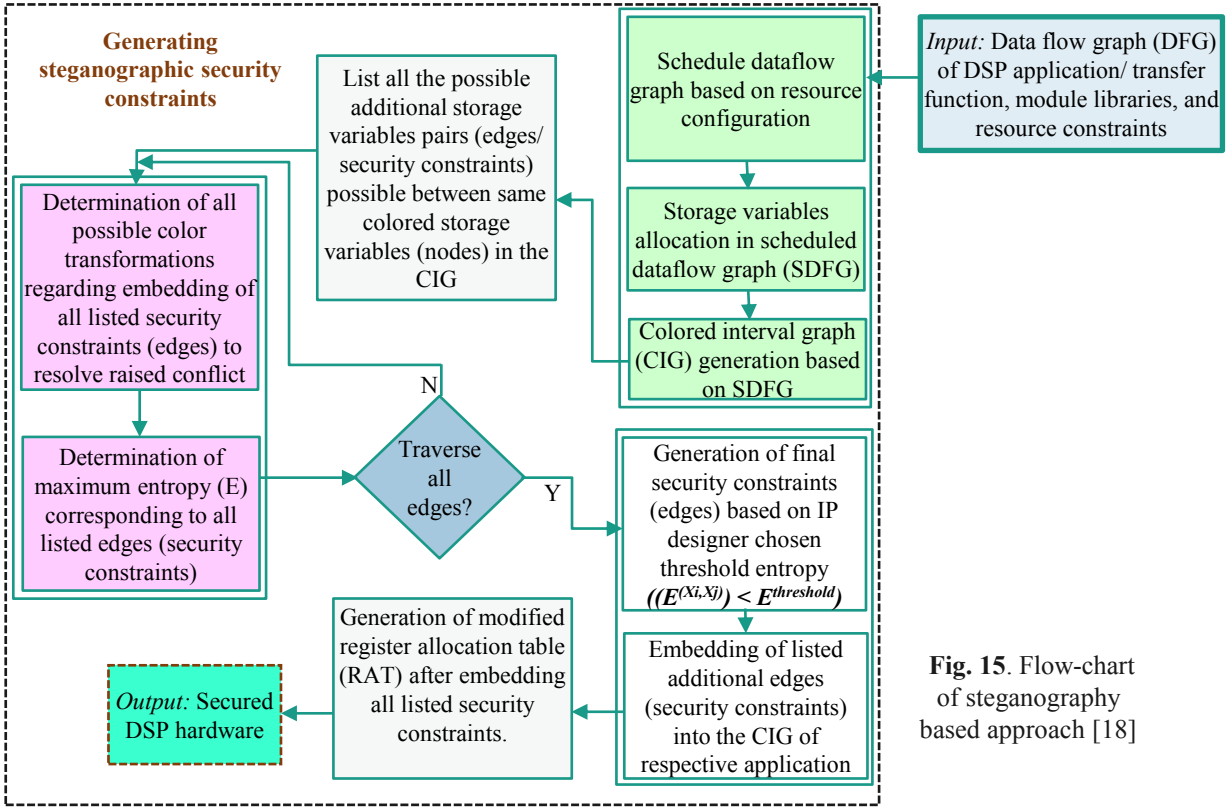


Fig. 15. Flow-chart of steganography based approach [18]

Generation of hardware security constraints from register allocation table [4]:

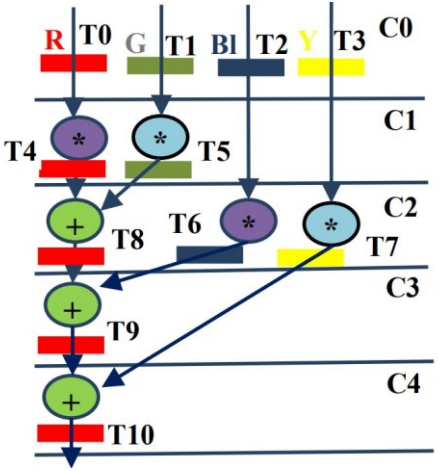


Fig. 16. Scheduled data flow graph of 4-point DCT with 1(+) and 2(*) before secret constraint embedding.

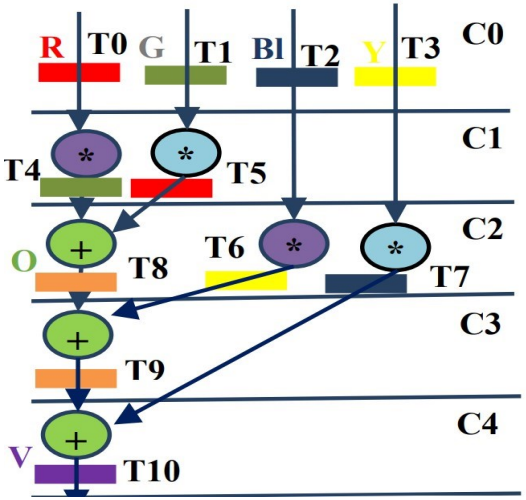


Fig. 17. Scheduled data flow graph of 4-point DCT with 1(+) and 2(*) after secret constraint embedding.

	R	G	Bl	Y
C0	T0	T1	T2	T3
C1	T4	T5	T2	T3
C2	T8	-	T6	T7
C3	T9	-	-	T7
C4	T10	-	-	-

Table 12. Register allocation table of storage variables (T0-T10) of DCT-4.

	R	G	Bl	Y	O	V
C0	T0	T1	T2	T3	-	-
C1	T5	T4	T2	T3	-	-
C2	-	-	T7	T6	T8	-
C3	-	-	-	T6	T9	-
C4	-	-	-		-	T10

Table 13. Register allocation table of storage variables (T0-T10) of DCT-4 post signature embedding.

Possible edge	Maximum entropy	Possible edge	Maximum entropy	Possible edge	Maximum entropy
<T1, T5>	2	<T0, T9>	3	<T4, T10>	3
<T2, T6>	3	<T0, T10>	3	<T8, T9>	2
<T3, T7>	3	<T4, T8>	3	<T8, T10>	3
<T0, T4>	2	<T4, T9>	3	<T9, T10>	3
<T0, T8>	3	-	-	-	-

Table 14. Additional edges (hardware security constraints) generated for DCT-4.

Results compared to [14], [20]:

Benchmark	# of registers before steganography	# of Registers required after embedding same number of constraints											
		$E^{th} = 4$				$E^{th} = 5$				$E^{th} = 6$			
		#C	P	R	% Reg^r	# C	P	R	% Reg^r	# C	P	R	% Reg^r
DCT	8	13	8	9	**11 %**	18	8	10	20%	24	8	10	**20%**
FIR	8	20	8	9	**11 %**	57	8	10	20%	57	8	10	**20%**
JPEG_IDCT	29	50	29	29	**--**	124	29	30	3.3%	203	29	30	**3.3%**
MPEG	14	21	14	15	**6.6 %**	46	14	15	6.6%	52	14	15	**6.6%**
JPEG_sample	12	18	12	13	**7.6 %**	20	12	13	7.6%	30	12	13	**7.6%**
IDCT	10	63	10	11	**9.1 %**	125	10	18	44.4%	125	10	18	**44.4%**
EWF	7	12	7	8	**12.5%**	30	7	8	12.5%	34	7	8	**12.5%**
					Avg.~ 8.25%								***Avg.~ 16%***

Table 15. Comparison of proposed approach with [14], [20] in terms of the number of registers for the same number of constraints.

Note: *#C = # of constraints added, P= proposed approach, R= related works [14], [20], %Reg^r= reg reduction % obtained w.r.t. [14], [20].*

Embedding Digital Signature Using Encrypted-Hashing for protection of DSP cores in CE [17]:

- A novel approach named multi-level encoding and encrypted-hash based digital signature for protection of complex reusable IP cores used in CE systems.
- The proposed approch is capable of encoding a DSP Kernel application.
- SHA-512 algorithm is used to generate intermediate bitstream digest.
- Digital signature is generated using RSA with the help of messege digest of encoded application.
- The generated signature is then mapped to its corresponding hardware security constraints based on a mapping rule and then implanted in IP cores degine to authenticate genuine IP Maker.

[17]. A. Sengupta, E. R. Kumar and N. P. Chandra, “Embedding Digital Signature Using Encrypted-Hashing for Protection of DSP Cores in CE,” in IEEE Transactions on Consumer Electronics, vol. 65, no. 3, pp. 398-407, Aug. 2019.

Digital-signature based security approach [17]:

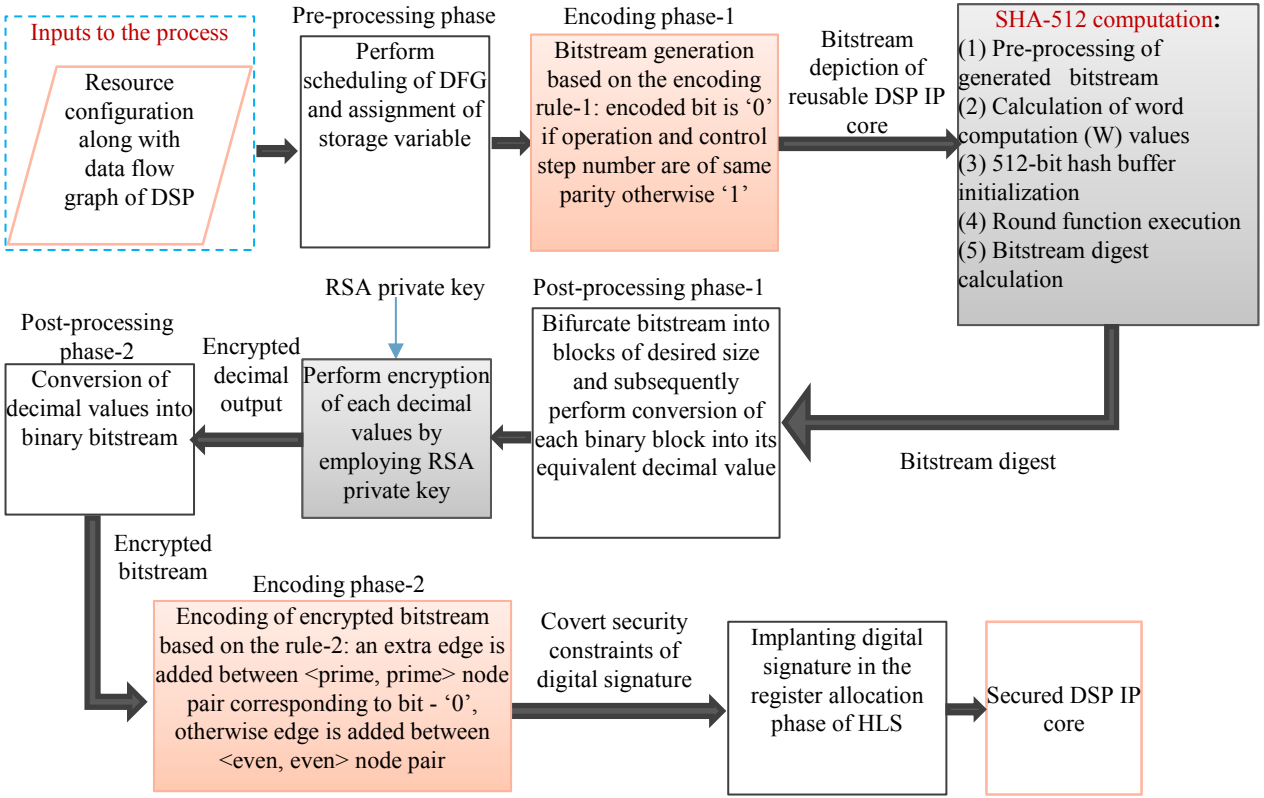


Fig. 18. Details of the digital signature embedding approach.

SDFG of 8-point DCT and its corresponding RAT [17]:

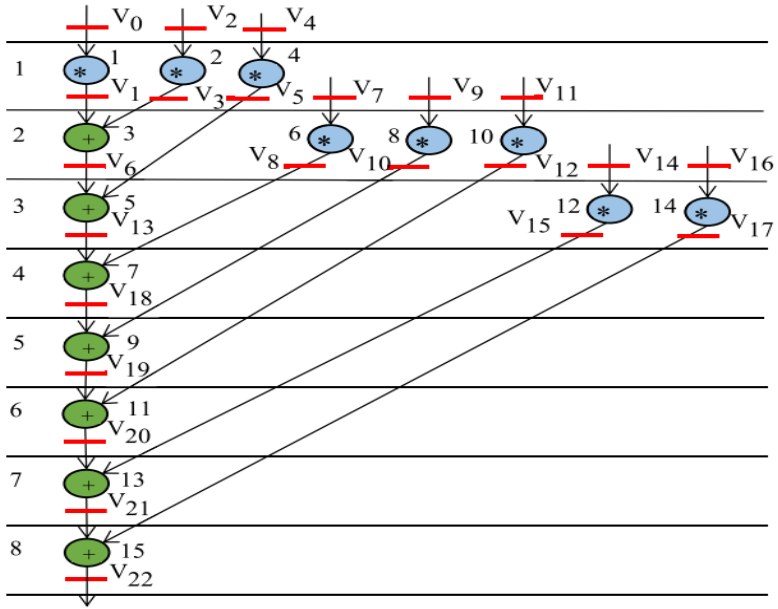


Fig. 19. Scheduled DFG of 8-point DCT with storage variables.

Control Step	B	R	G	O	Y	C	Bl
0	V_0	V_2	V_4	-	-	-	-
1	V_1	V_3	V_5	V_7	V_9	V_{11}	-
2	V_6	V_8	V_5	V_{10}	V_{12}	V_{14}	V_{16}
3	V_{13}	V_8	V_{15}	V_{10}	V_{12}	V_{17}	-
4	V_{18}	-	V_{15}	V_{10}	V_{12}	V_{17}	-
5	V_{19}	-	V_{15}	-	V_{12}	V_{17}	-
6	V_{20}	-	V_{15}	-	-	V_{17}	-
7	V_{21}	-	-	-	-	V_{17}	-
8	V_{22}	-	-	-	-	-	-

Table 16. Register allocation table of 8-point DCT before embedding digital security constraints

Steps of the proposed algorithm [17]:

- A bitstream is generated using IP vendor selected encoding mechanism.
- Next, bitstream was passed to SHA-512 to generate message digest.
- Further, message digest was segregated and grouped into blocks of fixed size and converted into decimal.
- Now, RSA asymmetric cryptographic algorithm was used to encrypt the decimal values.
- Again, the obtained Decimal value was converted back to binary bits.
- Then, generated digital signature is encoded into hardware security constraints using IP vendor selected mapping/ embedding rules.
- **Embedding rules:** If bit=0, then additional edge is between node pair(prime, prime) and if bit=1, then additional edge is between node pair(even, even) in CIG.

CIG of 8-point DCT and RAT after digital signature embedding [17]:

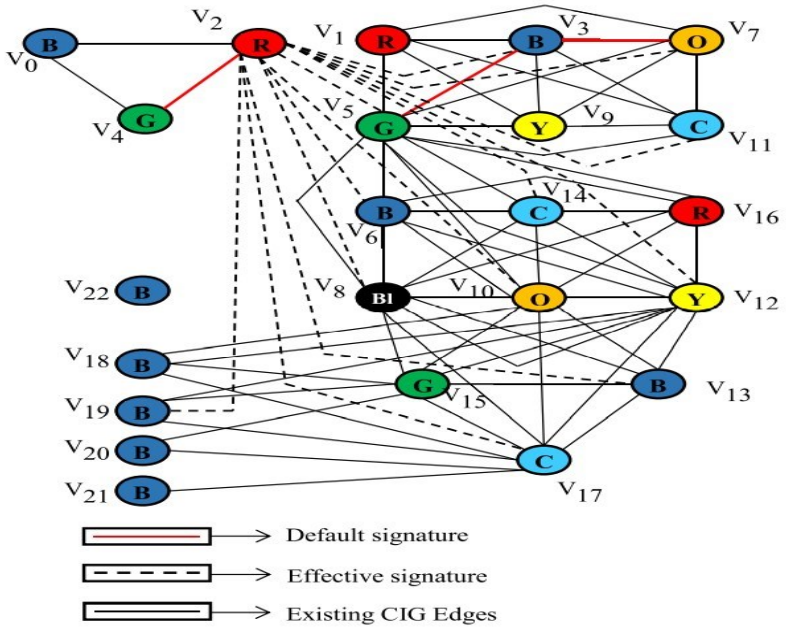


Fig. 20. Colored interval graph of 8-point DCT after embedding digital signature constraints

Control Step	B	R	G	O	Y	C	Bl
0	V_0	V_2	V_4	-	-	-	-
1	V_3	V_1	V_5	V_7	V_9	V_{11}	-
2	V_6	V_{16}	V_5	V_{10}	V_{12}	V_{14}	V_8
3	V_{13}	-	V_{15}	V_{10}	V_{12}	V_{17}	V_8
4	V_{18}	-	V_{15}	V_{10}	V_{12}	V_{17}	-
5	V_{19}	-	V_{15}	-	V_{12}	V_{17}	-
6	V_{20}	-	V_{15}	-	-	V_{17}	-
7	V_{21}	-	-	-	-	V_{17}	-
8	V_{22}	-	-	-	-	-	-

Table 17. Register allocation table of 8-point DCT post embedding digital security constraints

Results compared to [14]:

Benchmarks	Percentage reduction in P_c				
	S = 15	S = 30	S = 60	S = 120	S = 240
BPF	34.44	57.05	81.55	96.60	99.88
JPEG SAMPLE	24.05	42.31	66.74	88.94	98.78
JPEG IDCT	1.78	3.51	6.88	13.45	24.84
MESA FEEDBACK POINTS	9.36	17.87	32.56	54.47	79.27
DWT	0	0	0	0	0
ARF	0	0	0	0	0
IIR BUTTERWORTH	0	0	0	0	0
MESA MATRIX MULTIPLICATION	2.78	5.53	10.9	20.31	36.5

Table 18. Percentage reduction in P_C value achieved for proposed approach compared to prior work [14] for different DSP cores.

Results (Contd.):

Benchmarks	Design cost of proposed approach					Design cost of [14]				
	S = 15	S = 30	S = 60	S = 120	S = 240	S = 15	S = 30	S = 60	S = 120	S = 240
BPF	0.4129	0.4134	0.4134	NA	NA	0.4140	0.4145	0.4143	NA	NA
JPEG SAMPLE	0.4285	0.4294	0.4294	0.4294	NA	0.4506	0.4506	0.4512	0.4512	NA
MESA MATRIX MULTIPLICATION	0.2687	0.2687	0.2687	0.2687	0.2687	0.2790	0.2790	0.2792	0.2795	0.2795
JPEG IDCT	0.2160	0.2160	0.2160	0.2160	0.2160	0.2333	0.2333	0.2333	0.2335	0.2335
MESA FEEDBACK POINTS	0.3141	0.3141	0.3144	0.3144	0.3144	0.3189	0.3189	0.3191	0.3194	0.3196
DWT	0.6409	0.6409	NA	NA	NA	0.6630	0.6630	NA	NA	NA
ARF	0.3765	0.3765	0.3765	0.3766	NA	0.3771	0.3768	0.3773	0.3774	NA
IIR BUTTERWORTH	0.5465	NA	NA	NA	NA	0.5469	NA	NA	NA	NA

Table 19. Design cost analysis of proposed approach and [14] (after embedding signature).

Contact-Less Palmprint Biometric for Securing DSP Coprocessors Used in CE Systems [11]:

- A novel approach named Contact-Less Palmprint Hardware security is used.
- The proposed approch is contactless as while during verification process, it is not again required to capture the palmprint image.
- A digital template/ palmprint signature is generated based on the authentic palmprint biometric image.
- Generated palmprint biometric based signature is converted into its corresponding hardware secuirty constraints based on IP vendor/ designer selected mapping/ embedding mechanism.
- Generated secret palmprint biometric based security constraints are embedded in DSP(like FIR, 8-piont DCT) design to detect counterfeited versions.

[11]. A. Sengupta, R. Chaurasia and T. Reddy, "Contact-Less Palmprint Biometric for Securing DSP Coprocessors Used in CE Systems," in *IEEE Transactions on Consumer Electronics*, vol. 67, no. 3, pp. 202-213, Aug. 2021.

Advantages of the palmprint biometric based approach [11]:

- The digital template generated here is unique as palmprint image being biologically unique can not be replicated.
- Recapturing is not required.
- There is no need of optical scanner as like in fingerprint biometric methodology [19]..
- Inert from effects of external factors like grease and dirt.
- Having no dependencies on secret keys and contact-less in nature.
- Less complex than facial biometric [10] and fingerprint biometric [19] based hardware security methodologies..
- Huge combination of set of nodal points.

Flow chart of the proposed approach [11]:

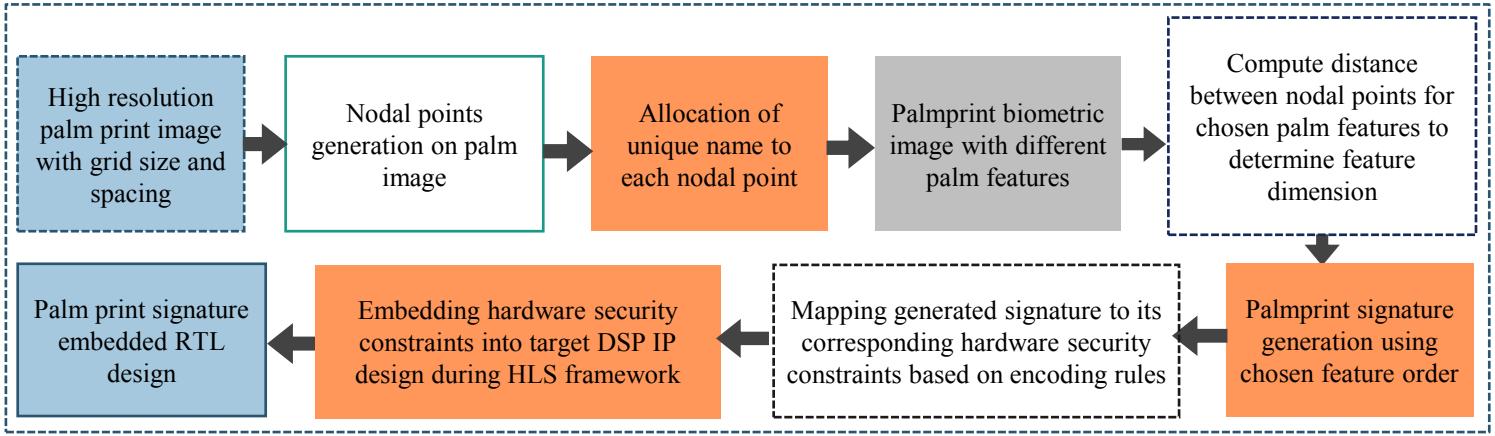


Fig. 21. Flow chart of palmprint biometric approach [11]

Feature generation on palmprint [11]:

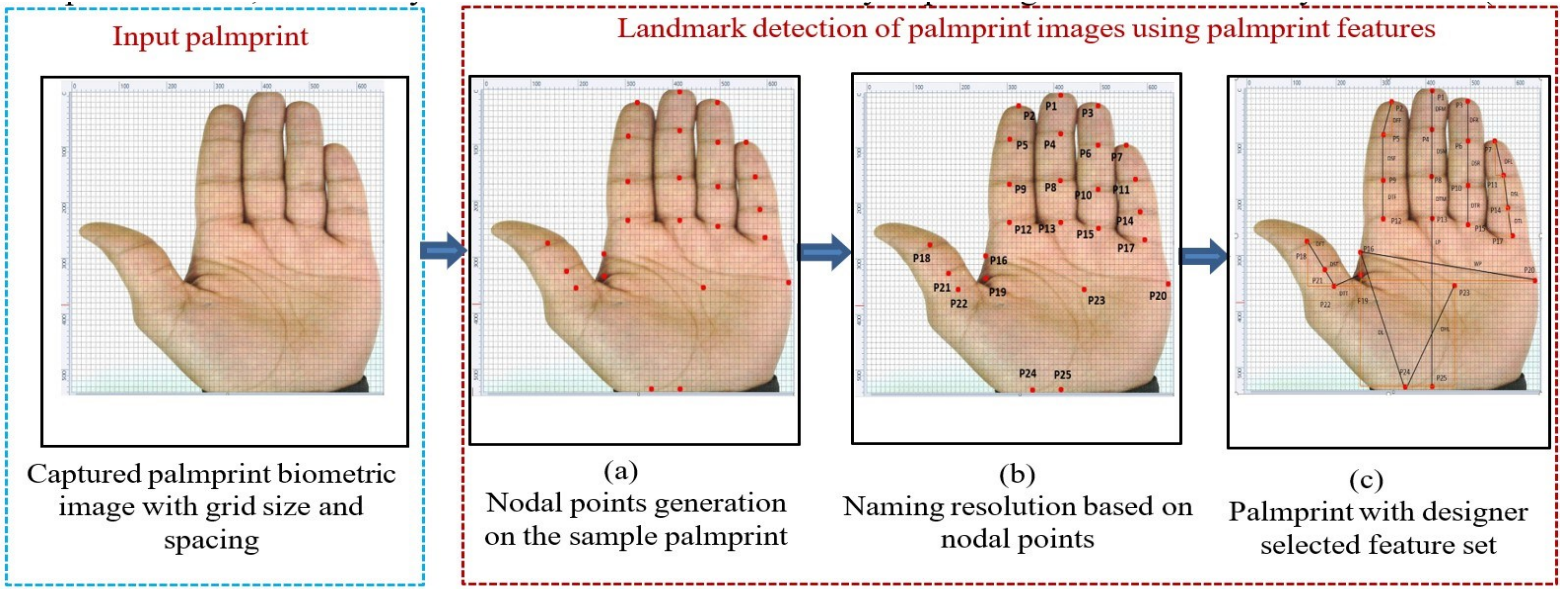
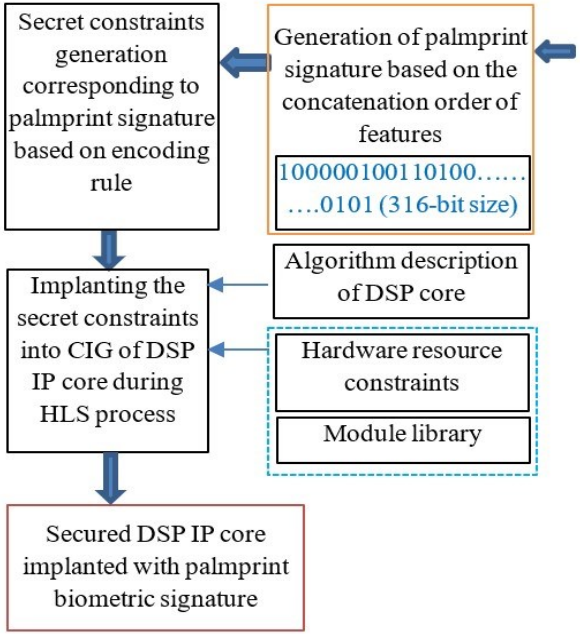


Fig. 22. Feature generation on palmprint image

Feature mapping and generation of secured DSP IP core [11]:

- Here Manhattan distance is used to compute dimension of straight line feature and for diagonal one Pythagoras theorem is used.



Feature #	Feature name	Feature dimension	Binary representation
F1	DL	260	100000100
F2	DHL	210.95	11010010.11110011001100110011
F3	WP	393.19	110001001.00110000101000111101
F4	LP	300	100101100
F5	DFF	63.24	111111.00111110101110000101
F6	DSF	80	1010000
F7	DTF	70	1000110
F8	DFM	70	1000110
F9	DSM	80	1010000
F10	DTM	80	1010000
F11	DFR	70	1000110
F12	DSR	80	1010000
F13	DTR	70	1000110
F14	DFL	63.24	111111.00111110101110000101
F15	DSL	60.82	111100.11010001111010111
F16	DTL	50.99	110010.11111101011100001 01
F17	DFT	64.03	1000000.00000111101011100001
F18	DST	36.05	100100.00001100110011001101
F19	DTT	63.24	111111.00111110101110000101

Fig. 23. Process of generation of secured reusable IP core using contact-less palmprint biometric based approach.

Mapping rules and Register allocation table corresponding to FIR:

Digits	Mapping rules
0	Implant an edge between node pair (even, even) into CIG
1	Implant an edge between node pair (odd, odd) into CIG
.	Implant an edge between node pair (0, integer) into CIG

Table 20. Palmprint signature to security constraints mapping rules.

	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9
P	T0	T8	T16	T24	T25	T26	T27	T28	T29	T30
I	T1	T9	T17	--	--	--	--	--	--	--
V	T2	T10	T18	T18	--	--	--	--	--	--
G	T3	T11	T19	T19	T19	--	--	--	--	--
Y	T4	T4	T12	T20	T20	T20	--	--	--	--
O	T5	T5	T13	T21	T21	T21	T21	--	--	--
R	T6	T6	T14	T22	T22	T22	T22	T22	--	--
B	T7	T7	T15	T23	T23	T23	T23	T23	T23	--

Table 21. Register assignment of storage variables (T0-T30) of FIR before embedding.

Register allocation table (after) corresponding to FIR:

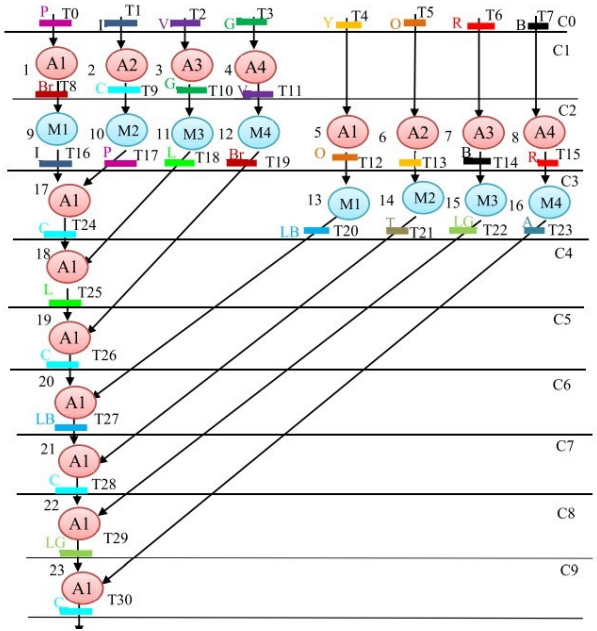


Fig. 24. Scheduled FIR post implanting palmprint biometric based hardware security constraints.

	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9
P	T0	--	T17	--	--	--	--	--	--	--
I	T1	--	T16	--	--	--	--	--	--	--
V	T2	T11	--	--	--	--	--	--	--	--
G	T3	T10	--	--	--	--	--	--	--	--
Y	T4	T4	T13	--	--	--	--	--	--	--
O	T5	T5	T12	--	--	--	--	--	--	--
R	T6	T6	T15	--	--	--	--	--	--	--
B	T7	T7	T14	--	--	--	--	--	--	--
Br	--	T8	T19	T19	T19	--	--	--	--	
C	--	T9	--	T24	--	T26	--	T28	--	T30
L	--	--	T18	T18	T25	--	--	--	--	--
LB	--	--	--	T20	T20	T20	T27	--	--	--
LG	--	--	--	T22	T22	T22	T22	T22	T29	--
T	--	--	--	T21	T21	T21	T21	--	--	--
A	--	--	--	T23	T23	T23	T23	T23	T23	--

Table 21. Register assignment of storage variables (T0-T30) of FIR.

Results compared to [13], [19]:

Bench-marks	Proposed		Related work [19]	
	Maximum constraints	Pc	Maximum constraints	Pc
4-point DCT	27	4.23E-4	25	7.52E-4
4-point IDCT	27	4.23E-4	25	7.52E-4
8-point DCT	125	5.63E-8	121	9.61E-8
8-point IDCT	125	5.63E-8	121	9.61E-8
FIR	231	4.01E-14	225	8.95E-14

Table 22. Comparison of Pc w.r.t. related work [19].

Bench-marks	Proposed		Related work [19]	
	Signature size (S)	TT	Signature size (S)	Tamper tolerance
4-point DCT	27	7.6E+12	25	3.3E+7
4-point IDCT	27	7.6E+12	25	3.3E+7
8-point DCT	125	4.3E+59	121	2.6E+36
8-point IDCT	125	4.3E+59	121	2.6E+36
FIR	231	1.6E+110	225	5.4E+67

Table 23. Comparison of TT w.r.t. related work [19].

Bench-marks	Pc		TT	
	Proposed	[13]	Proposed	[13]
4-point DCT	4.23E-4	1.00E-3	7.6E+12	1.6E+7
4-point IDCT	4.23E-4	1.00E-3	7.6E+12	1.6E+7
8-point DCT	5.63E-8	2.22E-4	4.3E+59	9.22E+18
8-point IDCT	5.63E-8	2.22E-4	4.3E+59	9.22E+18
FIR	4.01E-14	4.94E-4	1.6E+110	1.44E+17

Table 25. Comparison of proposed approach [13].

Facial Biometric for Securing Hardware Accelerators [10]:

- A novel approach hardware security approach using facial biometrics.
- The proposed approch is contactless as while during verification process, it is not again required to capture the facial image.
- A digital template/ facial signature is generated based on the authentic facial biometric image.
- Generated facial biometric based signature is converted into its corresponding hardware secuirty constraints based on IP vendor/ designer selected mapping/ embedding mechanism.
- Generated secret facial biometric based security constraints are embedded in DSP(like FIR, 8-piont DCT) design to detect counterfeited versions.

[10]. A. Sengupta and M. Rathor, “Facial Biometric for Securing Hardware Accelerators,” in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 29, no. 1, pp. 112-123, Jan. 2021.

Advantages of the facial biometric based approach [10]:

- The digital template generated here is unique as facial image being biologically unique can not be replicated.
- Recapturing of facial image is not required.
- There is no need of optical scanner as like in fingerprint biometric methodology [19].
- Inert from effects of external factors like grease and dirt.
- Having no dependencies on secret keys and contact-less in nature.
- Less complex than fingerprint biometric [19] as it does not contains pre-processing steps like binarization, thinning, etc., which are crucial steps in fingerprint biometrics based hardware security methodology [19].
- Huge combination of set of nodal points.

Flow chart of the proposed approach [10]:

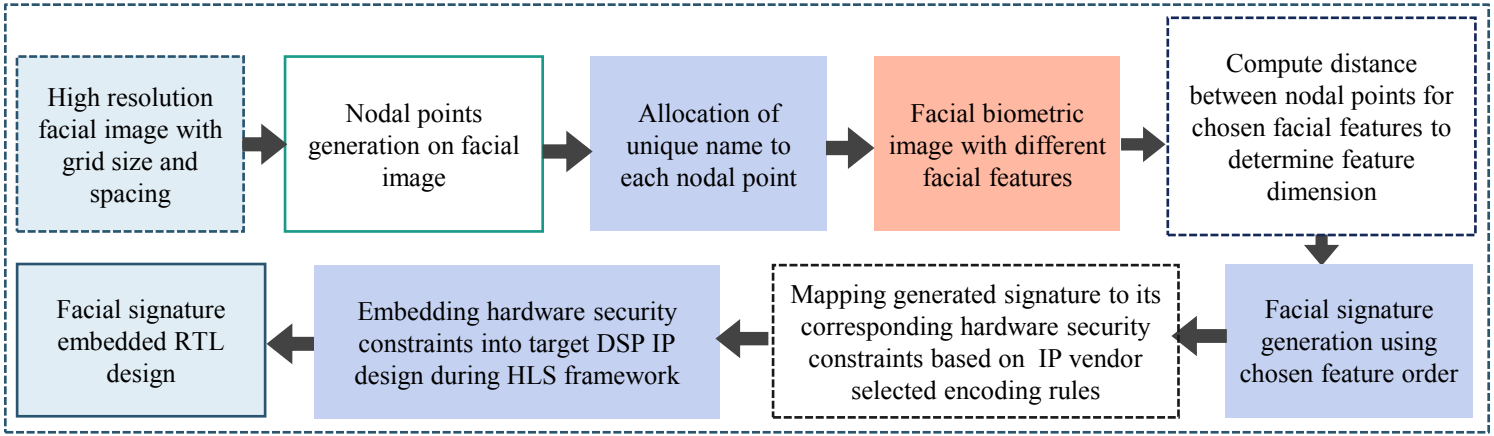


Fig. 25. Flow chart of facial biometric approach [10]

Overview of facial biometric based security methodology [10]:

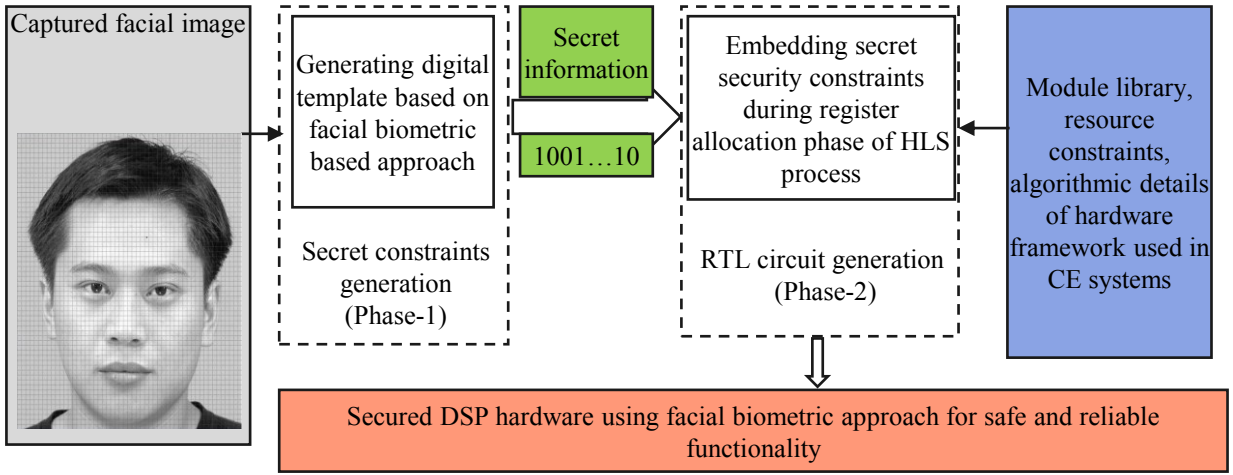


Fig. 26. Overview of facial biometric based hardware security methodology

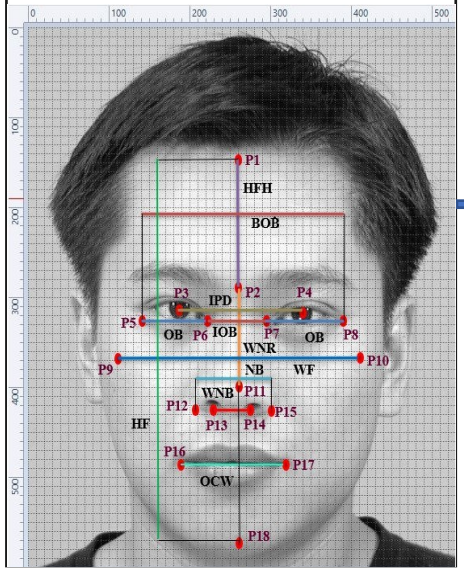
Feature generation on Facial biometric image [11]:

Assign naming conventions on nodal points

Nodal points generation process

Designer selected feature set
HFH: Height of Forehead (P1-P2)
IPD: Inter Pupillary Distance (P3-P4)
BOB: Bio- Ocular Breadth (P5-P8)
IOB: Inter – Ocular Breadth (P6-P7)
OB: Ocular Breadth (P5-P6) or (p7-p8)
WNR: Width of Nasal Ridge (P2-P11)
WF: Width of face (P9-P10)
HF: Height of Face (P1-P18)
WNB: Width of Nasal Base (P13-P14)
NB: Nasal Breadth (P12-P15)
OCW: Oral Commissure Width (P16-P17)

Generate the facial image with chosen features



Generating the facial signature based on the vendor decided feature order (Generated facial signature)

10010001100110111111010 1001011
10100001101110100101100 11010100
1101101101111110000010

Calculation of feature dimensions between nodal points of chosen features

Feature dimension evaluated using Manhattan distance = |x2-x1|+|y2-y1|	Binarize value
145	10010001
155	10011011
250	11111010
75	1001011
80	1010000
110	1101110
300	100101100
425	110101001
45	101101
95	1011111
130	10000010

Fig. 27. Feature generation on Facial image

Feature mapping and generation of secured DSP IP core [10]:

- Here Manhattan distance is used to compute dimension of straight line feature and for diagonal one Pythagoras theorem is used.

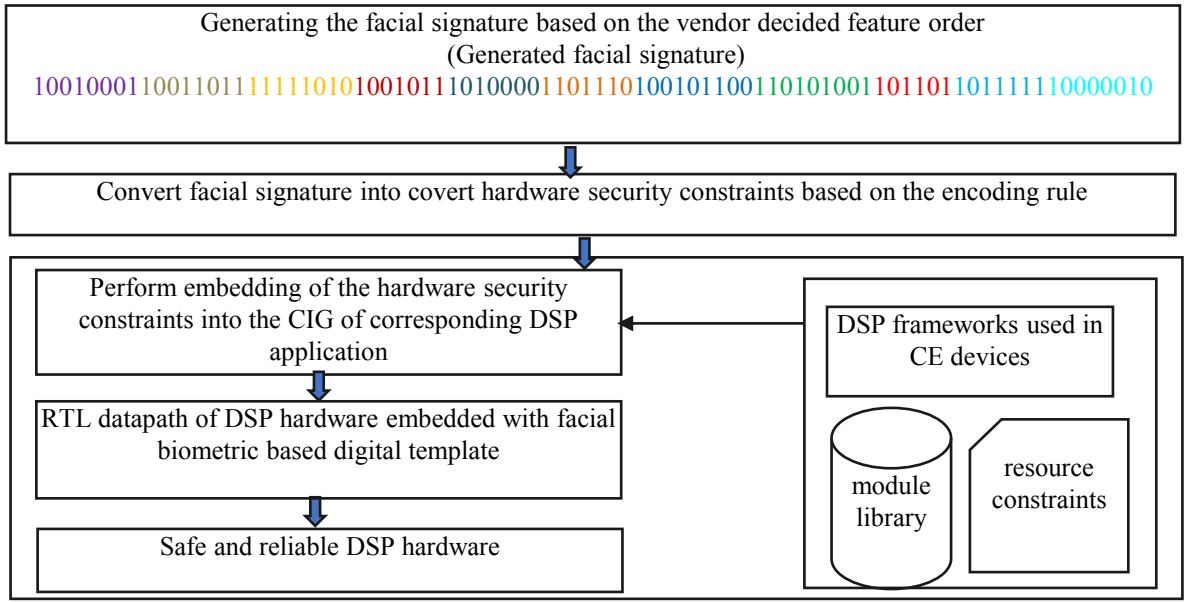


Fig. 28. Process of generation of secured reusable IP core using contact-less facial biometric based approach.

Mapping rules:

Bit	Encoding rules
0	Encoded as an edge between node pair (even, even) into the CIG
1	Encoded as an edge between node pair (odd, odd) into the CIG

Table 25. Facial signature to security constraints mapping rules.

SDFG (before and after embedding facial signature) for DCT:

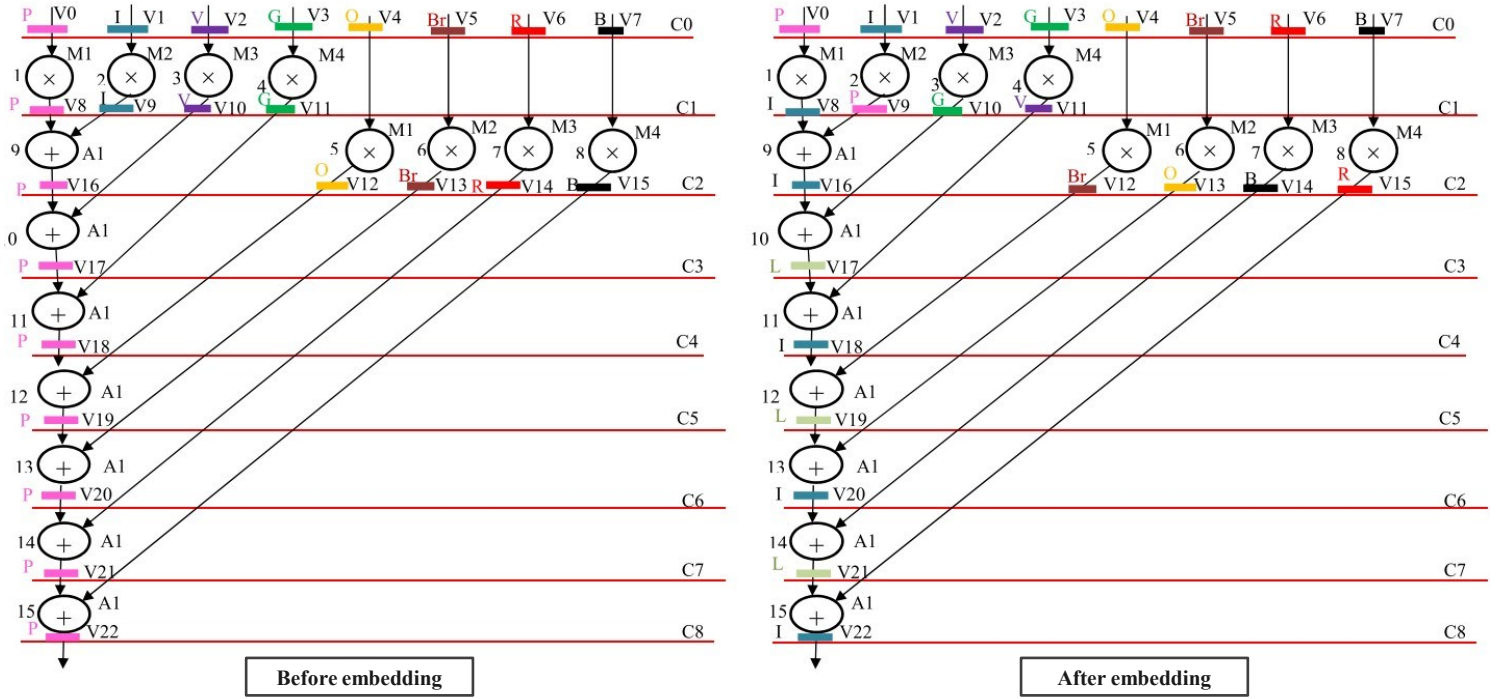


Fig. 29. Scheduled DCT before and post implanting facial biometric based hardware security constraints.

Results compared to [18]:

Bench-marks	# colors (registers)	Proposed work						Related work [18]	
		Image_1		Image_2, 3, and 4		Image_5		# stego-constraints	Pc
		(# biometric constraints)	Pc	(# biometric constraints)	Pc	(# biometric constraints)	Pc		
8-point DCT	8	81	2.01E-5	84	1.34E-5	83	1.54E-5	13	1.8E-1
								24	4.1E-2
								43	3.2E-3
8-point IDCT	8	81	2.01E-5	84	1.34E-5	83	1.54E-5	13	1.8E-1
								24	4.1E-2
								43	3.2E-3
FIR	8	81	2.01E-5	84	1.34E-5	83	1.54E-5	20	6.9E-2
								57	4.9E-4
MPEG	14	81	2.47E-3	84	1.98E-3	83	2.13E-3	21	2.1E-1
								52	2.1E-2
								59	1.3E-2

Table 26. Comparison of Pc with respect to related work [18] for the facial.

Securing Hardware Accelerators for CE Systems Using Biometric Fingerprinting [19]:

- A novel hardware security approach based on the fingerprint biometric.
- Recapturing of fingerprint image is not required at the time of verification (authentication of IPs).
- A digital template/ figerprint biometric based signature is generated based on the authentic fingerprint biometric image.
- Minutiea points feature of the fingerprint biometric has been used to genrete covert fingerprint based signature.
- Generated fingerprint biometric based signature is converted into its corresponding hardware secuirty constraints based on IP vendor/ designer selected mapping/ embedding mechanism.
- Generated secret palmprint biometric based security constraints are embedded in DSP(like FIR, 8-piont DCT) design to detect counterfeited versions.

[19]. A. Sengupta and M. Rathor, “Securing Hardware Accelerators for CE Systems Using Biometric Fingerprinting,” in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 9, pp. 1979-1992, Sept. 2020.

Advantages and limitations of the fingerprint biometric based approach [19]:

Advantages:

- The digital template generated here is unique as fingerprint being biologically unique can not be replicated.
- Recapturing is not required.
- Having no dependencies on secret keys and contact-less in nature.
- Huge combination of set of minutiae points.

Limitations:

- Here, an optical scanner is required at starting to capture the fingerprint biometric image.
- Gets affected due to external factors like grease and dirt.
- More complex than facial biometric [10], palmprint biometric [11].

Flow chart of the proposed approach [19]:

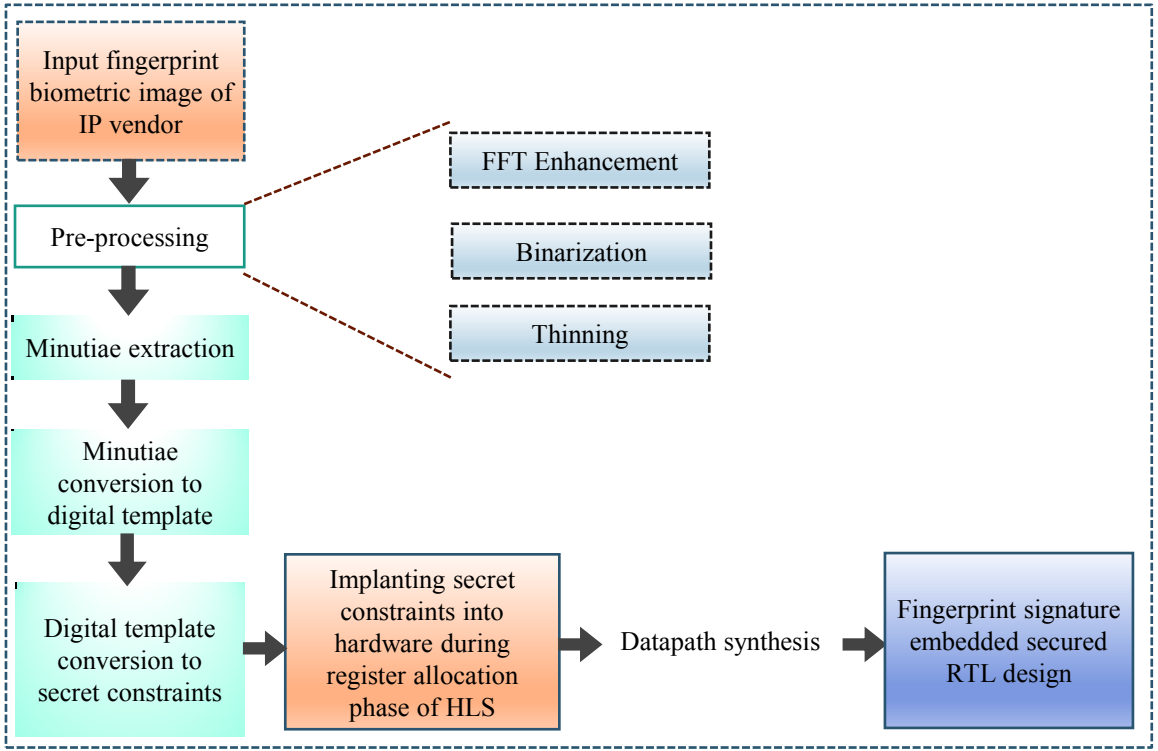


Fig. 30. Flow chart of fingerprint biometric approach [19]

Preprocessing step [19]:

- **FFT Enhancement:**

The use of FFT on sets of pixels of the fingerprint image allows the reconnection of broken ridges, finely separates the parallel ridges, and also makes the ridges thick.

- **Binarization:**

The image with only two intensity values is called as binary image. This image usually shows only black or white, where black is represented by 0 and white is represented by 255. The elementary principle of binarization is to compare the pixel intensities with the threshold, and setting the pixels whose intensities are less than threshold, to 0 and the other to 255.

- **Thinning:**

In this process, the thickness of ridge lines is reduced to one pixel width by deleting pixels at the edge of ridge lines.

Feature generation on fingerprint buiometric image [19]:

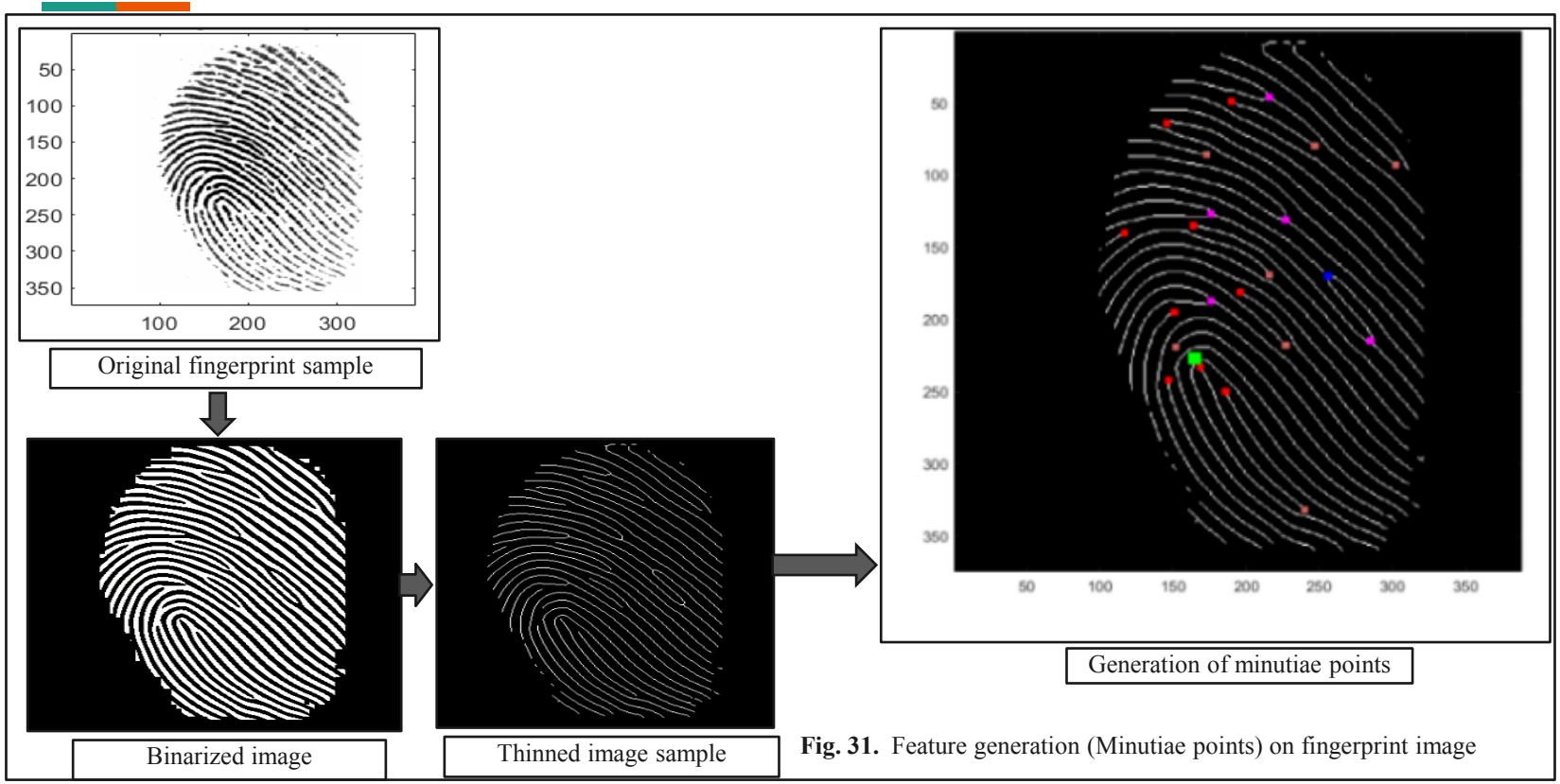


Fig. 31. Feature generation (Minutiae points) on fingerprint image

Feature determination [19]:

- Co-ordinates of minutiae's points and its angle acts as feature dimension and is further converted into its equivalent binary form to generate final fingerprint based signature.

No.	x	y	Minutiae type number	Minutiae type name	Minutiae type color	Angle in radian	Angle in degree
1	216	46	3	Ridge bifurcation	Pink	0.503045247761871	28.8
2	190	49	1	Ridge ending	Red	3.58273130505241	205.3
3	146	64	1	Ridge ending	Red	3.26835482366637	187.3
4	247	80	1	Ridge ending (short ridge)	Red (brown)	0.700240286597619	40.1
5	173	86	1	Ridge ending (short ridge)	Red (brown)	0.366566654990681	21.0
6	302	93	1	Ridge ending (short ridge)	Red (brown)	0.837181054258808	48.0

Table 27. Determination of features corresponding to minutiae points.

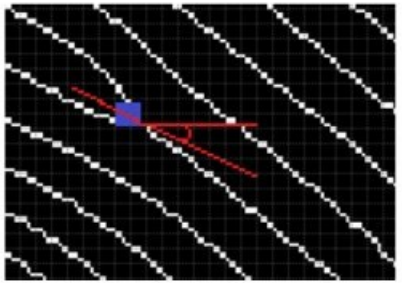


Fig. 32. Determination of ridge direction/angle of a minutiae point.

Signature generation [19]:

No.	x	y	Minutiae type number	Angle in degree	Binary	bits
1	216	46	3	29	110110001011 101111101	**21**
2	190	49	1	205	101111101100 01111001101	**23**
3	146	64	1	187	100100101000 000110111011	**24**
4	247	80	1	40	111101111010 0001101000	**22**
5	173	86	1	21	101011011010 110110101	**21**
6	302	93	1	48	100101110101 11011110000	**23**

Table 28. Binary representation of features corresponding to minutiae points.

Final generated signature:

"1101100010111011111011 0111110110001111001101 1 0010010100000011011 1 0111111011110100001101 0 0010101101101011011010 1 1001011101011101111 0 000".

Mapping rules:

Bit	Mapping rules
0	Embed an edge between node pair (even, even) into the CIG (during register allocation of ESL synthesis)
1	Embed an edge between node pair (odd, odd) into the CIG (during register allocation of ESL synthesis)

Table 29. Palmprint signature to security constraints mapping rules.

Results compared to [21]:

Approaches	Total constraints	Pc	Approximate run time
Related work [21]	W=20	3.0e-1	~400 ms
	W=40	9.8e-2	
	W=60	1.9e-2	
	W=80	3.7e-3	
	W=100	1.7e-3	
Proposed work	M=35	4.35e-6	~185 ms

Table 30. Comparison of the proposed approach with related work [21].

Exploring Handwritten Signature Image Features for Hardware Security [22]:

- A novel approach based on handwritten signature is used for securing DSP hardwares.
- During verification process, it is not again required to aquire and capture the handwritten signature image.
- A digital template/ handwritten signature based template is generated based on the authentic handwritten signature image.
- Generated handwritten signature biometric based signature is converted into its corresponding hardware secuirty constraints based on IP vendor/ designer selected mapping/ embedding mechanism.
- Generated secret handwritten signature biometric based security constraints are embedded in DSP(like FIR, 8-piont DCT) design to detect counterfeited versions.

[11]. M. Rathor, A. Sengupta, R. Chaurasia and A. Anshul, "Exploring Handwritten Signature Image Features for Hardware Security," in *IEEE Transactions on Dependable and Secure Computing*, 2022.

Advantages of the handwritten signature biometric based approach [22]:

- The digital template generated here is unique as handwritten signature being biologically unique can not be replicated.
- Recapturing of handwritten signature image is not required.
- There is no need of optical scanner as like in fingerprint biometric methodology [19].
- Inert from effects of external factors like grease and dirt.
- Having no dependencies on secret keys and contact-less in nature.
- Less complex than facial biometric [10] and fingerprint biometric [19].
- Huge combination of features possible.

Overview of the proposed approach [22]:

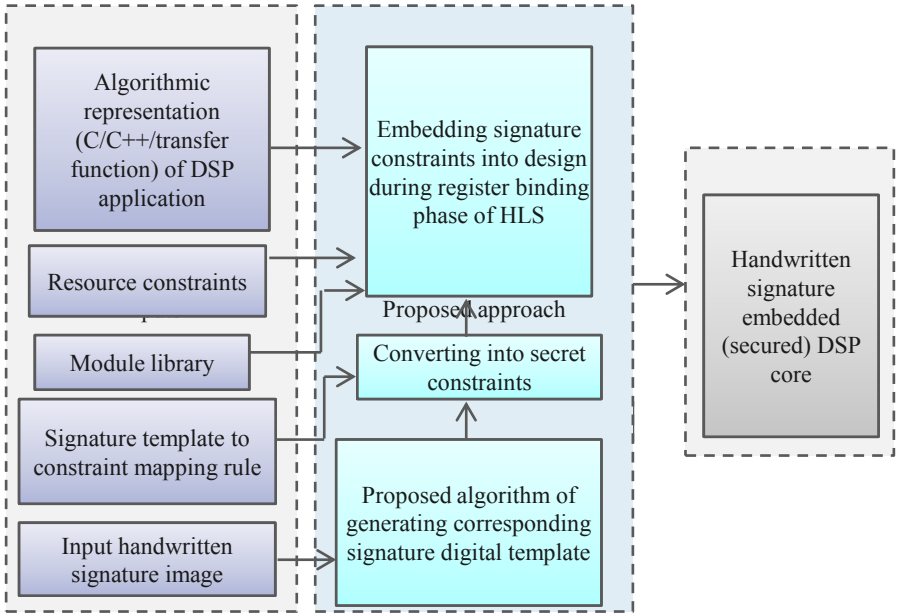


Fig. 33. Overview of handwritten signature biometric approach

Flow chart of proposed approach [22]:

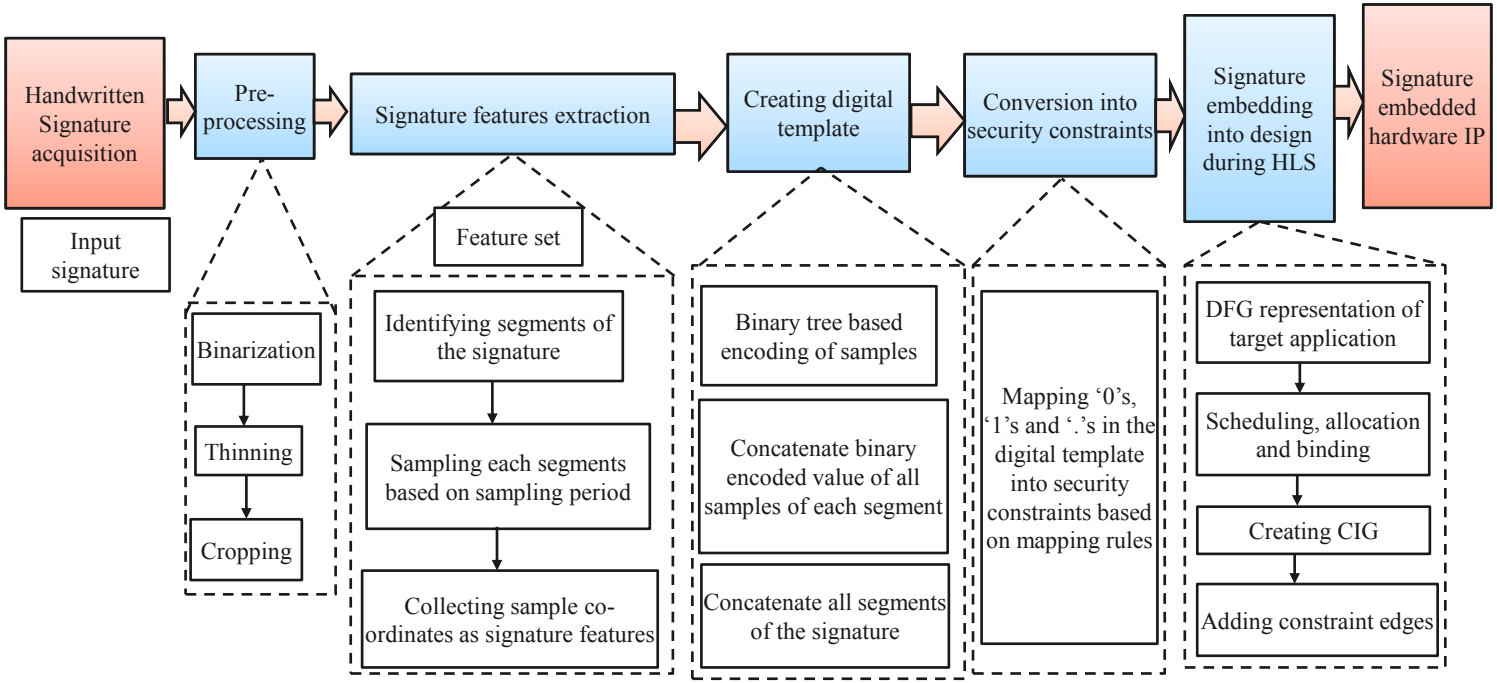


Fig. 34. Flow chart of handwritten signature based hardware security methodology[22]

Preprocessing step [22]:

- **Binarization:**

The image with only two intensity values is called as binary image. This image usually shows only black or white, where black is represented by 0 and white is represented by 255. The elementary principle of binarization is to compare the pixel intensities with the threshold, and setting the pixels whose intensities are less than threshold, to 0 and the other to 255.

- **Thinning:**

In this process, the thickness of signature lines is reduced to one pixel width by deleting pixels at the edge of ridge lines.

- **Cropping:**

This process crops the thinned signature image horizontally in between the leftmost and the rightmost pixel and vertically in between the uppermost and the bottommost pixel.

Segmentation [22]:

- **Segmentation:** In this step, the processed signature is segmented into chosen number of segments. To do so, the connected components in the handwritten signature are identified. A large segment can also be divided in further segments by dividing it based on chosen grid spacing.

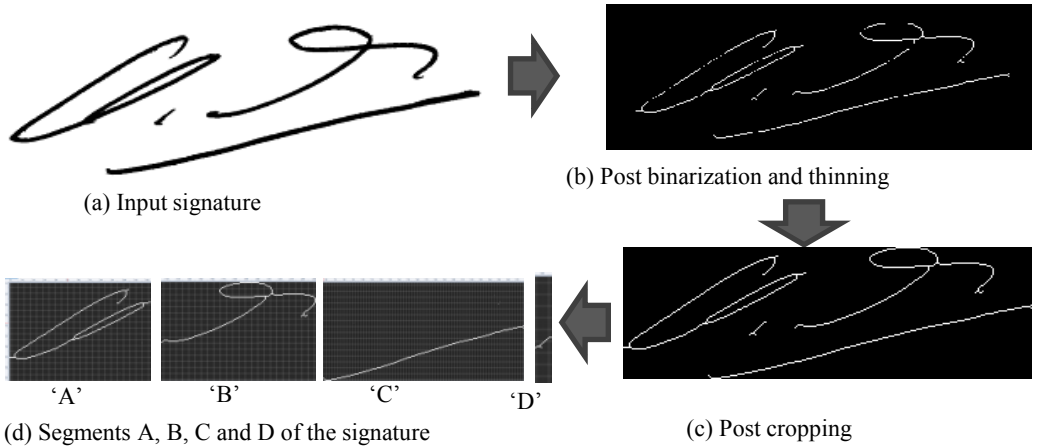


Fig. 35. Pre-processing and segmentation of signature

Sampling [22]:

- **Segmentation:** The sampling is performed based on the designer's chosen sampling period which is defined as the distance of two sample points in terms of the number of pixels the second sample point is situated farther from the first. The proposed sampling process starts to sample from the uppermost end point (pixel) of a segment.

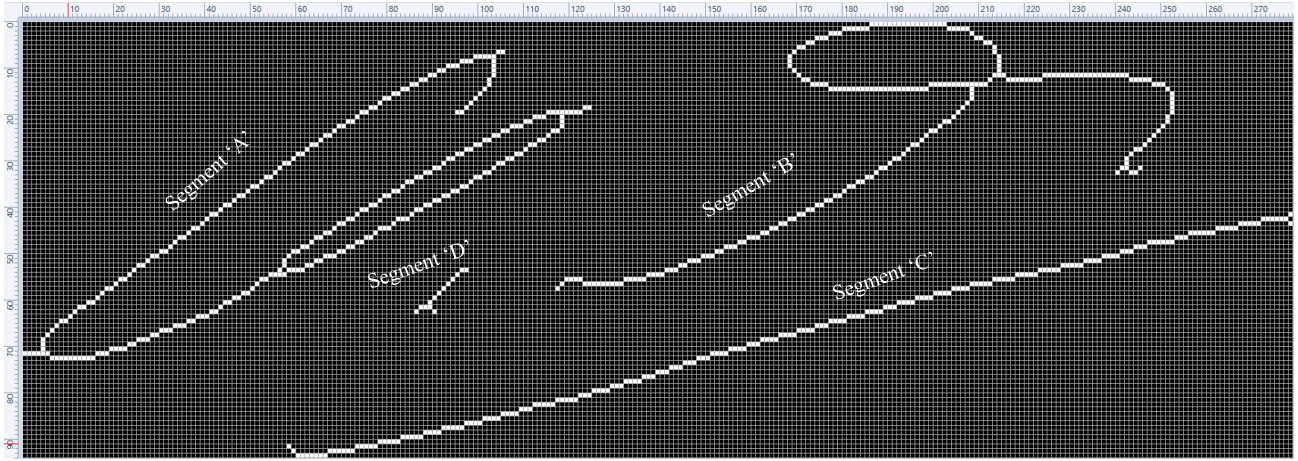


Fig. 36. Segmentation of signature

Features generation and their repective co-ordibates after sampling [22]:

S_t^A=50, S_t^B=20, S_t^C=70, S_t^D=14

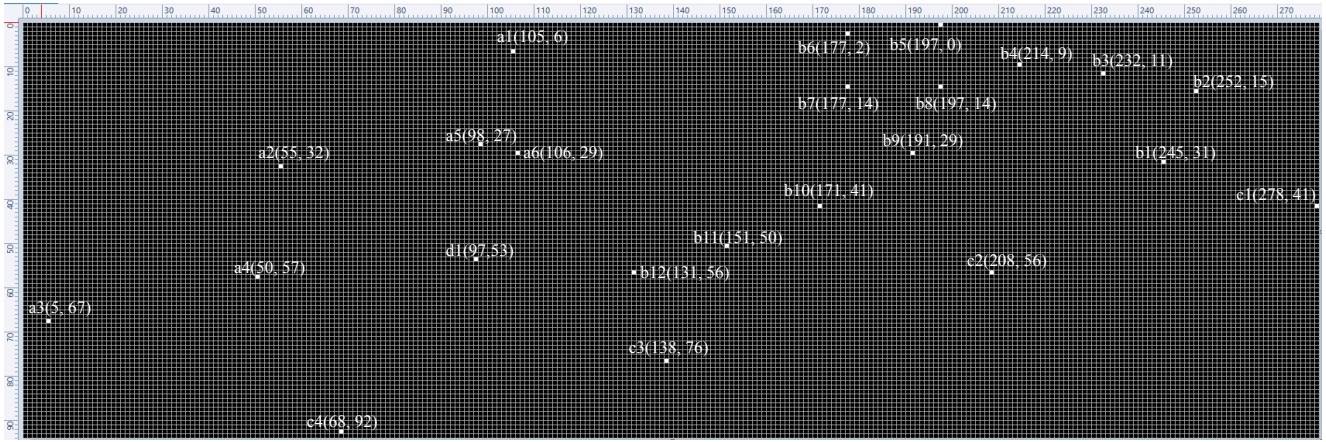
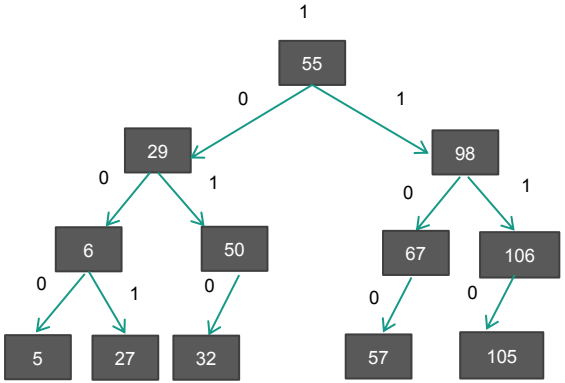


Fig. 37. Sampling of signature based on chosen sampling periods S_t^A, S_t^B, S_t^C and S_t^D for segments A, B, C and D respectively

F={{a1,a2,a3,a4,a5,a6},{b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,b11,b12},{c1,c2,c3,c4},{d1}}

Binary tree encoding [22]:

- **Encoding:** To encode the samples of a segment into binary values, the list of all ‘x’ and ‘y’ coordinates of the samples are first sorted in the increasing order and then a binary tree is created.
- **For example,** sorted list ‘L’ created using the ‘x’ and ‘y’ coordinates of the samples (a1 to a6) of segment ‘A’ is as follows: L={5,6,27,29,32,50,55,57,67,98,105,106}. Here, since E=12 therefore the element (55) at the 6th index becomes the root node. All the parent nodes and corresponding child trees (or elements) are chosen in the same fashion.



Samples of ‘A’	Coordinates (x,y)	Encoded value
a1	(105,6)	1110,100
a2	(55,32)	1,1010
a3	(5,67)	1000,110
a4	(50,57)	101,1100
a5	(98,27)	11,1001
a6	(106,29)	111,10

Fig. 38. Binary tree for segment ‘A’ and encoded values of its sample coordinates

Final signature and Mapping rules:

Final generated signature:

W="11001011.110101.11100100.110101001.111010010.111110.1110000.110101000.1111010.110001.110110100.111001.1011100.1110100.11010.1000110.111001.11110.1111000.110101.101.11100.10"

It is important to note here that the inter segment and intra segment concatenation points are coded as binary points for creating the digital template.

Bit '1'	encoded as an edge between node pair (even, even) into the CIG.
Bit '0'	encoded as an edge between node pair (odd, odd) into the CIG.
Binary point '.'	encoded as an edge between node pair (even, odd) into the CIG.

Table 31. Handwritten signature to security constraints mapping rules.

Results compared to [19], [10], [18], [13]:

DSP bench-mark	Proposed		[19]		[10]		[18]		[13]	
	# u	Pc	# u	Pc	# u	Pc	# u	Pc	# u	Pc
IIR	172	2.9e-6	169	3.6e-6	84	1.9e-3	57	1.4e-2	151	1.3e-5
DCT	128	3.7e-8	121	9.6e-8	84	1.3e-5	43	3.2e-3	111	3.6e-7
IDCT	128	3.7e-8	121	9.6e-8	84	1.3e-5	43	3.2e-3	111	3.6e-7
FIR	235	2.3e-14	225	8.9e-14	84	1.3e-5	57	4.9e-4	206	1.1e-12

Table 31. Comparison of Pc for proposed with related works.

IP Core Protection of Image Processing Filters With Multi-Level Encryption and Covert Steganographic Security Constraints [31]:

- The proposed approach based on multi-level encryption technique used for securing image processing filters IP cores. This paper discusses signature-based hardware security methodology on image processing IP cores for the first time.
- The proposed approach uses the register allocation table of the image processing application to generate secret data, which is used for multi-level encryption to determine hardware security constraints .
- The generated hardware security constraints then embedded in the image processing filter IP Cores design to authenticate genuine IP Maker.
- The huge variation in the key selection at different levels of encryption and the use of secret steganographic data for signature generation increases the robustness of the proposed hardware security methodology for image filters.
- Generated multi-level encryption based signature is converted into its corresponding hardware secuirty constraints based on IP vendor/ designer selected mapping/ embedding mechanism.

[31]. Aditya Anshul, Anirban Sengupta,"IP Core Protection of Image Processing Filters With Multi-Level Encryption and Covert Steganographic Security Constraints", Proceedings of *8th IEEE International Symposium on Smart Electronic Systems (IEEE – iSES)* , India, Accepted, Dec 2022.

Image processing filters:

- Image processing filters are mainly used to suppress either the high frequencies in the image, i.e. smoothing the image, or the low frequencies, i.e. enhancing or detecting edges in the image.
- The main objective of image processing is to extract some useful information from an image.
- From detection and recognition of license plates of vehicles on tolls (character recognition), advanced medical imagery (image analysis), biometric fingerprinting, robotics vision, and military operations to car driving automation, image processing plays a crucial role everywhere.
- Due to globalization of design supply chain, the design process of these image processing filters as a dedicated intellectual property (IP) core involves various hardware threats [27].

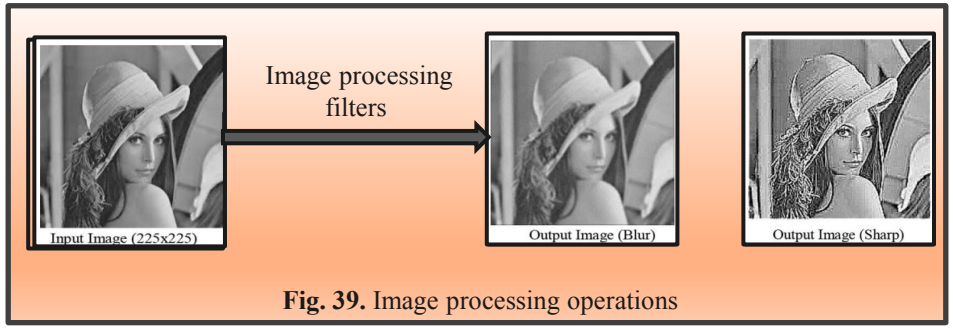


Fig. 39. Image processing operations

Flow chart of the proposed approach [31]:

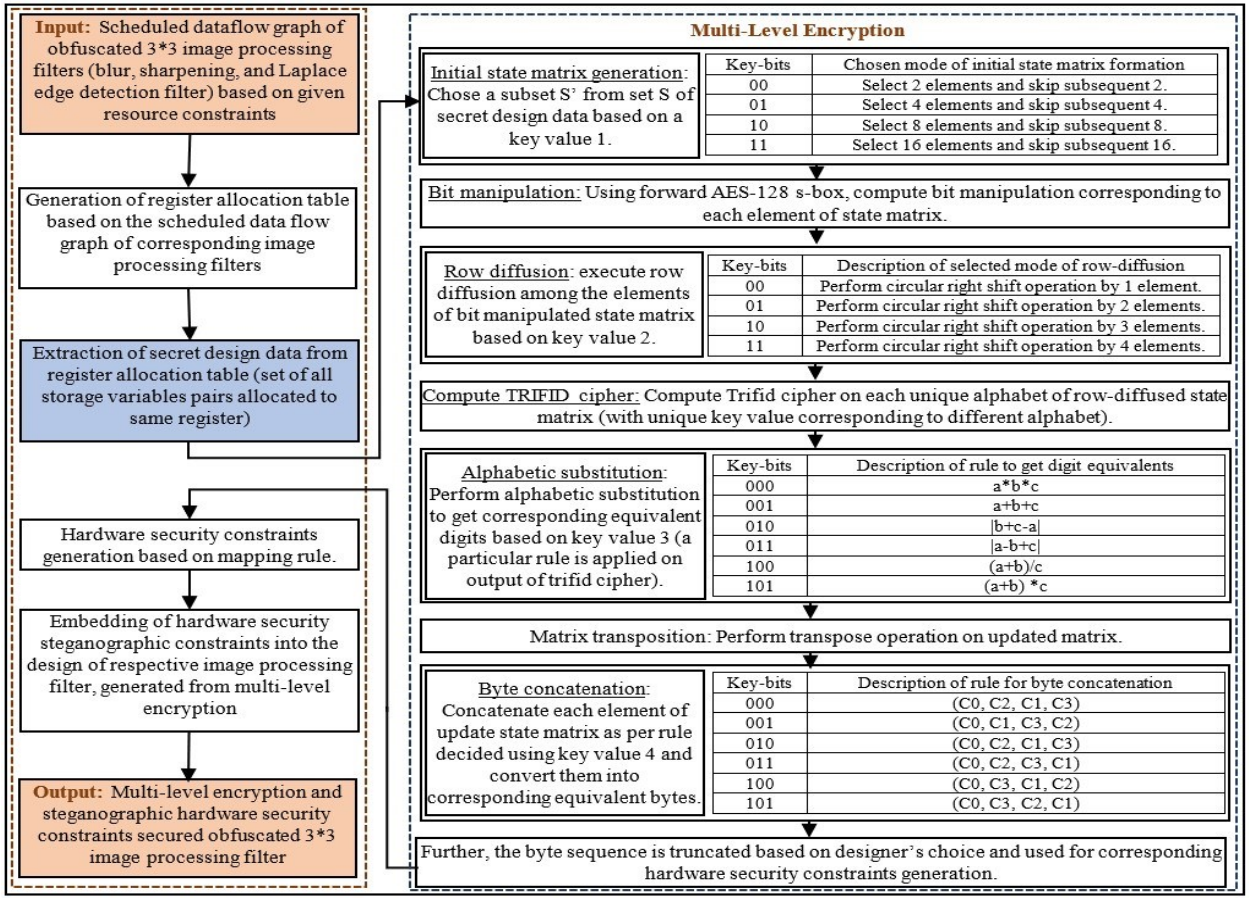


Fig. 40. Flow chart of multi-level encryption based approach [31]

Determination of secret design data based on scheduled data flow graph of image processing application [31]:

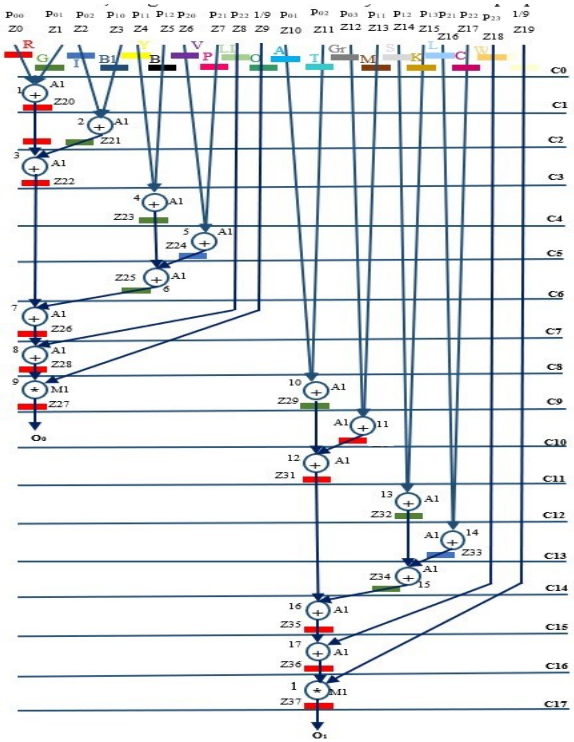


Fig. 41. Scheduled data flow graph of 3*3 blur filter with 1(+), 1(-) and 2(*) as resource constraints

CS	Red(R)	Green (G)	Indigo (I)	Blue (BL)	Yellow (Y)	Black (B)	Violet (V)	Pink (P)	Lime (LI)	Olive (O)	Aqua (A)	Teal (T)	Gray (G)	Maroon (M)	Silver (S)	Khaki (K)	Lavender (L)	Crimson (C)	Wheat (W)	Beige (B)
0	Z0	Z1	Z2	Z3	Z4	Z5	Z6	Z7	Z8	Z9	Z10	Z11	Z12	Z13	Z14	Z15	Z16	Z17	Z18	Z19
1	Z20	Z20	Z2	Z3	Z4	Z5	Z6	Z7	Z8	Z9	Z10	Z11	Z12	Z13	Z14	Z15	Z16	Z17	Z18	Z19
2	Z20/Z21	Z21/Z20	-		Z4	Z5	Z6	Z7	Z8	Z9	Z10	Z11	Z12	Z13	Z14	Z15	Z16	Z17	Z18	Z19
3	Z22	Z22	-		Z4	Z5	Z6	Z7	Z8	Z9	Z10	Z11	Z12	Z13	Z14	Z15	Z16	Z17	Z18	Z19
4	Z22/Z23	Z23/Z22	-	-	-	-	Z6	Z7	Z8	Z9	Z10	Z11	Z12	Z13	Z14	Z15	Z16	Z17	Z18	Z19
5	Z22/Z23	Z23/Z22	Z24	Z24	-	-	-	-	Z8	Z9	Z10	Z11	Z12	Z13	Z14	Z15	Z16	Z17	Z18	Z19
6	Z22/Z25	Z25/Z22	-	-	-	-	-	-	Z8	Z9	Z10	Z11	Z12	Z13	Z14	Z15	Z16	Z17	Z18	Z19
7	Z26	Z26	-	-	-	-	-	-	Z8	Z9	Z10	Z11	Z12	Z13	Z14	Z15	Z16	Z17	Z18	Z19
8	Z27	-	-	-	-	-	-	-	-	Z9	Z10	Z11	Z12	Z13	Z14	Z15	Z16	Z17	Z18	Z19
9	Z28/Z29	Z29/Z28	-	-	-	-	-	-	-	-	-	-	Z12	Z13	Z14	Z15	Z16	Z17	Z18	Z19
10	Z30/Z29	Z29/Z30	-	-	-	-	-	-	-	-	-	-	-	-	Z14	Z15	Z16	Z17	Z18	Z19
11	Z31	-	-	-	-	-	-	-	-	-	-	-	-	-	Z14	Z15	Z16	Z17	Z18	Z19
12	Z31	Z32	-	-	-	-	-	-	-	-	-	-	-	-	-	-	Z16	Z17	Z18	Z19
13	Z31	Z32	Z33	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	Z18	Z19
14	Z31	Z34	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	Z18	Z19
15	Z35	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	Z18	Z19
16	Z36	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	Z19
17	Z37	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Table 32. Register allocation table of 3*3 blur filter depicted in Fig. 41.

S = {(0,20), (0,22), (0,26), (0,27), (0,28), (0,30), (0,31), (0,35), (0,36), (0,37), (20,22), (20,26), (20,27), (20,28), (20,30), (20,31), (20,35), (20,36), (20,37), (22,26), (22,27), (22,28), (22,30), (22,31), (22,35), (22,36), (22,37), (26,27), (26,28), (26,30), (26,31), (26,35), (26,36), (26,37), (27,28), (27,30), (27,31), (27,35), (27,36), (27,37), (28,30), (28,31), (28,35), (28,36), (28,37), (30,31), (30,35), (30,36), (30,37), (31,35), (31,36), (31,37), (34,36), (35,37), (36,37), (1,21), (1,23), (1,25), (1,29), (1,32), (1,34), (21,23), (21,25), (21,29), (21,32), (21,34), (23,25), (23,29), (23,32), (23,34), (25,29), (25,32), (25,34), (29,32), (29,34), (32,34), (2,24), (2,33), (33,24)}.

Generation of initial state matrix and implementation of multi-level encryption on generated state matrix [31]:

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

Table 33
Generated initial state matrix

05	07	0B	0C
57	5B	5C	5D
7C	7D	71	76
CD	C1	16	D1
1A	1E	12	14
64	8A	8E	82
E2	E4	24	23

Table 34
State matrix after bit manipulation (s-box)

6B	C5	2B	FE
5B	39	4A	4C
10	FF	A3	38
BD	78	B4	3E
A2	72	C9	FA
43	7E	19	13
98	69	36	26

Table 35
State matrix after row diffusion

C5	2B	FE	6B
4C	5B	39	4A
10	FF	A3	38
B4	3E	BD	78
FA	A2	72	C9
43	7E	19	13
69	36	26	98

Trifid cipher computation and alphabetic substitution

- Computing TRIFID cipher on "A":

Let IP vendor selected key: EDRFTV$QAWSZMXNCBGYHUJIKOLP
Here, row number (a) is 3, column number (b) is 3, and square matrix (c) number is 1. The state corresponding to "A" is 331. Similarly, the state corresponding to the remaining alphabets is computed based on chosen key.

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

Table 36
Final obtained digit equivalents after alphabetic substitution

Assumed key	Alphabet	Corresponding TRIFID cipher state	Defined rule	Output
100	A	331	(a+b)/c	6
010	B	122	|b+c-a|	3
101	C	222	(a+b) *c	8
011	D	233	|a-b+c|	2
000	E	212	a*b*c	4
001	F	313	a+b+c	7

Generation of multi-level encryption based signature:

Table 37
State matrix after alphabetic substitution

85	23	74	63
48	53	39	46
10	77	63	38
34	34	32	78
76	62	72	89
43	74	19	13
69	36	26	98

Table 38
State matrix after performing transpose

85	48	10	34	76	43	69
23	53	77	34	62	74	36
74	39	63	32	72	19	26
63	46	38	78	89	13	98

- The generated final sequence after byte concatenation is: "85742363484639531077633834783432766289724319137469 362698".
- The generated final signature through the proposed approach is:"1000101111100101111011100100010011011100110111101111111011110000111001111000111001110111110110101000100111110100111100111111111001101001111101011010011000" .
- The generated signature is mapped to its corresponding hardware security constraints as per the IP vendor selected mapping rule (if encoding bit of signature is '0' then embed an edge between (even, even) storage variable pair, otherwise embed an edge between (odd, odd) storage variable pair). The generated hardware security constraints are <Z0,Z2>, <Z0,Z4>, <Z0,Z6>,-------,<Z12,Z16>, <Z12,Z18>, <Z1,Z3>, <Z1,Z5>,----------,<Z7,Z19>, <Z7,Z21>.

Results compared to [13]:

Table 39
P_C and TT comparison between proposed and [13]

Benchmarks	P_C (Proposed)	TT (Proposed)	P_C [13]	TT [13]
Blur filter	3.71E-04	2.28E+46	5.98E-01	4.19E+06
Sharpening filter	1012E-03	8.92E+43	5.72E-01	1.04E+06
Laplace edge detection filter	3.87E-04	8.11E+31	5.52E-01	3.27E+04

Table 40
Area, Latency, Cost, and Resource configuration of proposed hardware security methodology

Benchmarks	Resource configuration	Baseline design (before signature embedding)			Signature embedded design			Design cost overhead %
		Design area (um)	Design latency (ps)	Design cost	Design area (um)	Design latency (ps)	Design cost	
Blur filter	1(+), 1(*)	110.10	1523.58	0.673	110.10	1523.58	0.673	0
Sharpening filter	1(+), 1(*)	111.67	1921.04	0.675	111.67	1921.04	0.675	0
Laplace edge detection filter	1(+), 1(*)	105.38	1258.61	0.722	105.38	1258.61	0.722	0

Part 2 (Preventive Control)

Obfuscation classification tree:

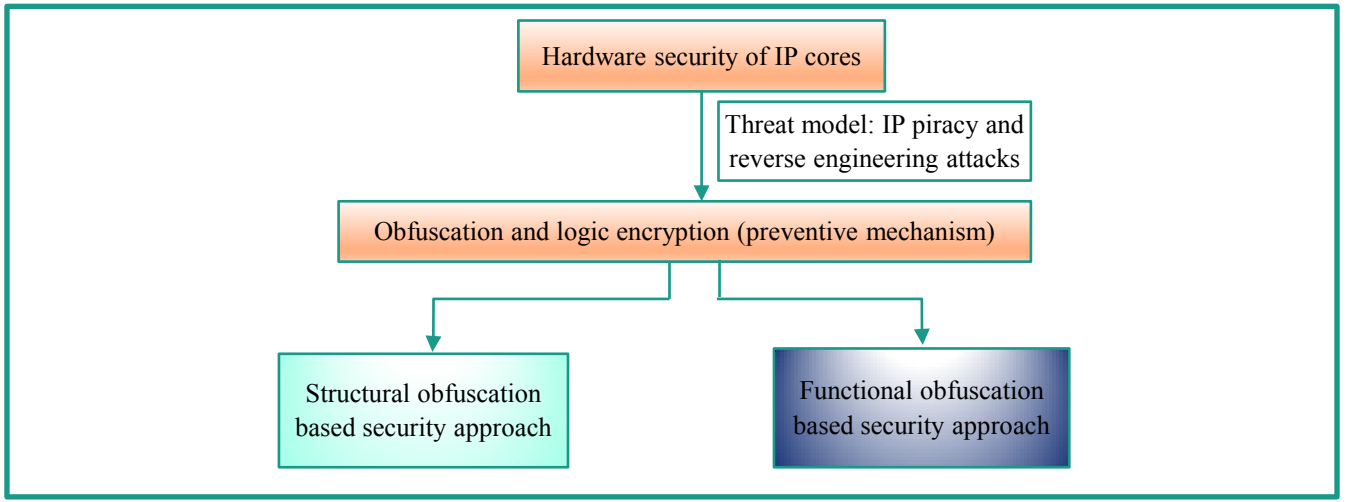


Fig. 42 (a). Taxonomy representation of obfuscation methodologies

Obfuscation (in terms of hardware security):

- Obfuscation is the art of hindering the original structure of the hardware design without affecting its functionality (art of hiding the architecture or description of a design through some modifications in order to cover its functionality). It can be performed either by altering the structure of the hardware design (Structural obfuscation) or with the help of logic locking (Functional obfuscation).

- (Lao and Parhi,2015 [25]; Sengupta et al., 2017 [26]) : Obfuscation is the art of making an object of interest unobvious to be understood by attacker.

- Obfuscation also makes harder for the attacker to reverse engineer as an adversary (attacker) can reverse engineer (back engineer) the design netlist (GDS-II file), masks, packaged non-functional IPs/Ics to extract designer's IP. The adversary can easily pirate, counterfeit and clone the original IP and may also insert malicious logic (called hardware trojans) inside the design after extracting the original design through reverse engineer.

Structural vs Functional (logic) obfuscation :

Comparison perspective	Structural obfuscation	Functional obfuscation
Threat model	Protecting IP cores against piracy, overbuilding, trojan insertion and other RE attacks.	Protecting IP cores against piracy, overbuilding, trojan insertion and other RE attacks.
Objective	To thwart the IP/IC rights violations using structural modifications.	To thwart the IP/IC rights violations using logic locking.
Security Protocol	Modification in structures/ descriptions through and algorithm.	Locking of functionality through key gates.
Abstraction level of deployment	ESL (HLS), RTL, and gate level.	Gate level.
Abstraction mechanism	Same as non-obfuscated design (no additional effort required).	Activation through a valid key.
Vulnerability	Not vulnerable to key-based attacks and removal attack.	Susceptible to key-based attacks and removal attack.
Design overhead	Low (extra hardware may not be required).	Medium (additional key gates are essentially added).

Hardware threats:

Key Sensitization attack : The attacker can extract the keys of a locked (functionally obfuscated) netlist through a key sensitization attack. Attacker need a functional IC (easily obtained through open market) along with obfuscated netlist (may be obtained through RE) for mounting key-based attack. The attacker can sensitize a key-bits at primary output by controlling the primary inputs of the design. Attacker also tries to find out isolated key-gates.

SAT attack : To mount this attack, an attacker needs a locked netlist and an activated/ functional IC. The SAT attack algorithm first generate distinguished I-O pairs using a SAT formula. These distinguished I-O pairs are exploited to eliminate wrong key combinations.

Removal attack : Attacker has access of obfuscated netlist. The attacker attempts to eliminate the additionally inserted key-gates (IP core logic locking) by detecting them using sophisticated algorithm or tools.

Multi-Phase Obfuscation of Fault Secured DSP Designs With Enhanced Security Feature [24]:

- A Novel multi-phase structural obfuscation meyhodology has been proposed.
- The proposed approch proposes the use of multiple high level transformations techniques such as loop unrolling, logic transformation, tree height tranformation, etc. to employ structural transformation in one phase.
- Further, double modular redundancy (DMR) technique has been used to generate fault secured design corrsponding to structurally obfuscated design.
- Next, reconfiguring Mux/DeMux I/Os (cut insertion) has been used to employ obfuscation in phase 2 of multi-phase obfuscation technique.
- Finally, the obtained multi-phase obfuscated design hinders the adversary to gain complete information about the design (Reverese Engineering) in order to insert Trojan at safe places.

[24]. A. Sengupta, S. P. Mohanty, F. Pescador and P. Corcoran, "Multi-Phase Obfuscation of Fault Secured DSP Designs With Enhanced Security Feature," in IEEE Transactions on Consumer Electronics, vol. 64, no. 3, pp. 356-364, Aug. 2018.

Flow chart of the proposed approach [24]:

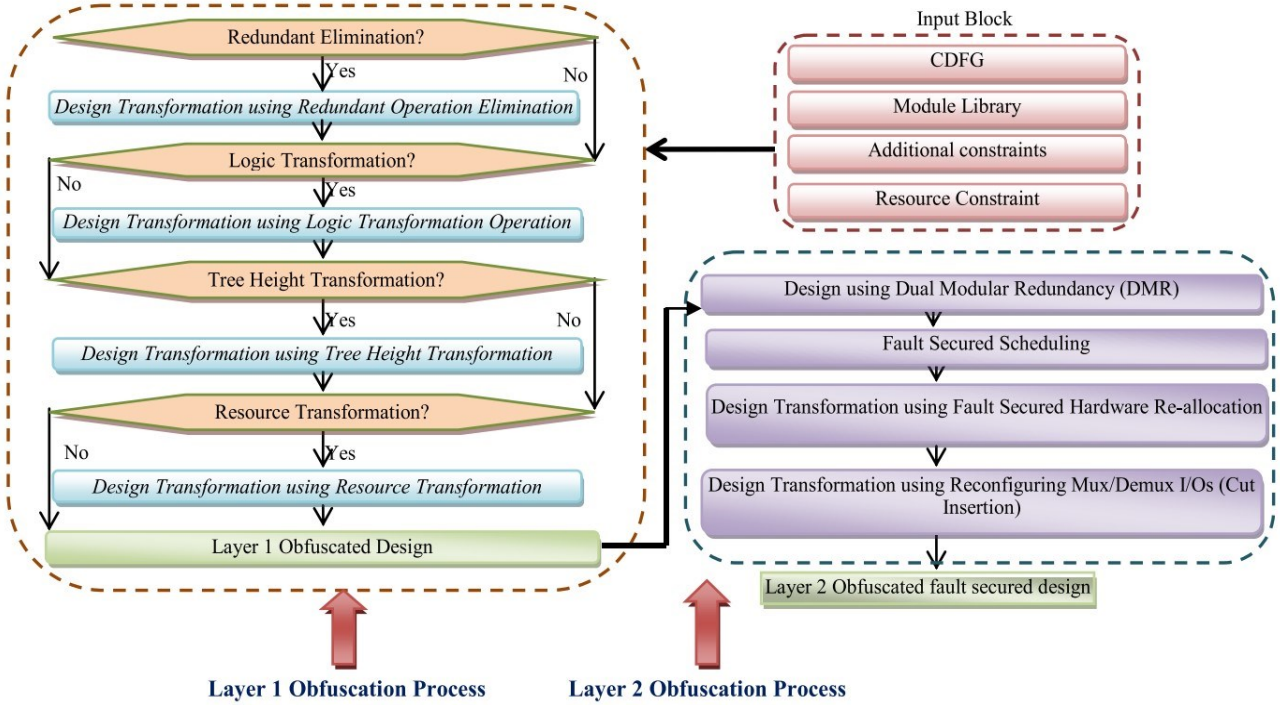


Fig. 42. (b). Flow chart of multi-phase obfuscation based approach

Phase I:

Here, in phase I, structural obfuscation performs **high-level transformations (HLT)** (such as redundent operation elimination **(ROE)**, logic transformation **(LT)**, and tree-height transformation **(THT))**, followed by functional unit transformation, which affects its equivalent gate-level design netlist.

1. **Redundent operation elimination (ROE) :** It involves elimination of redundant nodes (representing operations) in a CDFG of a DSP application without affecting the functionality of the design.

The nodes having operation and inputs same as other nodes are termed as redundant nodes. All such redundant nodes are enlisted and at last all the nodes except the node having least operation number are eleminated.

This elimination results into the changes in the inputs of the remaining operator. However, this elimination of the redundent nodes and corresponding changes in the inputs of remaining nodes should not affect the functionality of the design. This transformation affects the gate-level design netlist in terms of interconnectivity and circuit structure, while preserving the functional behaviour.

Demonstration of HLT- Redundant operation elimination (ROE):

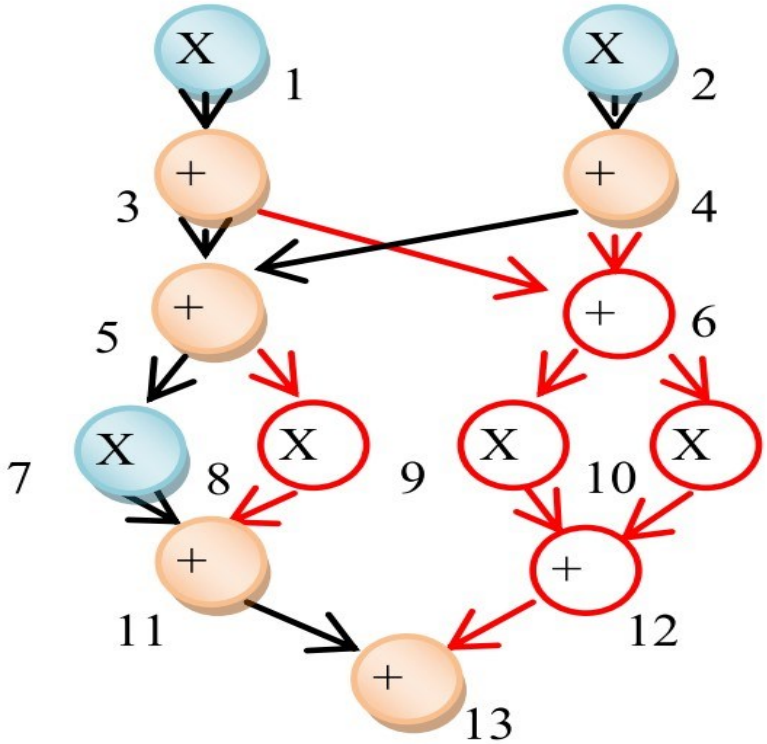


Fig. 43. Redundant operation elimination

Phase I :

2. Logic tranformation (LT) : It is the process of altering nodes logically in a control data flow graph (CDFG) in order to introduce some structural changes without affecting the functionality.

Sometimes, it also leads to increase in nodes in graph, ultimatly affecting the gate-level design netlist in terms of interconnectivity and circuit structure, while preserving functional behaviour.

3. Tree height transformation (THT) : In this process, the height of the CDFG is intentionally reduced to modify the tree structure. First the critical path dependency is divided into temporary sub-computations. These subicomputations are executed in parallel, while ensuring the correct functionality of the design.

Demonstration of HLT-LT and THT:

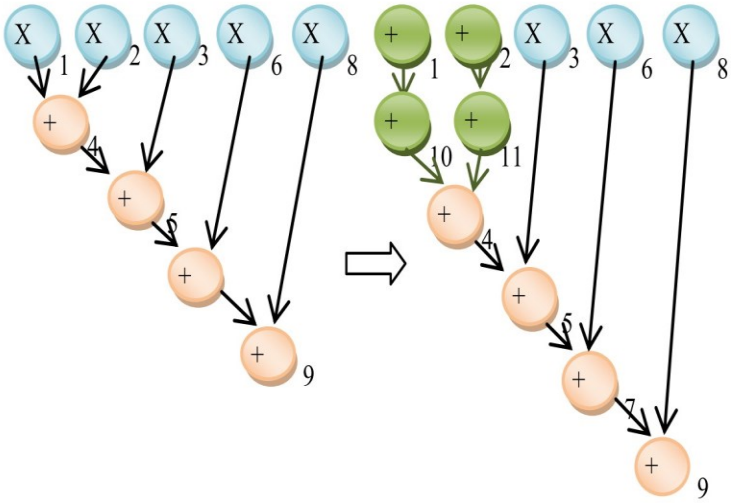


Fig. 44. Logic transformation

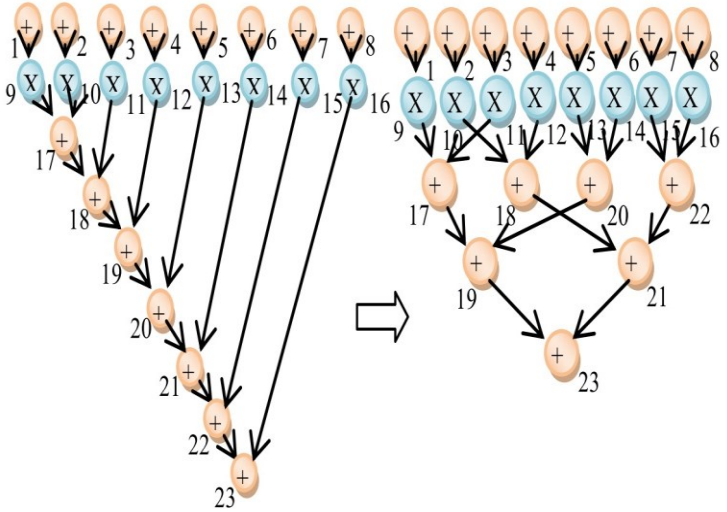


Fig. 45. Tree height transformation

Phase I (Contd.):

4. Resource transformation/ functional unit transformation (RT/ FUT) : It is basically amalgamation of operator, in which two Fus (operations) are merged into a single unit. This newly generated FU represents a customised resource that has a new input configuration and mux/demux connectivity. The custom-function unit is denoted as “$”. It can only be performed for the nodes having independent inputs and whenever an addition operation is followed by a multiplication operation in CDFG.

Demonstration of RT:

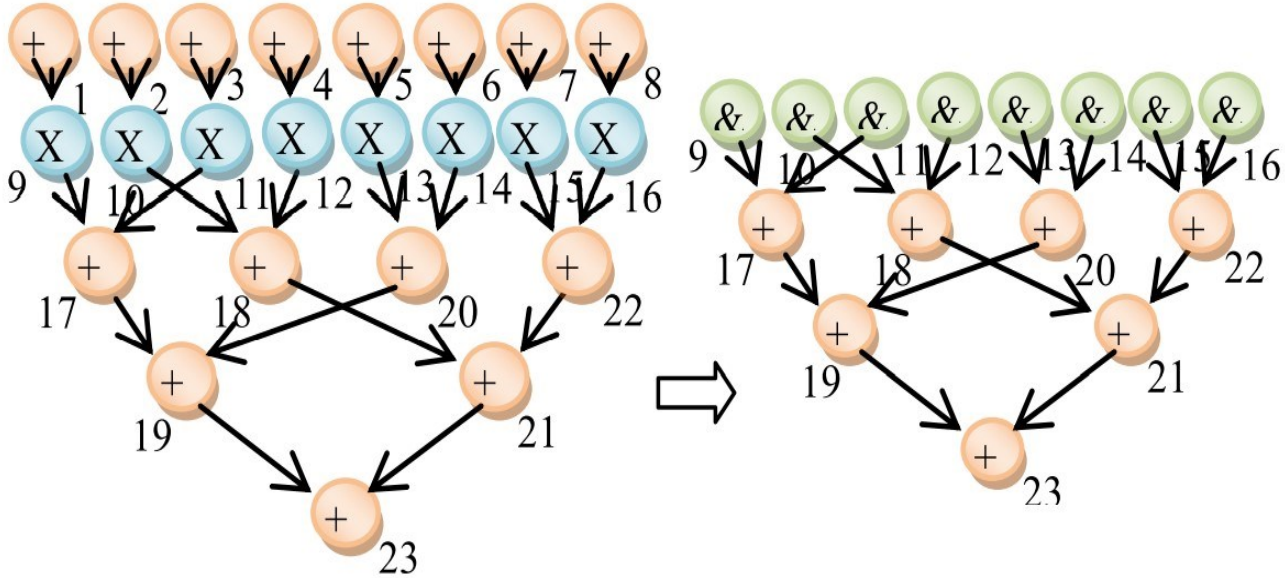


Fig. 46. Resource transformation

Phase II (generating fault secured hardware design):

To protect the hardware from transient faults (upset the device momentarily (transiently)), it is necessary to make the hardware design fault secured. It is achieved with the help of double modular redundancy (DMR).

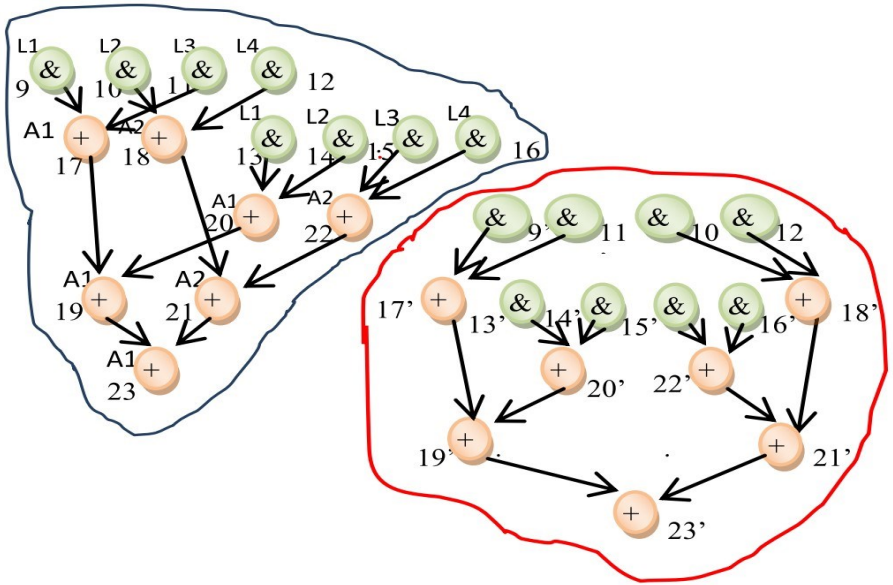


Fig. 47. Obfuscated DMR design

Phase II:

Now, in phase II, obfuscated fault-secured DMR design is again obfuscated using mux/demux reconfiguration and additional comparator hardware.

Cut-insertion technique is applied on fault secured DMR design for this. Cuts are applied on some data dependency edges in the graph which leads to the shifting of the corresponding operations by at least one control step upward with respective changes in operation inputs. **This shifting of operations results in FU reallocation, mux/demux reconfiguration and comparator hardware insertion.**

Further, through cut insertion, multiple checkpoints are deployed in the graph to enhance the fault security. **Additionally, it also optimizes the delay overheads caused due to fault secured DMR design.**

It also results in nominal area overhead, the reduction in delay overheads (due to cut insertion) balances the overall design cost.

It also leads to multiple changes in interconnection = more RTL modifications.

Delay optimization = Shift of operation upside.

Demonstration of reconfiguring mux/demux architecture using cut insertion:

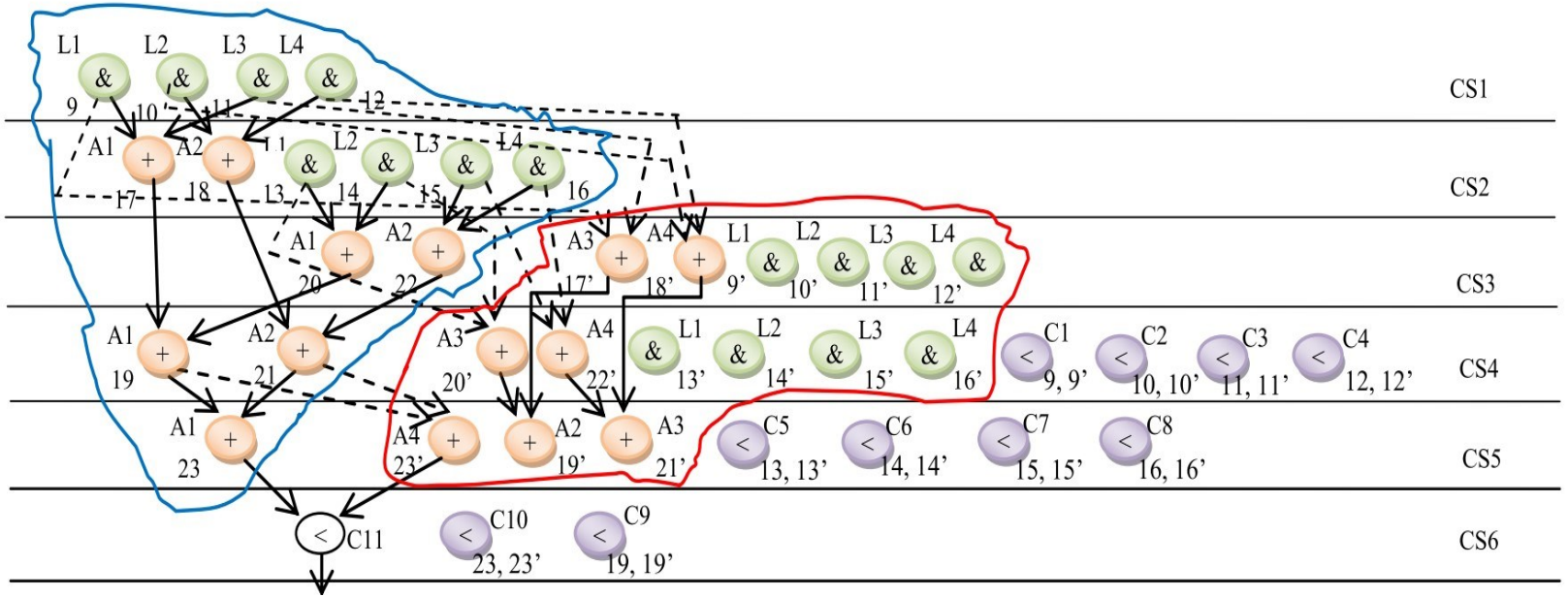


Fig. 48. Obfuscated Fault Secured DSP design of FIR filter (with enhanced fault security feature – multi-cuts).

Security metric:

➢ **Evaluation of Robustness Using Probability of Coincidence (Pc):**

$$SoO = \sum_{i=1}^{m} (Number\ of\ unique\ nodes\ modified)\ /\ (Number\ of\ nodes\ before\ obfuscation)$$

❑ The properties of the modified nodes are as follows:

a) The primary input of a node or parent node is different than its original in an obfuscated CDFG.

b) The child node is different than its original in an obfuscated CDFG.

c) In an obfuscated CDFG, the operation type (such as addition, multiplication, etc.) of a node is changed.

d) In an obfuscated CDFG, a node of original CDFG is non-existent.

Results:

Benchmark	Proposed approach						[28]	Enhancement in Strength of Obfuscation (%)
	REO (Unique nodes)	LTO (Unique nodes)	THT (Unique nodes)	ALU (Unique nodes)	# of gates affected	SoO		
IIR	-	3	3	-	3040	0.666667	0.33333	100
ARF	10	6	-	-	2400	0.571429	0.42857	33.33
BPF	5	6	2	2	960	0.517241	0.44827	15.38
DWT	-	10	-	-	6688	0.588235	0.52941	11.11
FIR	-	-	12	11	4288	1	0.5	100

Table 25. Comparison of proposed obfuscation with [28] in terms of obfuscation strength.

Protecting DSP Kernels Using Robust Hologram-Based Obfuscation [29]:

- A novel structural obfuacation approach based on hologram technology.
- **Hologram:** A hologram is a special feature whereby multiple images can be overlapped with each other, while simultaneously being capable of being displayed at the same time. The overlapping process is similar to concept of camouflaging. At different angle of viewing, distinct images evolve through switch elements (Matrix technology, Hologram feature, 2016). This phenomenon can be harnessed for performing structural obfuscation of DSP circuits.
- **Hologram based obfuscation has been formulated such that at least two DSP applications are integrated together in camouflaged fashion. These camouflaged DSP applications (in circuit form) are possible to be viewed and activated at a specific switching values.**
- Hologram based obfuscated designs enhance the complexity of understanding the functionality of the architecture, thus hindering RE attacks. Additionally, it also satisfies the property of minimal design overhead.

[29]. A. Sengupta and M. Rathor, "Protecting DSP Kernels Using Robust Hologram-Based Obfuscation," in *IEEE Transactions on Consumer Electronics*, vol. 65, no. 1, pp. 99-108, Feb. 2019.

Overview hologram based obfuscation [29]:

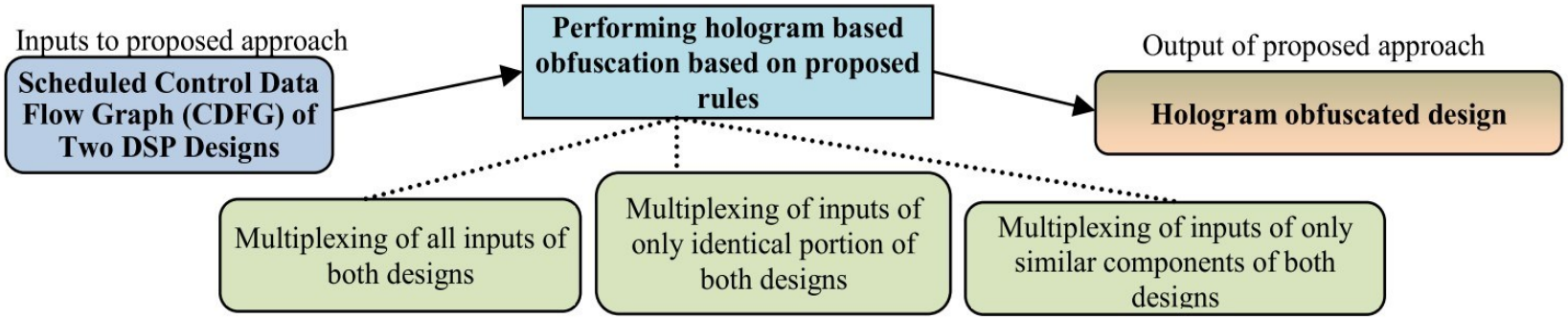


Fig. 49. Overview of hologram based obfuscation approach.

Flow diagram hologram based obfuscation [29]:

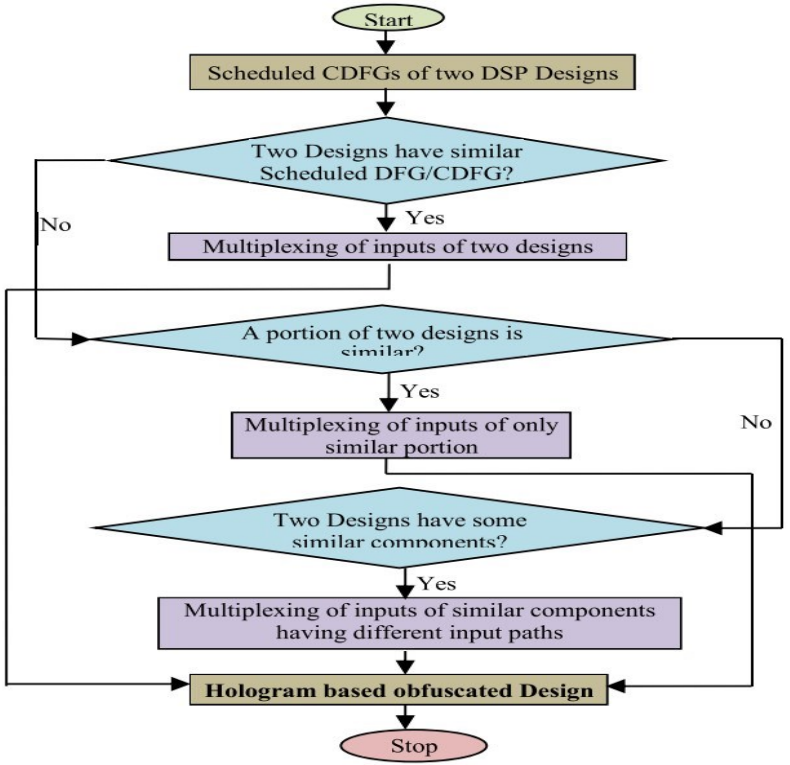


Fig. 49. Flow diagram of hologram based obfuscation approach.

Proposed approach [29]:

- DSP cores such as FIR, IIR, DCT, IDCT, and FFT are widely used in electronics systems to facilitates several applications, such as video /audio processing, image processing and de-noising, etc.
- A combination of similar application such as FIR-IIR, DCT-IDCT, and similar ones (whose architecture are identical) are used to perform hologram based obfuscation, by integrating their architecture into a single path.

❑ **Rule 1 :** Multiplexing sub-set of inputs of two DSP application.

If a portion of the SDFG of two different DSP applications is structurally identical (**identical portions of both applications require similar resources with similar input output connectivity**).

i. The input of only similar portions of both applications are multiplexed.
ii. The output of both application are multiplexed.
iii. The switching between two designs is performed using 2:1 multiplexer.

Proposed approach (Cont.) [29]:

❑ **Rule 2 :** Multiplexing all of inputs of two DSP application.

If two SDFG representing two different DSP applications are identical (**both having similar input-output interconnectivity**).

i. All inputs of both applications are multiplexed.

ii. At S=0, one DSP design gets functionally enabled and at S=1, functionality of other DSP design gets enabled.

❑ **Rule 3 :** Multiplexing of inputs of similar components in two DSP application.

This rule is applied when two application have some similar component in their datapath. However, the paths of inputs of those components are different. When this condition is satisfied, multiplexing of inputs of only similar components of both applications is performed. **Components of one application accepts inputs at S=0 while of other accepts at S=1.**

Case study of 4-point DCT-IDCT obfuscated design :

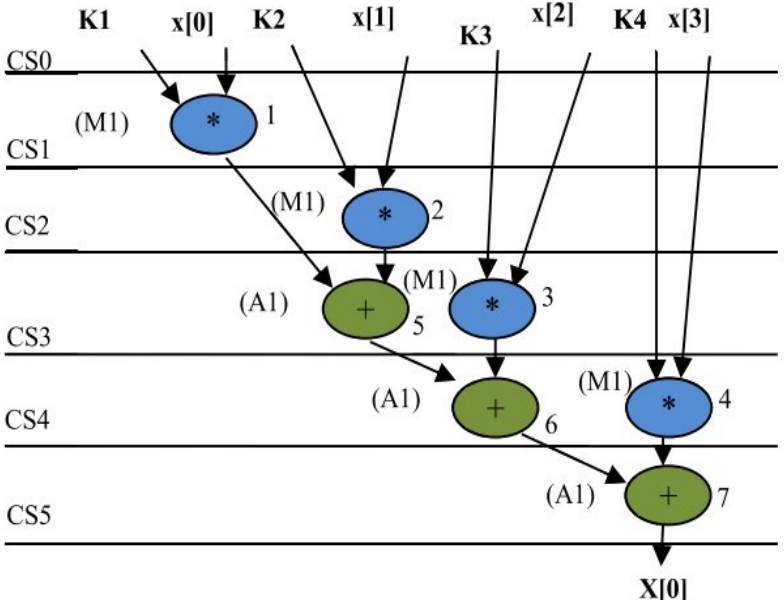


Fig. 50. Scheduling of 4-point DCT based on 1 adder and 1 multiplier.

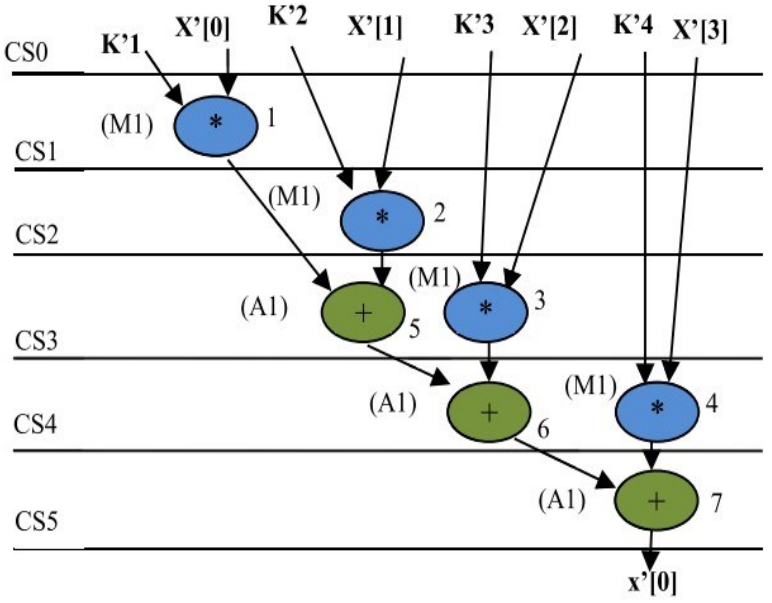


Fig. 51. Scheduling of 4-point IDCT based on 1 adder and 1 multiplier.

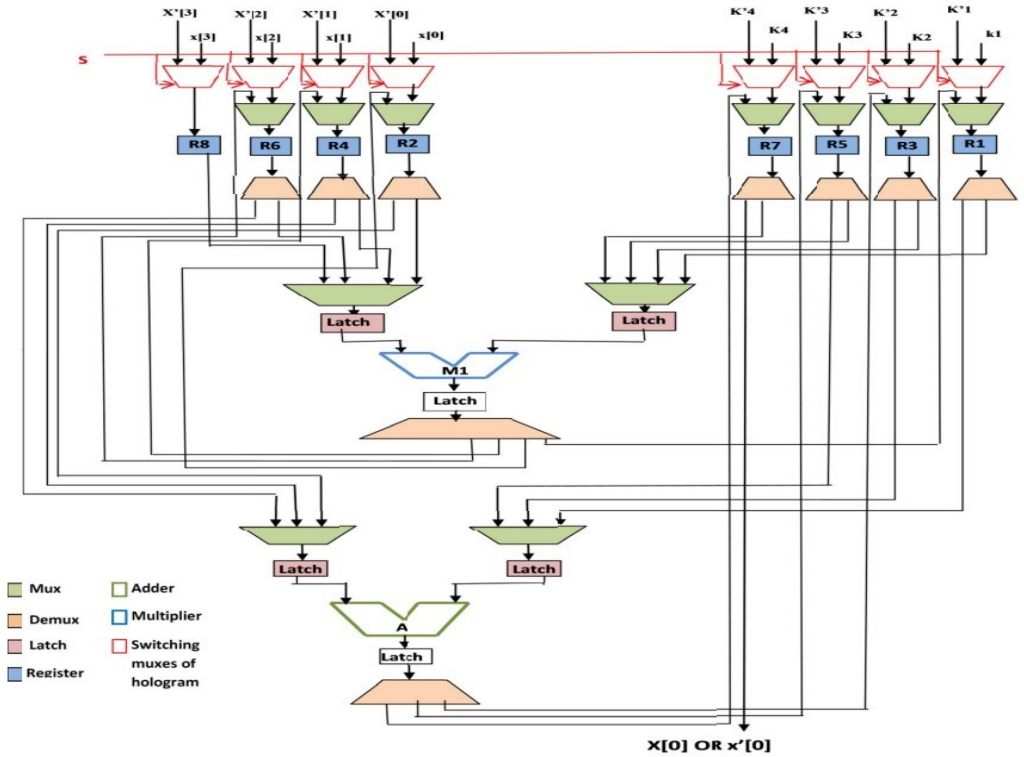
Case study of 4-point DCT-IDCT obfuscated design :

$$\textbf{4-Point DCT} : X[0] = k1 * x[0] + k2 * x[1] + k3 * x[2] + k4 * x[3]$$

$$\textbf{4-Point IDCT} : x'[0] = k'1 * X'[0] + k'2 * X'[1] + k'3 * X'[2] + k'4 * X'[3]$$

- where x[0], x[1], x[2], and x[3] denote the input samples and X[0] denotes the first output value of 4-point DCT whereas k1, k2, k3, and k4 are the generic values of coefficients of 4-point DCT. Similarly where X'[0], X'[1], X'[2], and X'[3] denote the input samples and X'[0] denotes the first output value of 4-point IDCT whereas k'1, k'2, k'3, and k'4 are the generic values of coefficients of 4-point IDCT.

Case study of 4-point DCT-IDCT obfuscated design :



- Here, hologram based obfuscation is employed based on onfuscation rule-2 because structurally a 4-point DCT is identical to a 4-point IDCT. Input of both designs are multiplexed with the help of 2:1 muxes.
- Similarly, in the case of FIR-IIR, rule I is applicable.

Fig. 52. Hologram based obfuscated RTL datapath of proposed 4point DCT- 4point IDCT.

Determination of gate count affected due to hologram based obfuscation :

- The process of estimating the gate count of the hologram-obfuscated design and the high level transformation-obfuscated design can be generically represented below formula.
- In case of THT (as it only affects interconnectivity), difference in gate count is 0, and it only targets minimizing the critical delay of the application.

$$G_C^a = \#\ of\ gates\ affected\ = (difference\ in\ gate\ count\ wrt\ baseline)\ + (\#\ of\ gates\ changed\ in\ terms\ of\ input\ connectivity)$$

$$S^{OB} = G_c^a / G_C^t$$

S^{OB} indicates strength of obfuscation in terms of affected (modified) gate count, G_C^a = # of gates affected , and G_C^t indicates total # of gates in an unobfuscated (baseline)design.

Results:

DSP kernels	# of gates affected (Proposed obfuscation)	# of gates affected [30]	Affected gate count (Proposed obfuscation)	Affected gate count [30]
IIR+FIR	2704	224	29.2 %	2.4%
8-pt DCT+ 8-pt IDCT	49792	3584	45.5 %	3.5%
4-pt DCT+ 4-pt IDCT	9024	512	45.1%	2.0%
8-pt DCT + 4-pt DCT	12096	2048	18.1%	3.2%
8-pt DIT-FFT + 4-pt DIT-FFT	4160	0	29.7%	0%

Table 26. Comparison of protection capability between proposed obfuscation and [30].

Low Cost Functional Obfuscation of Reusable IP Ores Used in CE Hardware Through Robust Locking [23]:

- A novel functional obfuacation approach based IP locking blocks is discussed.
- The discussed functional obfuscation based security approach is employed in two blocks: (i) PSO-based optimization block and (ii) logic encryption block.
- During gate level synthesis, the ILBs are inserted into the design at different appropriate locations. Post insertion of ILBs, the design is resynthesized, and a functionally obfuscated (or logically encrypted) netlist design file is generated.
- ILBs are reconfigured based on the 128-bit custom lightweight AES.
- Use of custom 128-bit AES and reconfigured ILBs in proposed approach [31] provides immunity against removal and key based attacks.
- The DSP designs used in the proposed work are SAT attack resistant. It is observed that even for a small size multiplier, the resulting CNF is quite large (e.g., 20 clauses for a 3 bit multiplier [32]).

[23]. A. Sengupta, D. Kachave and D. Roy, "Low Cost Functional Obfuscation of Reusable IP Cores Used in CE Hardware Through Robust Locking," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 4, pp. 604-616, April 2019.

Flow chart of the proposed approach [23]:

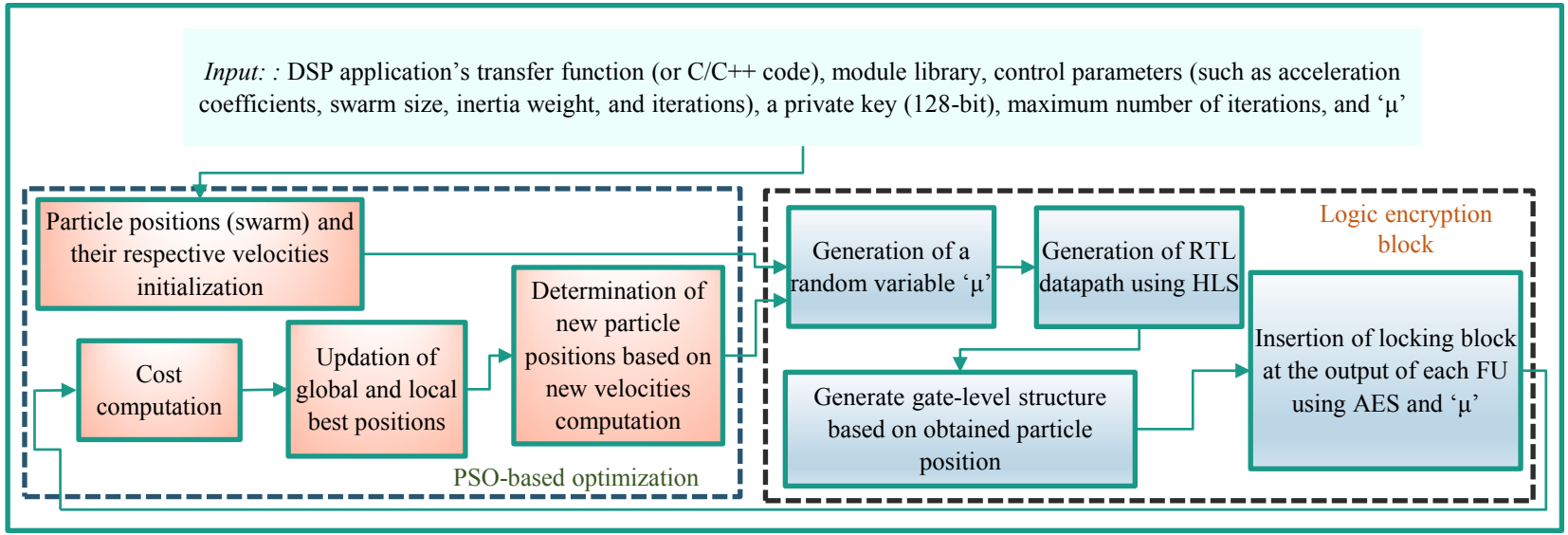


Fig. 53. Flow chart of the low-cost logic locking approach [23].

Security of functionally obfuscated DSP cores:

- Functional obfuscation is a mechanism of obscuring the functionality of a design by locking it logically through some key gates. As the secret key is not known to the attacker in advance, he/she is unable to use IP core illegally. **More explicitly, an attacker cannot pirate, counterfeit, clone or overbuild an IP core, thus being incapable of inserting hardware trojans.**
- Due to insertion of locking IP core blocks using key gates, key-based attacks therefore becomes automatically applicable. Additionally, an attacker may also attempt to remove or deactivate the key gates.
- The functional obfuscation of DSP cores is performed by inserting a certain number of IP core locking blocks (ILBs) at appropriate locations into the design.
- The structure of an ILB is constituted using a combination of basic XOR, XNOR, AND, NAND, OR, and NOT gates.

Using custom AES-128 –based ILB key generation hardware:

- 128-bit input and 128-bit key is provided as input to AES block and based on the corresponding output ILB are reconfigured each time. So, it provides resistance against removal attack (as no such fixed architecture is there).
- Since the attacker is unaware of any fixed ILB architecture, detection of ILB in the complex DSP netlist is thus highly challenging. And the used AES block is also custom AES, so detection of AES post synthesis in design netlist is itself challenging.
- AES makes use of non-linier logic responsible for bit-manipulation using S-box. Block ciphers generally comprises of SPN (substitution and permutation network), which is very hard to break.
- The security technique using custom AES-128 bit is a relatively heavyweight solution against removal attack. With the help of 4 instances of custom AES-128 units, we can reconfigure up to 64 ILBs (each requiring 8-bit key).

Security features of ILBs used in functionally obfuscated DSP cores:

1. **Multi-pair wise security :** The key-bits of an ILB are multi-pair wise secured because a single key can not be sensitized to o/p without knowing/controlling the other key bits of the same ILB. This is because the path of the sensitization of a key-bit is interfered by the other multiple key-bits.
2. **Prohibiting key gate isolation :** A shown in previous figure (ILB) structure, a path essentially exists between any two key-gates.
3. **Resilient to muting of key gates :** The key gates in the ILB structure are intertwined in such a fashion that the effect of any of the key gate cannot be muted. Hence, the attacker's attempt of sensitizing of the key-bits to output by muting the effect of remaining key-gates is thwarted.
4. **Resilient to run of key-gates :** As depicted in previous figure (in ILB structure), there is no such sequence of key-gates that can be substituted by a single key-gate. Hence, the attacker's attempt to sensitize a key-bit by increasing the valid key-space is thwarted.

ILB representation [23]:

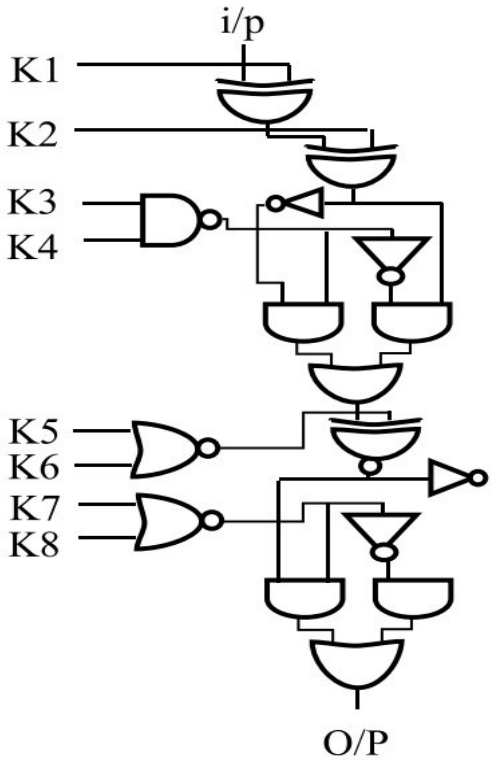


Fig. 54. Sample ILB.

Security metric [23]:

$$O_S = (2)^z$$

The strength of obfuscation O_S indicates the logic encryption key space and 'z' is a measure of the following *q*l*g*; where *q*= key bits per ILB, *l*= number of ILBs per FU, *g*= number of FUs in the datapath.

Result:

DSP Core Benchmarks [33]		No. of key-bits encoded for proposed obfuscation (r)	Strength of obfuscation of proposed approach (using eq. 2)	No. of key-bits encoded for [33]	Strength of obfuscation of [33] (using eq. 2)	Strength of obfuscation enhancement of proposed approach (by factor of)
Name	**Size**					
IIR	**9919**	192	6.28 e+57	96	7.92 e+28	7.92 e+28
Mesa Horner	**10842**	192	6.28 e+57	80	1.2 e+24	5.19 e+33
DWT	**10958**	128	3.40 e+38	96	7.92 e+28	4.29 e+ 9
ARF	**14833**	256	1.15 e+77	112	5.19 e+33	2.23 e+43
FIR	**16047**	320	2.13 e+96	144	2.23 e+43	9.57 e+52
JPEG IDCT	**42710**	1344	3.83 e+404	432	1.10 e+130	3.46 e+274
Mesa Interpolate	**48853**	832	2.86 e+250	464	4.76 e+139	6.01 e+110

Table 27. Strength of obfuscation comparison of proposed functionally obfuscated approach with respect to [33].

On improving the security of logic locking [33]:

- A novel functional obfuscation approach based on key-based logic locking methodology with improved security is discussed.
- In order to perform functional obfuscation, logic locking is performed using **non-mutable** (means muting one of the key gates will not help in sensitizing the output of the key value) **XOR-XNOR key gate pairs.**
- A judicious location selection technique to insert the first key gate in the IP core design has also been discussed.
- The careful selection of target locations for the insertion of additional key gates makes it hard for an adversary to easily identify and eliminate the inserted key gates even after carefully inspecting the design netlist file.

[33]. M. Yasin, J. J. Rajendran, O. Sinanoglu, and R. Karri, "On improving the security of logic locking," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst*, vol. 35, no. 9, pp. 1411–1424, Sep. 2016.

Security of functionally obfuscated DSP cores :

- **Protection against SAT based attack:**
  - In order to protect the logic encryption from SAT-based attacks, one-way random functions (such as AES with a secret key) have been used to generate the key for the logic locking block.
  - This increases the adversary's efforts in terms of guessing the key value as they also have to guess the IP designer selected AES secret key.
  - Further, to protect the logically encrypted design from removal attack, the AES block has been synthesized such that it is different from the public AES block.
- **Protection against removal attack:**
  - Further, to protect the logically encrypted design from removal attack, the AES block has been synthesized such that it is different from the public AES block.

Advantage over PUFs [33]:

- Block cipher such as AES has been used to provide additional security (thwarting SAT-based attacks) to inserted XOR-XNOR key gate pairs instead of physically unclonable functions (PUFs) [35] because of the following reasons:

a) PUFs are prone to modeling attacks [36], and
b) using block ciphers incorporate various well-known cryptographic properties and also protects against modeling and cryptanalysis attacks.

- Further, to improve execution time, all necessary conditions regarding pairwise security of keys are developed and listed to remove unnecessary test conditions for security.

Overview of the logic locking approach:

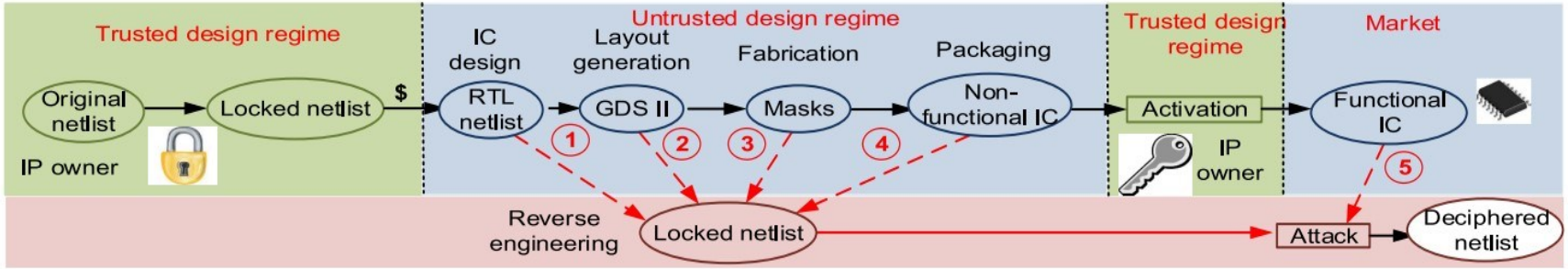


Fig. 55. The design is in the locked form in the untrusted design regime. In the untrusted regime, an attacker can obtain the locked netlist from (1) the IC design, or by reverse engineering the (2) layout, (3) mask, or (4) a fabricated IC, and (5) the functional IC from the market. Using this attack, the attacker can get a deciphered netlist and make pirated copies.

Judicious location selection technique:

- A logic locking algorithm must insert key-gates that are pairwise secure with respect to one another. The first key-gate inserted in a netlist is important as it affects the insertion of the subsequent key-gates in an iterative fashion.
- In the notion of pairwise security, two gates are good candidates to be pairwise secure when they are convergent with respect to one another.
- **Pairwise Security:** We say that key-gates *K1* and *K2* are pairwise secure if an attacker cannot sensitize key-bit K1 to a primary output (PO) without knowing or controlling the value of key-bit K2 and vice versa.
- Convergent Key-Gates: Even if there are no paths between two key-gates, the sensitization paths might interfere. Such scenarios happen if these two or more key-gates converge. Depending upon the type of convergence, key-gates can be classified into: 1) concurrently mutable; 2) sequentially mutable; and 3) nonmutable key-gates.

Security features of ILBs used in functionally obfuscated DSP cores :

1. **Concurrently mutable convergent key-gates:** If two key-gates *K1* and *K2* converge at some other gate, such that value of key-bit *K1* can be determined by muting *K2*, and key-bit *K2* can be determined by muting *K1*, then *K1* and *K2* are called concurrently mutable key-gates.

Example: Consider the circuit shown in Fig. 56(a). The key-gates *K1* and *K2* converge at the gate G5. The value of key-bit *K1* can be determined by applying a pattern that mutes *K2 (B = 0)*. Similarly, the value of key-bit *K2* can be determined by applying a pattern that mutes *K1 (A = 1)*.

2. **Sequentially mutable convergent key-gates:** If two gates *K1* and *K2* converge at some other gate, such that key-bit *K2* can be determined by muting *K1* while *K2* cannot be muted to determine key-bit *K1*, then *K1* and *K2* are called sequentially mutable convergent key-gates, as they can be deciphered only in a particular order.

Example: Consider the circuit shown in Fig. 56(b). The value of key-bit *K2* can be determined by applying a pattern that mutes *K1 (A = 1),* while *K2* cannot be muted as it directly feeds the gate where *K1* and *K2* converge.

ILB representation [31]:

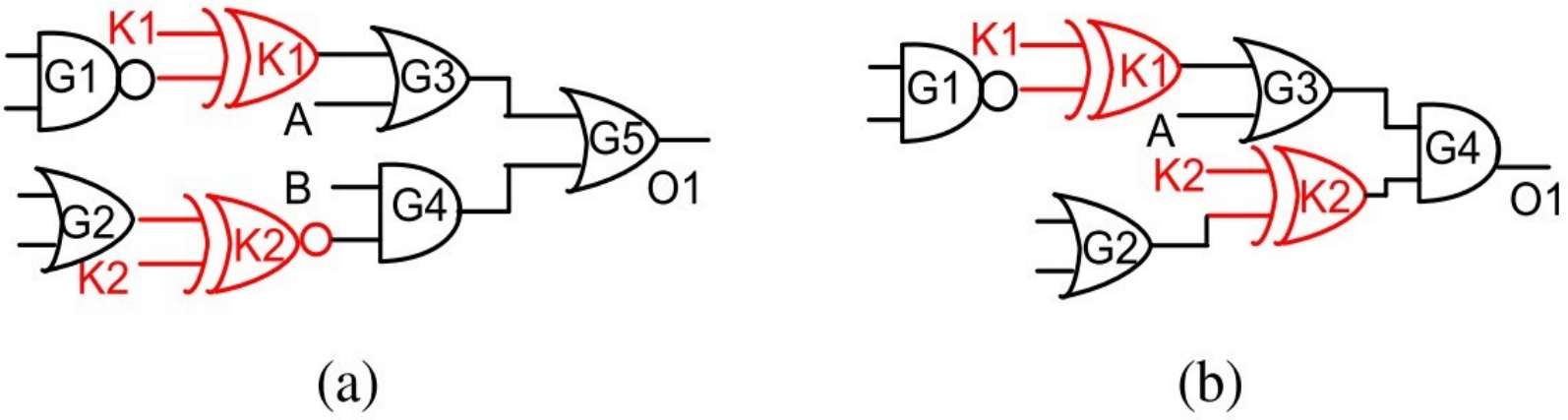


Fig. 56. (a) Concurrently mutable key-gates: K1 and K2 converge at G5 and can be muted. (b) Sequentially mutable key-gates: K1 and K2 converge at G4, but only K1 can be muted.

Security features of ILBs used in functionally obfuscated DSP cores :

3. Nonmutable convergent key-gates: If two key-gates *K1* and *K2* converge at some other gate such that neither of the key-bits can be muted, then *K1* and *K2* are called nonmutable convergent key-gates.

Example: Consider the circuit shown in Fig. 57. The key-gates *K1* and *K2* are connected to the same gate G4 and are nonmutable convergent key-gates. Here, only brute force attack is possible.

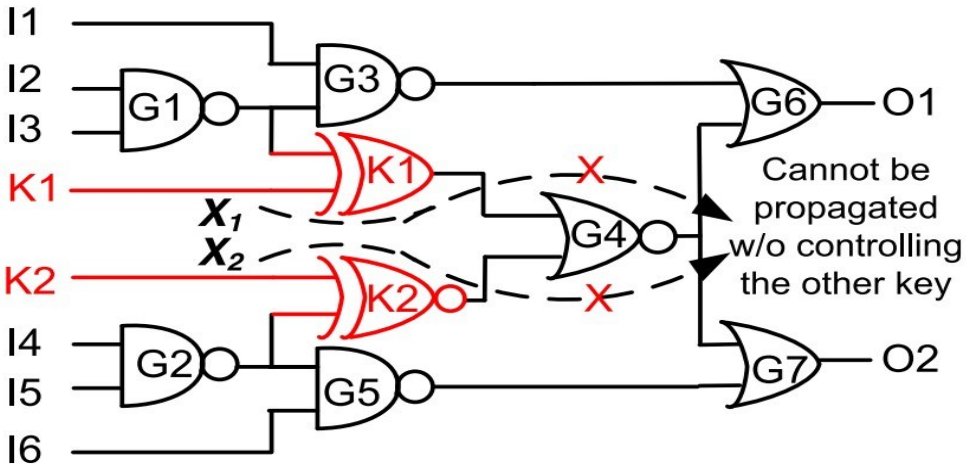


Fig. 57. Attacker cannot sensitize the key-bits K1 and K2 individually to the outputs. He/she has to brute force to determine the values of K1 and K2..

Result:

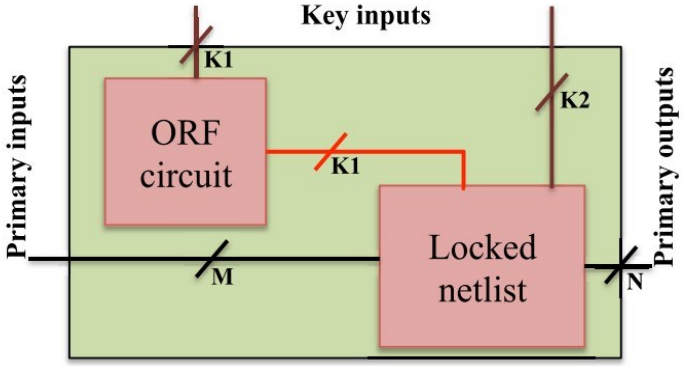


Fig. 58. ORF-based (AES) countermeasure against SAT-based attacks. K1 out of K key-inputs in the locked netlist are connected to the ORF circuit.

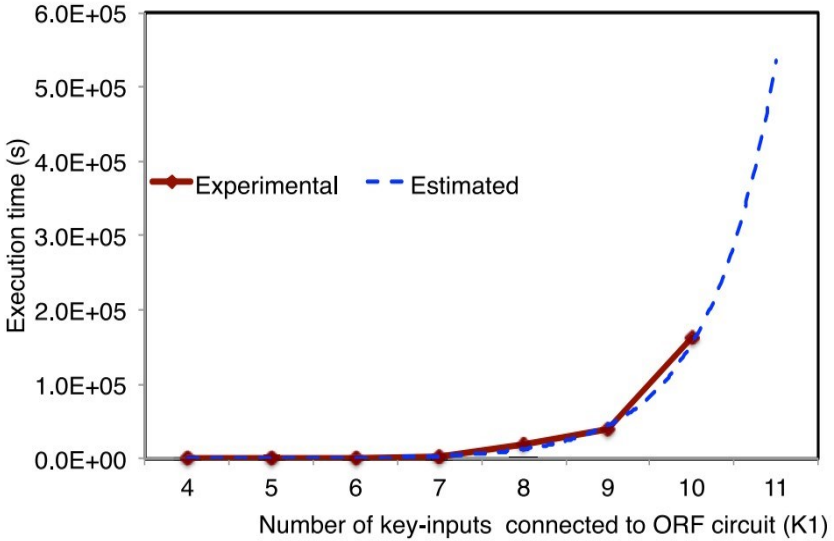


Fig. 59. Time required by the SAT-based attack [19] to break the ORF+c7552 circuits for different key-inputs connected to the ORF circuit (K1). The blue line represents the best-fitting exponential curve to the experimental data.

High-Level Synthesis of Key-Obfuscated RTL IP with Design Lockout and Camouflaging [34]:

- A structural obfuscation approach integrated with design lockout and camouflaging mechanisms to generate key-obfuscated RTL design is discussed.
- The complete algorithm, which is used to generate the obfuscated RTL designs with lockout and camouflaging capabilities, is divided into three blocks: (a) obfuscation aware HLS block, (b) design lockout block, and (c) camouflaging block.
- In the first block, different structural obfuscation technicques are incorporated to produce a structurally obfuscated design after scheduling.
- Further, in second block, a design lockout mechanism is employed using counter and checker finite state machine (FSM).
- Finally, last block performs design camouflaging using MUXs and XOR gates.

[34]. Sheikh Ariful Islam, Love Kumar Sah, and Srinivas Katkoori. 2020. High-Level Synthesis of Key-Obfuscated RTL IP with Design Lockout and Camouflaging. *ACM Trans. Des. Autom. Electron. Syst*. 26, 1, Article 6 (January 2021), 35 pages.

Flow chart of the proposed approach [34]:

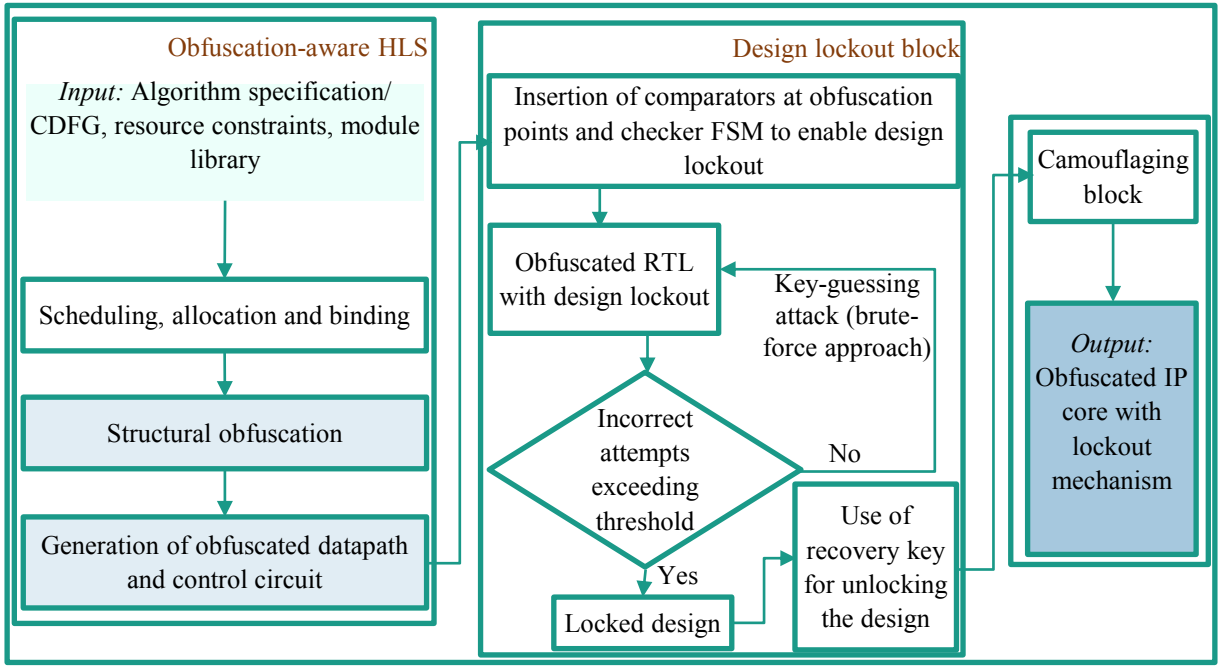


Fig. 60. Flow chart of the hardware obfuscation approach with design lockout and camouflaging mechanism [34].

Details of the proposed approach [34]:

- **Block 1 (Structural obfuscation block):**

- First, the CDFG is scheduled based on designer-specific resource constraints, followed by allocation and binding to generate the scheduled DFG (SDFG). Next, structural obfuscation is performed on the SDFG to generate obfuscated SDFG.
- The operations with the highest mobility on a non-critical path are listed. Non-critical operations in an SDFG are operations that can be shifted in upward or downward control steps without violating data dependency and functionality. The shifting of non-critical operations (either in upward or downward control steps) is performed accordingly.
- Further, different algorithmic transformations (e.g., associative ←→ commutative and distribution ←→ commutative) are performed along with an exchange between low-complexity operations (such as shifting) and high-complexity operations (such as division and multiplication).
- After applying all structural transformations, an obfuscated RTL datapath is generated along with the control circuit.

Details (Contd.):

❑ **Block 2 (Design lockout block):**

- In the second (design lockout) block, comparators and checker finite state machines (FSM) are inserted at various obfuscation points to enable design lockout mechanisms.
- Glushkovian model (a multiplexer (mux)-based key with a combination of XOR gates) is used along with a counter (to check the number of total incorrect attempts).
- The design is locked on exceeding the wrong attempts to unlock the design.
- Even if an adversary manages to find the general structural pattern (the combination of mux and XOR gates), it is still impossible for them to access any information about the key using the general structural pattern.
- The lockout mechanism works in two phases: (a) partial lockout and (b) complete lockout. The design will be partially locked on entering a wrong key combination, and further, it will be fully locked on exceeding the correct attempts threshold.
- A recovery key is used to unlock the design in case of a completely locked design.

Details (Contd.):

- **Block 3 (Design camouflaging block):**

- In the third block of design, camouflaging at the layout level is performed.
- To an attacker, camouflaged cells are designed to look alike, and they disguise actual functionality. In modern design, there exists a large number of gates and connections that can be camouflaged to prevent reverse engineering.
- The selection of appropriate gates at the layout level for camouflaging is non-trivial to obfuscate the design in acceptable power and area overhead limit (for example, the power and area overhead is very high in the case of [37]).
- Two basic blocks: (i) Mux in the obfuscated RTL and (ii) XOR in design lockout, have been used to perform camouflaging. Only a valid configuration bit will help to choose the correct block in a camouflaged cell (containing four different functionalities).

Camouflaging [34]:

Configuration Bit	Block	Function	Comment
"00"	MUX	$AS + B\overline{S}$	Regular MUX & Obfuscation Module
"01"	XOR	$A\overline{B} + \overline{A}B$	Regular XOR
"10"	MUX+XOR1	$\overline{S}(A\overline{B} + \overline{A}B)$	Output = 0 for correct key
"11"	MUX+XOR2	$S(A\overline{B} + \overline{A}B)$	Output = 1 for correct key

S, selector; A, correct input; B, incorrect input.

Table 28. Specification of Camouflage Cell.

- **For example**, when correct input is present as one input of XOR in the compound block (MUX + XOR1), XOR output should be 0 for correct key propagation and vice versa for incorrect input at one input of XOR. Table 3 shows the configuration bit of four blocks and their functionality.

Result:

Benchmark	No. of Registers	No. of Resources (+, A; *, M)	No. of Resources [23] (+, A; *, M)	Obf. Strength	Obf. Strength [23]	Obf. Enhancement (Factor of)	Attack Time (Years)	Attack Time (Years) [23]	Attack Time (Enhancement)
IIR	9	(1+, 2*)	(1+, 2*)	8.47e+111	6.28e+57	1.35e+54	2.69e+95	1.99e+41	1.35e+54
Mesa Horner	12	(1+, 1*)	(1+, 2*)	6.22e+115	6.28e+57	9.91e+57	1.98e+99	1.99e+41	9.93e+57
DWT	11	(2+, 2*)	(1+, 1*)	1.43e+229	3.4e+38	4.21e+190	4.55e+212	1.07e+22	4.25e+190
ARF	14	(2+, 4*)	(2+, 2*)	6.85e+231	1.15e+77	4.42e+154	2.18e+215	3.65e+60	5.96e+154
FIR	19	(1+, 2*)	(3+, 2*)	2.07e+170	2.13e+96	9.73e+73	6.58e+153	6.75e+79	9.75e+73
JPEG IDCT	25	(5+, 5*)	(11+, 10*)	2.18e+470	3.83e+404	5.6e+65	6.8e+453	1.21e+388	5.61e+65
Mesa Interpolate	18	(2+, 2*)	(8+, 5*)	9.75e+288	2.86e+250	3.41e+38	3.10e+272	9.07e+233	3.41e+38

Table 29. Comparison of Resource Configurations and Security Enhancement of the Proposed Approach with Those of *Sengupta et al.* [23].

References

1. A. Sengupta, "Hardware Security of CE Devices [Hardware Matters]," in IEEE Consumer Electronics Magazine, vol. 6, no. 1, pp. 130-133, Jan. 2017, doi: 10.1109/MCE.2016.2614552.
2. A. Sengupta, "Evolution of the IP Design Process in the Semiconductor/EDA Industry [Hardware Matters]," in IEEE Consumer Electronics Magazine, vol. 5, no. 2, pp. 123-126, April 2016, doi: 10.1109/MCE.2016.2516119.
3. C. Pilato, S. Garg, K. Wu, R. Karri and F. Regazzoni, “Securing hardware accelerators: a new challenge for high-level synthesis,” *IEEE Embedded Syst. Lett*., vol. 10, no. 3, pp. 77-80, Sept. 2018.
4. X. Wang, Y. Zheng, A. Basak and S. Bhunia, “IIPS: Infrastructure IP for Secure SoC Design,” *IEEE Trans. Comput*., vol. 64, no. 8, pp. 2226-2238, 1 Aug. 2015.
5. Michael C. McFarland, Alice C. Parker, and Raul Camposano. 1988. Tutorial on high level synthesis. In Proceedings of the 25th *ACM/IEEE Design Automation Conference (DAC '88)*. IEEE Computer Society Press, Washington, DC, USA, 330–336.
6. R. Camposano, "From behavior to structure: high-level synthesis," in *IEEE Design & Test of Computers*, vol. 7, no. 5, pp. 8-19, Oct. 1990, doi: 10.1109/54.60603.
7. 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.

References

8. V. Krishnan and S. Katkoori, "A genetic algorithm for the design space exploration of datapaths during high-level synthesis," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 213-229, June 2006.
9. A. Sengupta and S. Bhadauria, "Automated exploration of datapath in high level synthesis using temperature dependent bacterial foraging optimization algorithm," 2014 *IEEE 27th Canadian Conference on Electrical and Computer Engineering (CCECE),* 2014, pp. 1-5.
10. A. Sengupta and M. Rathor, "Facial Biometric for Securing Hardware Accelerators," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 29, no. 1, pp. 112-123, Jan. 2021.
11. A. Sengupta, R. Chaurasia and T. Reddy, "Contact-Less Palmprint Biometric for Securing DSP Coprocessors Used in CE Systems," in *IEEE Transactions on Consumer Electronics*, vol. 67, no. 3, pp. 202-213, Aug. 2021.
12. A. Sengupta, D. Roy, S.P. Mohanty, and P. Corcoran, "DSP Design Protection in CE through Algorithmic Transformation based Structural Obfuscation," IEEE Trans. Consum. Electron., Vol. 63, no. 4, pp. 467 – 476, Nov. 2017.
13. Koushanfar F, Hong I, Potkonjak M. Behavioral synthesis techniques for intellectual property protection. *ACM Trans Des Autom Electron Syst* 2005;10(3):523–45.
14. 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.
15. A. Sengupta, D. Roy and S. P. Mohanty, "Triple-Phase Watermarking for Reusable IP Core Protection During Architecture Synthesis," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 4, pp. 742-755, April 2018.

References

16. Mahendra Rathor, Aditya Anshul, K Bharath, Rahul Chaurasia, Anirban Sengupta, Quadruple phase watermarking during high level synthesis for securing reusable hardware intellectual property cores, *Computers and Electrical Engineering*, Volume 105, 2023, 108476,ISSN 0045-7906.

17. A. Sengupta, E. R. Kumar and N. P. Chandra, “Embedding Digital Signature Using Encrypted-Hashing for Protection of DSP Cores in CE,” in *IEEE Transactions on Consumer Electronics*, vol. 65, no. 3, pp. 398-407, Aug. 2019.

18. A. Sengupta and M. Rathor, “IP Core Steganography for Protecting DSP Kernels Used in CE Systems,” in *IEEE Transactions on Consumer Electronics*, vol. 65, no. 4, pp. 506-515, Nov. 2019.

19. A. Sengupta and M. Rathor, “Securing Hardware Accelerators for CE Systems Using Biometric Fingerprinting,” in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 9, pp. 1979-1992, Sept. 2020.

20. A. Sengupta and D. Roy, “Antipiracy-aware IP chipset design for CE devices: A robust watermarking approach,” *IEEE Consum. Electron. Mag*., vol. 6, no. 2, pp. 118–124, Apr. 2017.

21. A. Sengupta and M. Rathor, “Structural obfuscation and crypto-steganography-based secured JPEG compression hardware for medical imaging systems,” *IEEE Access*, vol. 8, pp. 6543–6565, 2020.

22. M. Rathor, A. Sengupta, R. Chaurasia and A. Anshul, "Exploring Handwritten Signature Image Features for Hardware Security," in *IEEE Transactions on Dependable and Secure Computing*, 2022.

References

23. A. Sengupta, D. Kachave and D. Roy, "Low Cost Functional Obfuscation of Reusable IP Cores Used in CE Hardware Through Robust Locking," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 4, pp. 604-616, April 2019.

24. A. Sengupta, S. P. Mohanty, F. Pescador and P. Corcoran, "Multi-Phase Obfuscation of Fault Secured DSP Designs With Enhanced Security Feature," in *IEEE Transactions on Consumer Electronics*, vol. 64, no. 3, pp. 356-364, Aug. 2018.

25. Lao and K. K. Parhi, "Obfuscating DSP Circuits via High-Level Transformations," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 5, pp. 819–830, May 2015.

26. A. Sengupta and D. Roy, "Protecting an intellectual property core during architectural synthesis using high-level transformation based obfuscation," *IET Electronics Letters*, Vol: 53, Issue: 13, pp. 849 – 851, June 2017.

27. R. Schneiderman, "DSPs evolving in consumer electronics applications," *IEEE Signal Process. Mag.*, vol. 27 (3), pp. 6–10, 2010.

28. A. Sengupta, D. Roy, S. P. Mohanty, and P. Corcoran, "DSP design protection in CE through algorithmic transformation based structural obfuscation*," IEEE Trans. Consum. Electron.*, vol. 63, no. 4, pp. 467–476, Nov. 2017.

29. A. Sengupta and M. Rathor, "Protecting DSP Kernels Using Robust Hologram-Based Obfuscation," in *IEEE Transactions on Consumer Electronics*, vol. 65, no. 1, pp. 99-108, Feb. 2019.

References

30. A. Sengupta, D. Roy, S. P. Mohanty, and P. Corcoran, “DSP design protection in CE through algorithmic transformation based structural obfuscation,” IEEE Consum. Electron. Mag., vol. 63, no. 4, pp. 467–476, Nov. 2017.
31. Aditya Anshul, Anirban Sengupta,"IP Core Protection of Image Processing Filters With Multi-Level Encryption and Covert Steganographic Security Constraints", Proceedings of 8th IEEE International Symposium on Smart Electronic Systems (IEEE – iSES) , India, Accepted, Dec 2022.
32. M. Finke. (2015). Equisatisfiable SAT Encodings of Arithmetical Operations. [Online]. Available: http://www.martin-finke.de/documents/Masterarbeit_bitblast_ Finke.pdf.
33. M. Yasin, J. J. Rajendran, O. Sinanoglu, and R. Karri, “On improving the security of logic locking,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst*, vol. 35, no. 9, pp. 1411–1424, Sep. 2016.
34. Sheikh Ariful Islam, Love Kumar Sah, and Srinivas Katkoori. 2020. High-Level Synthesis of Key-Obfuscated RTL IP with Design Lockout and Camouflaging. *ACM Trans. Des. Autom. Electron. Syst*. 26, 1, Article 6 (January 2021), 35 pages.
35. R. Pappu, B. Recht, J. Taylor, and N. Gershenfeld, “Physical one-way functions,” *Science,* vol. 297, no. 5589, pp. 2026–2030, 2002.

References

36. U. Rührmair, J. Sölter, and F. Sehnke, “On the foundations of physical unclonable functions,” *IACR Cryptol.*, vol. 2009, p. 277, 2009.

37. J. Rajendran, M. Sam, O. Sinanoglu, and R. Karri. 2013. Security analysis of integrated circuit camouflaging. In Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security (CCS’13). ACM, New York, NY, 709–720.

Thank You!