Low Cost Security Aware High Level Synthesis Methodology

Anirban Sengupta
Computer Science and Engineering
Indian Institute of Technology Indore, India.
Email: asengupt@iiti.ac.in
S. Bhadauria
Computer Science and Engineering
Indian Institute of Technology Indore, India.
Saraju P. Mohanty
Computer Science and Engineering
University of North Texas, USA.
Email: saraju.mohanty@unt.edu

Abstract

Owing to massive complexity of modern digital integrated circuits (ICs) disabling complete in-house development, globalization of the design process establishes itself as an inevitable solution for faster and efficient design. However, globalization incurs importing intellectual property (IP) cores from various third party (3P) vendors, rendering an IP susceptible to hardware threats. To provide trust and security in digital integrated circuits within user constraints, design of a low cost optimized Dual Modular Redundant (DMR), through Trojan secured high level synthesis (HLS) methodology, is crucial. This paper presents exploration of a low cost optimized HLS solution capable of handling hardware Trojan (providing security) that alters computational output. The key contributions of the paper are as: (1) novel low cost security aware HLS approach, (2) novel encoding for representing bacterium in the design space (comprising of candidate datapath resource configuration and vendor allocation information for Trojan secured solution), (3) novel exploration process of an efficient vendor allocation procedure that assists in yielding a low cost Trojan secured schedule. Experimental results indicate significant reduction in the cost of security aware HLS solution (82.4 %) through the proposed approach compared to a recent approach.

***Index terms*—** High level synthesis, Hardware Security, Third party IP, Bacterial Foraging Optimization

I. Introduction

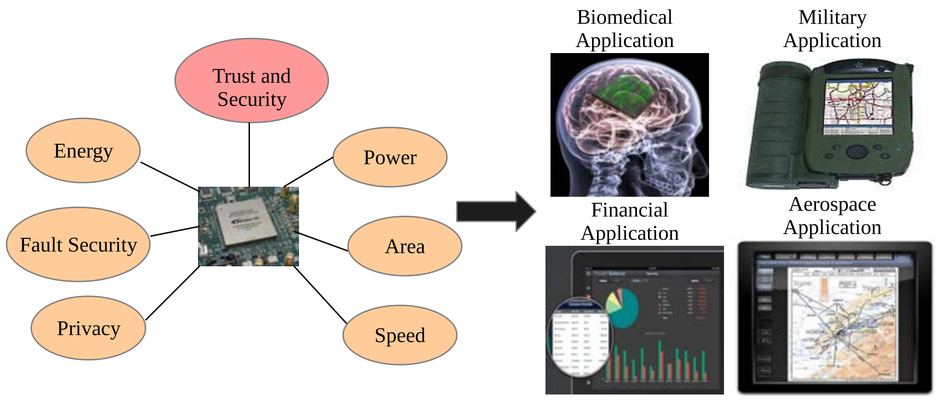
Digital integrated circuits which are main workhorses of applications in military, aerospace, biomedical etc. have security and trust as primary challenges. A paradigm shift in design cycle of digital ICs is needed to incorporate trust and security besides issues like power, area, and energy, as shown in Fig. 1a. Hardware Trojan's are malicious logic embedded in an IP core of a digital IC by an adversary in order to induce malfunctioning. With the globalization in the design process of ICs, there have been key concerns on the security and trustworthiness of 3PIPs [1], [2]. This is because companies worldwide participate in the development of ICs which makes the design process vulnerable due to increase in possibility of untrustworthy vendors. Designs at each levels of abstraction are outsourced to companies worldwide. Fig. 1b shows the hardware treats in a typical IC development cycle. During the design process an adversary may corrupt an IP by inserting hardware Trojan by either external activation (such as antennas or sensors) or internal activation (such as internal states of finite state machine (FSM) or counters) [3].

Threat Model: The paper targets Trojan in 3PIPs that only change computational output value (and produces no other impact). This work does not address Trojan in control and interconnect fabric of the IPs. This paper doesn't target any other class of Trojans including those that steal information/data. In other words, following threat model is considered: a malicious third party IP used as module is present in HLS library in inactive state. The insertion of Trojan is possible in the following way: A malicious IP/module present in the library of a HLS tool is designed either by a rogue in the 3PIP design house or by a rogue in the in-house design team. Since the adversary may not provide Trojan-related information, hence detection by the validation team is difficult.

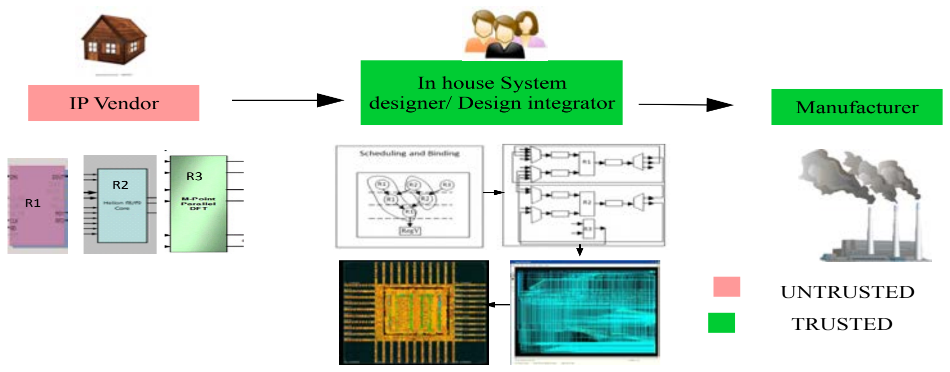
The **novel contributions of the current paper** include following:

1) Presents a novel low cost security aware HLS methodology for optimized hardware Trojan secured schedule based on user constraints.
2) Presents a novel problem encoding using bacterial foraging that explores an efficient vendor allocation procedure besides datapath resource configuration.
3) Provides lower cost Trojan secured solutions that are of better quality than solution obtained through recent approach [4].

The notations used for the proposed approach and its explanations are presented in Table I. The rest of the paper is organized as follows: Section II presents related prior research. Section III provides broad perspective on security aware HLS. Section IV presents our formulation and models for security



(a)



(b)

Fig. 1: Development phase of an IC with likelihood of untrustworthy vendor involving security as design objective

aware HLS. Section V presents our novel security aware HLS methodology. Section VI presents the experimental result and finally Section VII concludes the paper.

II. Related Work

A. *Security Aware Approaches at Behavioural/Lower levels*

The hardware Trojan issue so far have mainly been dealt at the lower levels of chip designing [5], [6], except in [4], [7]. Researchers in [8] used multi supply transient current integration methodology to detect hardware Trojans. In [9], Trojan detection based on path delay and power of the manufactured chips is used to identify Trojan's in manufactured chips. But such techniques are not effective in HLS since there are no golden solutions available to compare with. However, none of the previous approaches above considers Trojan security during HLS except in [4], [7]. In [4], [7], Concurrent Error Detection (CED) technique is adopted where distinct 3PIP vendors are used to identify the Trojan's. However, [4],

TABLE I: Notations used in the Current Paper.

p	Bacterium population size indicating the number of initial design solutions participating in DSE	D	Total available resource types in solutions during HLS
x_i	A bacterium position denoting a candidate design solution in HLS	V_j	Number of vendors used during hardware allocation in HLS
x_i^{Last}	indicating the last visited design solution of i^{th} bacterium	L_T^{DMR}	Total delay of a dual modular redundant (DMR) schedule during HLS
x_i^{New}	indicatig the new design solution of i^{th} bacterium	$N(R_d)$	Number of instances of a resource type $'d'$ of a design solution
$C(i)$	step size taken by a bacterium indication exploration drift during DSE	$N(R_d)_i^{New}$	New value of the d^{th} resource type of i^{th} bacterium (design solution)
$N(R_d)_i^{Min}$	Minimum value of d^{th} resource type of i^{th} bacterium (design solution)	$N(R_d)_i^{Max}$	Maximum value of d^{th} resource type of i^{th} bacterium (design solution)
N_c	Maximum number of chemotactic steps i.e. the terminating criteria of DSE in HLS	A_{max}^{DMR}	DMR solution with maximum area in the design space during HLS
R_x^i	Candidate design solution indicating resource configuration and vendor dimension	L_{cons}	User specified execution time constraint i.e. the constraint specified by the user as upper limit to latency during fitness evaluation of design solution
A_{cons}	User specified area constraint i.e. the constraint specified by the user as upper limit to latency during fitness evaluation of design solution	L_{max}^{DMR}	DMR solution with maximum time /delay in the design space during HLS
A_T^{DMR}	Total area utilized by a DMR schedule during HLS	$C_f(R_x^i)$	Cost of a candidate design solution (R_x^i) participating in DSE process
A_v	Vendor allocation Procedure i.e. how hardware from different vendors are internally allocated in a DMR schedule	$A(R_d^{V_j})$	Area of a resource type (R_d) corresponding to vendor (V_j)
$R_d^{V_j}$	Number of instances utilized from vendor V_j for a n and n' resource type R_d	n and n'	Size of original and duplicate DFG used during scheduling in HLS
$D(op_z^{V_j})$	Delay of operation z, assigned to vendor V_j	cs	Control steps of a schedule in HLS
m	Maximum number of instances of resource type R_d for vendor V_j		

[7] does not provide a low cost economical security aware HLS solution. It may be noted noted that "low cost" in this paper refers to low design implementation cost in terms of hardware area and latency of a Trojan secured design solution. The details of this hybrid design cost function is described later in eqn. 3. Further, approach [7] does not investigate into exploration of efficient distinct vendor allocation procedure of similar operations as it affects final delay and area of the schedule. Moreover, in [10] authors developed an ATPG scheme for detection of Hardware Trojan Horses (HTHs) at the logic level, which depends on rare input triggering conditions. The detection scheme is based on both GA and Boolean satisfiability approach. The approach models the effect of each Trojan instance as stuck-at fault. However, the approach presented in [10] does not deal with third party IP cores used at higher abstraction levels such as during HLS. Further, the approach does not perform any optimization of hardware overheads associated with Trojan detection.

In [11], authors have inspected class of hardware Trojan that causes set-up time violation at clock input of flip-flops. The assumption used in their attack model considers a collusion (nexus) between the third party designer of the IP core and the person-in-charge of deployment (i.e. system integrator of the IP core). However, the proposed approach handles another type of Trojan that changes computational output value. Further, we assume that in the IC design flow, the system integrator is trustworthy (therefore with no chance of collusion or malicious nexus with the third party IP designer).

Authors in [12] explored class of hardware Trojans referred as ”Malicious Off-chip Leakage Enabled by Side-channels” (MOLES), which may intentionally leak secret information through side-channels. The Trojan considered is capable to leak multi-bit information/secret keys under different noise power levels of the host IC to evade detections. However, the proposed approach handles another type of Trojan that changes computational output value (and does not aim to leak confidential information). Further, the proposed approach considers malicious logic in the component library of a HLS process, unlike [12].

B. Non Security aware approaches at Behavioural level

Authors in [13] have used clustering based technique for a simulated annealer explorer for performing area-delay trade-off between exploration speed and quality of solution. Further in [14] an integrated DSE methodology considering compiler and architectural level transformations for performing area-delay trade-off in SPARK-HLS tool [15] is presented. Moreover, in [16] an approach on spectral techniques and resource surface models for area-delay trade-off is presented. Besides above, in [17] and [18] exploration problem is solved by machine learning algorithms, ‘random forest’ and ‘genetic algorithm predictive model’. DSE process runs by predictive models without running synthesis for each new configurations during area-delay trade-off. However, the proposed approach is driven through bacterial foraging and uses Trojan security, area and delay as design objectives. Moreover, the proposed approach uses bacterial foraging which is a more simplified model that produces high quality solutions using tumble vector as well as considers Trojan security, area and delay as design objective.

III. Security Aware HLS: A Broad Perspective

HLS involves translation from an algorithmic/behavioral description of an application to its equivalent register transfer level (RTL) structure which implements behavioral description represented in the form of a directed graph, referred as control data flow graph (CDFG) [19], [20]. It involves exploring alternative design solutions by analyzing its scheduling and allocation details through corresponding area and delay evaluation [21]–[29]. In recent years, there is a paradigm shift from traditional area/performance aware HLS methodology towards security aware HLS methodology.

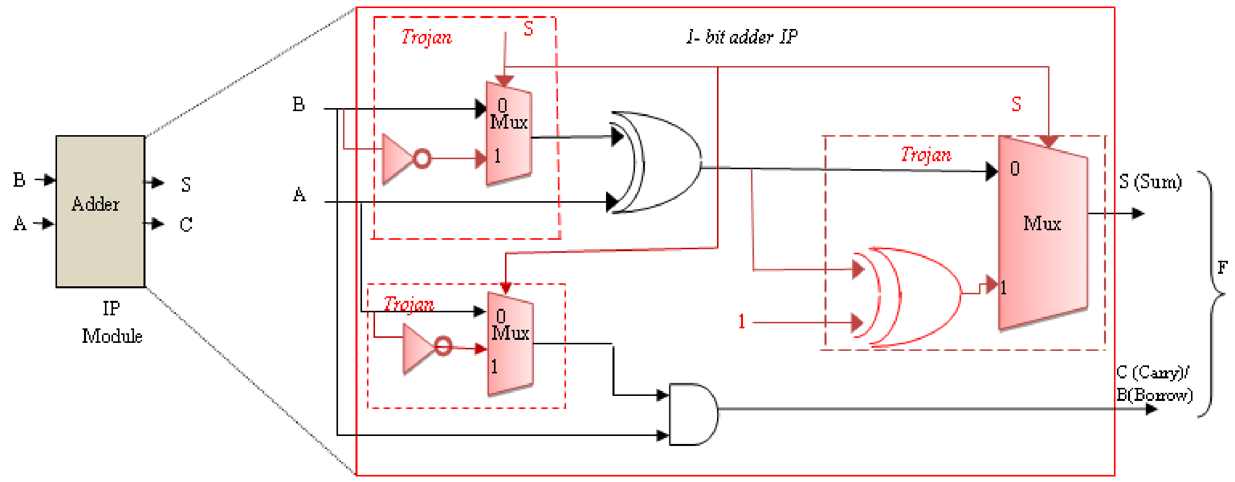


Fig. 2: Trojan logic inserted by an adversary in a sample IP. *Note* : When select (S) = 1, then, Trojan blocks get activated and the adder module starts performing subtraction / (A + B + 1).

A. *Motivational example: Hardware Security/Trust at Behavioral Level*

Let us consider an example during HLS through a case of Trojan as shown in Fig. 2. It shows an adder IP designed by an untrusted third party vendor and included in HLS tool library. In this design, a possible scenario, where insertion of Trojan is made by malevolent alteration of the hardware/logic of the adder IP by an untrusted vendor is shown in figure (dotted blocks marked in red). This Trojan remains dormant until adversary triggers the select (S) of the circuit at the moment of its choice through external or internal activation(as discussed before), when the adder IP maliciously starts behaving as a subtractor without the knowledge of HLS designer. However, during design verification such as RTL simulation the adder IP continues to function as an adder, owing to inactive state/dormancy of the Trojan. Thus despite carrying Trojan logic in the IP, it provides an impression to the HLS designer that the IP is functionally correct. When integrated in the design, the adder IP starts behaving maliciously, due to rare triggering by an adversary. Therefore, during HLS design it should be ensured through proper allocation rules the design is Trojan secured.

B. *Motivation of proposed work*

In case of HLS, the 3PIPs are the IPs/modules present in the HLS module library, which may have been imported from an outside (third party) vendor and may contain malicious logic inserted by a rogue in the third party design house. Typically, the RTL files of the modules/IPs of the library are provided by the HLS company which it may have imported from third party (untrusted) vendors as RTL files. These are highly likely to contain dormant/untriggered Trojan logic. The proposed approach deals with Trojan in the modules/IP that affects the computational output value of HLS designs. However the key is that Trojan only becomes visible at runtime (after external triggering), thereby remaining completely obscured before that, thus difficult to be detected during RTL simulation. Thus during functional verification steps the Trojan remains unnoticed in most cases either because it remains dormant (not triggered) or some of the outputs reflect no change in functionality. Therefore, detection from the high level description (in C/HDL code) of the IP design is difficult. Further, for the case when the Trojan is 'always on' in an IP, it may be possible that most of the outputs reflect no change during RTL simulation (except a few). Therefore, without careful examination, the Trojan may sometimes go unnoticed. Thus during rare input triggering conditions, the Trojan from dormant state gets activated causing wrong computational output for a specific hardware unit. This erroneous computed value of the specific FU results in erroneous value of the primary output as well. However as the other unit of the DMR utilizes FUs from a distinct vendor therefore it remains Trojan free or produces a different Trojan logic, thereby producing a different value at the primary output of this unit. Comparison of the two primary outputs from the two units results in difference in magnitude thus yielding detection of Trojan.

One of the possible ways for successful detection of Trojan in HLS is using atleast two vendors in a DMR design. However so far in the literature, low cost Trojan security aware HLS based on user provided constraints has not been done. This normally results in area and delay overhead. The cost increase due to involvement of two vendors is also minimized in our proposed approach through the following techniques: a) Exploration of optimal vendor allocation procedure b) Exploration of optimal datapath resource configuration.

IV. Low Cost Security Aware HLS: Formulation And Models

The goal of proposed approach is to explore a low cost security aware HLS solution based on user constraints. As an example, area and delay have been considered as constraints in the current paper.

A. Problem Definition

Determine an optimal solution, R_x^i = $\{x_i,\ A_v\}$ = $\{$N(R_1), N(R_2),..N(R_D), $A_v\}$, while exploring the design space of an input CDFG and conflicting constraints. The problem can be formulated as: Find: Optimal (R_x^i), with minimum, *Cost(*A_T^{DMR}*,* L_T^{DMR}*)* subjected to: $A_T^{DMR} \leq A_{cons}$ and $L_T^{DMR} \leq L_{cons}$ and Trojan security, where the variables have been explained in Table. I. 'A_v' is the vendor allocation procedure (where A_v = '1' or '0'), 'i' indicates bacterium number in the search space, R_x^i indicate a candidate design solution corresponding to bacterium 'i'. Vendor allocation procedure in this paper indicates the process of assigning hardwares from distinct vendor in a DMR schedule.

A Trojan secured solution is defined as a DMR schedule that has the ability to produce either a correct output or signal the presence/detection of a Trojan (due to unidentical outputs generated from original and duplicate units of a DMR).

B. Evaluation Models

In the proposed work, bacterium foraging optimization algorithm is used as exploration backbone for yielding a low cost security aware HLS solution (motivation of using BFOA is provided in subsequent sections). Each bacterium position represents a Trojan secured design solution in the design space.

1) Proposed area model: Total silicon area consumed $\left(A_T^{DMR}\right)$ by a Trojan secured design solution is given by the following model:

$$A_T^{DMR} = \sum_{j=1}^{2}\sum_{d=1}^{m}\left(A\left(R_d^{V_j}\right) \times R_d^{V_j}\right) + (A(mux) * N(mux)) + (A(demux) * N(demux)) + \quad (1)$$
$$A(errordet) + (A(buffers) * N(buffers)),$$

'm' is the maximum number of instances of resource type 'R_d' belonging to vendor V_j. Total area utilized by a DMR schedule during HLS, where DMR schedule comprises of original CDFG unit and duplicate CDFG unit scheduled concurrently based on candidate design solution (R_x^i). It may be noted that the area comprises of components due to hardware resources (16-bit carry look ahead (CLA) adder, 32-bit asynchronous multiplier and 16-bit subtractor), interconnect units (16-bit/32-bit mux and 16-bit/32-bit demux), error detection unit such as 16-bit/32-bit comparator and 16-bit storage units due to overhead incurred from buffering during storage of operation output in DMR scheduling. The area evaluated is with respect to module library generated with CMOS 90nm technology node [30], [31], [32]. The area model therefore accurately captures the most important portions of a datapath including steering logic such as multiplexers and demultiplexers, storage hardware such as buffers, error detection block such as comparator and FUs hardware described in the form of transistors. In this paper the area model does not consider the impact of flip flops and interconnects.

2) Proposed Delay Model: For given 'D' functional resources the delay of a Trojan secured design solution is given as follows:

$$L_T^{DMR} = \sum_{cs=1}^{n} Max\left(D(op_z^{V_j}), ..D(op_n^{V_j}), D(op_{z'}^{V_j}), ..D(op_{n'}^{V_j})\right), \quad (2)$$

where $1 \leq j \leq 2$, $1 \leq z \leq n$ and $1' \leq z' \leq n'$. (z and z' = indicate operations in original unit and duplicate unit of a DMR). 'cs' indicates total control steps of a DMR schedule. The delay model is based on the latency values of each functional hardware described at 90nm technology scale [30].

3) Fitness Function: The fitness function considers delay and area of a design solution, formulated as follows [24]:

$$C_f(R_x^i) = W_1\left(\frac{A_T^{DMR} - A_{cons}}{A_{max}^{DMR}}\right) + W_2\left(\frac{L_T^{DMR} - L_{cons}}{L_{max}^{DMR}}\right). \tag{3}$$

In the above expression, user defined weights W_1 and W_2 are assigned values of 0.5 during exploration to offer equivalent priority. Further, it considers the maximum values of area and delay during evaluation to yield a normalized value of cost (between 0 and 1). Since the objective of the proposed exploration approach is to satisfy the user constraints as well as minimize the hybrid cost, as presented in Section IV.A, hence a higher negative value is better.

C. Motivation on using BFOA

A regular exploration process with area/power - delay as design objective is considered intractable [33]. Inclusion of Trojan security as additional design objective intricates the mechanism and requires advanced search algorithm capable to combat the intricacy and find an optimal low cost design solution. BFOA would be considered suitable for such problems over Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) because of the following advantages:

- unique feature of chemotaxis that enables usage of tumble vector for change in direction of search path in a design space, if needed.
- simplified framework than GA and PSO as it does not heavily depend on pre-tuning of stochastic parameters.
- unique feature of elimination dispersal that allows killing a certain design solution if found unproductive i.e. higher in cost (while simultaneously injecting a fresh low cost candidate design solution) [24].

V. Proposed Novel Low Cost Optimized Trojan Secured HLS Methodology

This section presents a novel low-cost secured HLS flow based on DMR logic. DMR typically results in duplication of operations, which in turn may lead to more steering logic and internal buffering compared to a normal (non- DMR) design. However, proposed low cost Trojan secured DMR scheduling does not lead to functional resource overhead as it is designed based on datapath resource constraints (which is iteratively obtained through BFOA process). Further, the hardware/power overhead of proposed Trojan secured DMR (i.e. steering logic and internal buffers) is also optimized in our approach. Therefore, all the associated DMR overheads are minimized in our proposed methodology through integration with an advanced design space exploration framework.

A. Proposed Low Cost Security aware HLS

The block diagram of the proposed methodology is shown in Fig. 3a. In our design flow DMR scheduling is performed using list scheduling which is a resource constrained scheduler. The operations of the original unit is given higher priority than operations of duplicate during concurrent scheduling. After scheduling, allocation is performed through exploration of vendor assignment process that enables low cost hardware allocation to operations. Details are provided in section V-C and V-D respectively. Once scheduling and allocation are over, binding of functional units (FU) is performed to estimate the details of steering logic (such as size of multiplexers and demultiplexers). The details of steering logic derived after binding is fed as a component to our area model that enables fitness evaluation of potential design solutions (bacteria) in proposed exploration process. The flow diagram of the proposed methodology for low cost Trojan secured HLS solution based on area-delay constraint is shown in Fig. 4. Fig. 4 is divded into two portions: the blue boxes indicate the Trojan secured design steps while the red boxes indicate the steps for BFOA-DSE framework. The inputs to the proposed approach include library details, CDFG, 'Nc', 'p' and user constraints (example: area and delay). The proposed approach terminates on either of the condition:

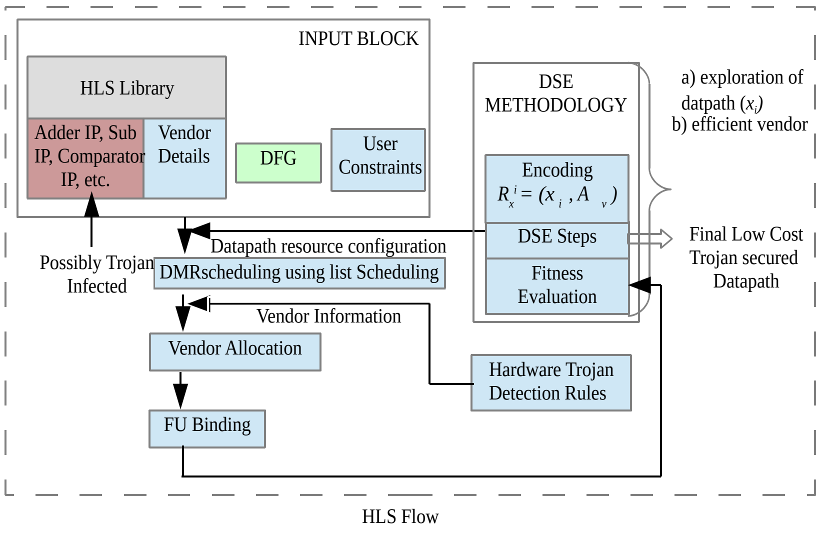
- s1: Designer specified 'N_c'
- s2: No improvement is observed in global best in last 10 chemotactic steps.

A novel encoding is presented in the proposed approach for bacterium that includes a datapath resource configuration for bacterium position and vendor allocation procedure 'A_v' indicated as R_x^i:

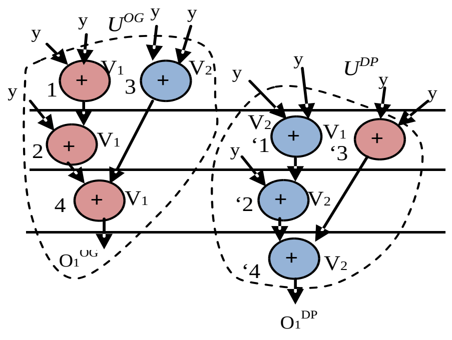
$$R_x^i = (x_i, A_v), \tag{4}$$

where x_i has been defined in Table. I. Motivation of adding dimension 'A_v' is discussed in next sections.

A simple demonstrative example is shown in Fig.3b where a infected adder in HLS library is acting as a subtractor (as shown in Fig. 2); y is the primary input. Let us consider a scenario: Adder of vendor V_1 is malicious: In such case assuming the adder maliciously performs subtraction. The subsequent output from original unit is $-y$, however, the duplicate output is $3y$, indicating Trojan security that is detection (due to difference in computation value). As discussed in [4], it is highly unlikely that different Trojans in different 3PIPs will produce identical wrong outputs. In other words, chance of IP from both third party vendors being Trojan infected or carrying same Trojan logic is very rare. In terms of security of the design as explained above, the proposed approach is capable of detecting any malicious infection in a 3PIP that results in change in computational value at the output. In the proposed approach both


INPUT BLOCK
HLS Library
Adder IP, Sub IP, Comparator IP, etc.
Vendor Details
DFG
User Constraints
Possibly Trojan Infected
Datapath resource configuration
DMRscheduling using list Scheduling
Vendor Information
Vendor Allocation
FU Binding
Hardware Trojan Detection Rules
DSE METHODOLOGY
Encoding
DSE Steps
Fitness Evaluation
a) exploration of datpath (x_i)
b) efficient vendor
Final Low Cost Trojan secured Datapath
HLS Flow

(a) The Proposed novel low cost security aware HLS methodology.



(b) Simple motivational example

Fig. 3

the distinct vendor allocation procedure (A_v = 0 and A_v =1) are able to provide comprehensive security against malicious Trojan in an IP (that results change in computational value).

B. *Initialization of Bacterium*

The position of bacterium represents a Trojan secured HLS solution. The bacterium position 'x_i' of an 'i^{th}' bacterium is given as follows:

$$x_i = (N(R_1), N(R_2), ...N(R_d), ...N(R_D)), \tag{5}$$

In the above expression the rests of the bacteriums ($x_4...x_n$) are initialized by the following:

$$x_n = (\frac{min(R_1) + max(R_1)}{2} \pm \alpha, \frac{min(R_2) + max(R_2)}{2} \pm \alpha, ... \frac{min(R_d) + max(R_d)}{2} \pm \alpha), \tag{6}$$

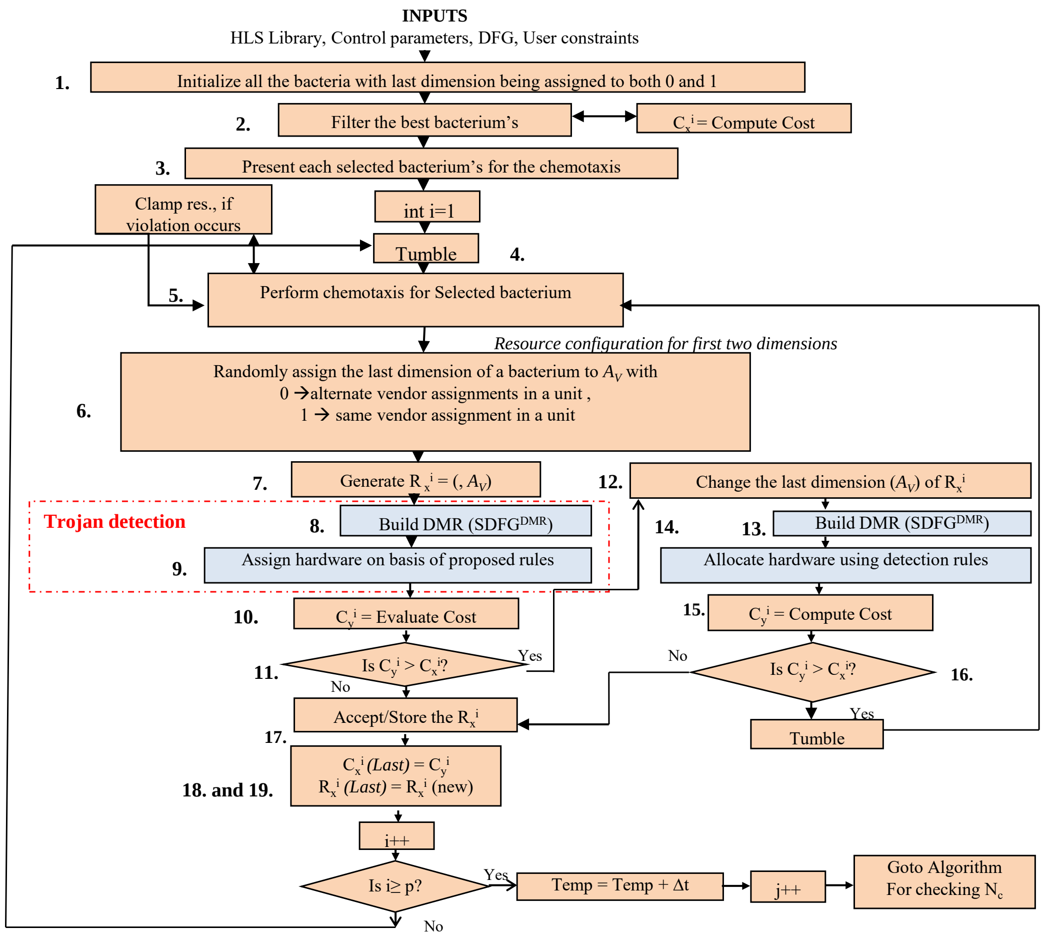

INPUTS
HLS Library, Control parameters, DFG, User constraints
1. Initialize all the bacteria with last dimension being assigned to both 0 and 1
2. Filter the best bacterium's
C_x^i = Compute Cost
3. Present each selected bacterium's for the chemotaxis
Clamp res., if violation occurs
int i=1
Tumble
4.
5. Perform chemotaxis for Selected bacterium
Resource configuration for first two dimensions
6. Randomly assign the last dimension of a bacterium to A_V with 0 → alternate vendor assignments in a unit, 1 → same vendor assignment in a unit
7. Generate R_x^i = (, A_V)
12. Change the last dimension (A_V) of R_x^i
Trojan detection
8. Build DMR (SDFG^{DMR})
9. Assign hardware on basis of proposed rules
14.
13. Build DMR (SDFG^{DMR})
Allocate hardware using detection rules
10. C_y^i = Evaluate Cost
15. C_y^i = Compute Cost
11. Is $C_y^i > C_x^i$?
Yes
No
16. Is $C_y^i > C_x^i$?
No
Yes
Tumble
17. Accept/Store the R_x^i
18. and 19. C_x^i (Last) = C_y^i R_x^i (Last) = R_x^i (new)
i++
Is i≥ p?
Yes
No
Temp = Temp + Δt
j++
Goto Algorithm For checking N_c

Fig. 4: Flowchart of the Proposed Low Cost Security-Aware HLS Methodology.

where, α is a random integer between min and max of particular resource type. For example, as seen in Fig. 5, the benchmark shown has two types of resources (i.e. D=2). Therefore, a bacterium position for the given CDFG can be given as: $x_i = (N(add), N(mul))$.

C. *Movement using Chemotaxis*

In BFOA-DSE, locomotion of a bacterium (design solution) from one point to other is symbolized as chemotaxis [24], [34], [35]:

- $Tumble$: This variable regulates the step size, $C(i)$ using , Δ a random vector whose elements lie in [-1, 1].

- $Stepsize, C(i)$: This variable regulates the step of a movement and is taken in any random direction: $C(i) = C(i) + 2$.
- $Move$:

$$x_i^{New} = x_i^{Last} + C(i)\frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}}. \tag{7}$$

In the above expression, new and old hardware resource array of i^{th} bacterium solution is represented as x_i^{New} and x_i^{Last}; the step size and tumble vector is denoted by $C(i)$ and $\Delta(i)$. A Trojan secured DMR schedule ($SDFG^{DMR}$) with distinct vendor assignment rule is generated based on each bacterium position.

Each Trojan secured/security aware solution obtained is assessed on objectives of area and delay to determine the cost of the solution. This process of assessing design solutions evolves through the proposed exploration using chemotaxis to yield an optimal low cost security aware HLS solution based on A_{cons} and L_{cons}.

D. Designing a Trojan Secured DMR schedule for a design solution

In the proposed approach for yielding a low cost Trojan secured HLS solution, DMR logic with specific vendor allocation rule is employed. In order to obtain a Trojan secured solution, its corresponding DMR schedule first needs to be obtained. In proposed DMR logic, complete duplication is done for all the operations of the CDFG. This is because, comparison of final output cannot be performed if completely duplicated unit is not available. It is possible that any subsequent operation assigned to hardware modules after the intermediate stage is the activation point of Trojan. Hence comparison of final output from original and intermediate stage output from duplicate unit may not provide Trojan detection for all cases. However, comparison of final outputs of original and duplicate comprehensively covers all Trojan detection possibilities. Operations of original unit and duplicate unit in a DMR design are concurrently scheduled using list scheduling based on the information of the datapath architecture $(N(R_1), N(R_2), ..N(R_d), ..N(R_D))$ in the candidate design solution (x_i). Employing DMR scheduling concept for Trojan security results in overheads in the form of hardware area and delay. This is because, due to concurrent scheduling of operations in duplicate unit, additional latency is incurred. Further, due to duplication of all operations, additional steering logic (mux and demux) and buffering (storage hardware) is required. All these factors contribute to the area overhead. It therefore becomes mandatory to generate a low cost Trojan secured solution that optimizes the delay and area overhead incurred due to DMR scheduling. We resolve this by two factors: a) Exploring optimal vendor allocation (discussed next paragraph); b) exploring optimal datapath resource combination.

The vendor allocation rule states that two distinct vendors are required for operation assignment in DMR such that the similar operations v of original unit (U^{OG}) and v' of duplicate unit (U^{DP}) are assigned to distinct vendor [4]. This enables Trojan security as it is highly unlikely that both third party vendors are simultaneously untrusted (i.e. having same Trojan embedded). Checking through error detection block/comparator is not performed between operations scheduled in different clock cycles or in the same cycle on different operator type. Rather checking or comparison is only performed at the final/primary output of both original and duplicate units of DMR schedule. If a DMR schedule comprising of original and duplicate units has multiple final outputs, then each of the final output of original and duplicate units needs comparison. If a difference in magnitude of final output values between both values are observed, then it indicates Trojan detected (as one of the vendors is certainly malicious), else, the final similar output is passed to the next stage. However it is important to note that distinct vendor assignment rule for Trojan security can be implemented in more than one ways, therefore, the most optimal distinct vendor allocation to similar operations (in original and duplicate) is explored through our proposed scheme. Therefore, how the two distinct vendors are allocated inside the DMR scheduling dictates the final delay and area of the design. This is due to the fact same IP from two distinct vendors can never have exactly same area and delay.

It is assumed in this paper that the IP characteristics from vendors (V_1 and V_2) are as follows: multiplier and adder provided by vendor V_1 has area = '2468au' and '2034au', delay = '10000ns' and '265ns', while multiplier and adder provided by vendor V_2 has area = '2464au' and '2032au', delay = '11000ns' and '270ns' respectively. The values of area and delay of modules assumed is with respect to 90nm technology scale adopted from [30], [31], [32]. Therefore, only using distinct vendor assignment for detection without investigating into the procedure of distinct vendor assignment in DMR may lead to missing a low cost security aware HLS solution. Let us now discuss the two different ways (denoted by 'A_v' = 1 and 'A_v' = 0) how distinct vendor allocation can be made inside a DMR scheduling in order to provide Trojan security. The binary value of 'A_v' as 0 or 1 is inferred as follows:

1) Technique 1: Vendor allocation procedure (Type 0): 'A_v' = 0:

- In a control step of a unit, alternate vendor allocation to operations is made. (For instance, in Fig. 5, operation 3 and 6 are allocated alternatively to hardware units of vendor 'V_1' and 'V_2').
- Different vendors are allocated to similar operations of both (U^{OG}) and (U^{DP}).

2) Technique 2: Vendor allocation procedure (Type 1): 'A_v' = 1:

- Strict assignment of vendor 'V_1' for all operations of U^{OG} and assignment to vendor 'V_2' for all operations of U^{DP}. Fig. 6 shows an example of this vendor assignment.
- Similar operations of both U^{OG} and U^{DP} are allocated to different vendors.

In both above techniques, priority is provided to the operation of U^{OG} at any time there is a conflict of operation during scheduling between operation of U^{OG} and U^{DP}. For example, as shown in both Fig. 5 and 6, there is a conflict between operations 1', 2' and 8 at 3^{rd} control step during scheduling. Since, in our approach operations of original is given priority over operations of duplicate hence opn 8 is scheduled first. One more multiplier resource is still free, hence opn 1' is given priority over 2' as in the input CDFG file, 1' comes before 2' in sequence. We note that scheduling 2' before 1' would have resulted in same latency. However since 3' is only useful when 4' is complete, (which is dependent on 1') hence 3' is scheduled after 1' in fourth control step.

As shown in Fig. 5 and 6, comparison of final outputs from both U^{OG} and U^{DP} is performed through an error detection block/comparator. This enables to indicate presence of malicious (Trojan) logic if difference in magnitude is found or absence of any Trojan if no difference is observed.

As seen in Fig. 5 and 6, for a hardware resource array, x_i = 2(+), 2(*), two potential security aware DMR schedules exists for IIR filter benchmark on the basis of vendor allocation technique 'A_v' = 0 and 1. In other words, for R_x^i = (2(*), 2(+), 'A_v' =0), the delay is: 55, 810 nano-seconds (ns) and area is: 9880 area unit (au); while, for R_x^i = (2(*), 2(+), 'A_v' =1), the delay is: 53, 810 ns and area is: 14812 au. Undoubtedly, a difference in magnitude is visible in the delay and area of the two possible security aware DMR scheduling solutions both preserving the Trojan security rule. The solutions generated in Fig. 5 and 6 are both security aware, but, one is better than the other in a specific objective.

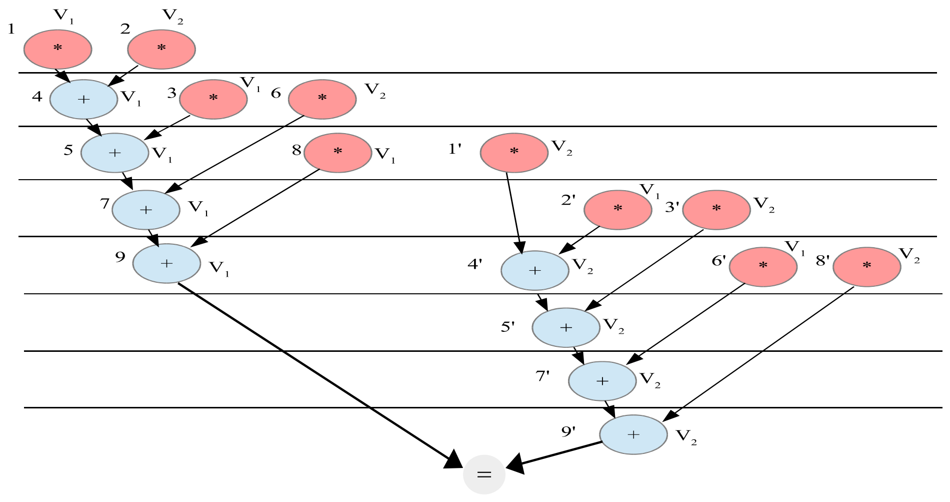


Fig. 5: *DMR schedule of IIR for A_v=0; x_i = 2(+), 2(*) assignment of two vendor types.*

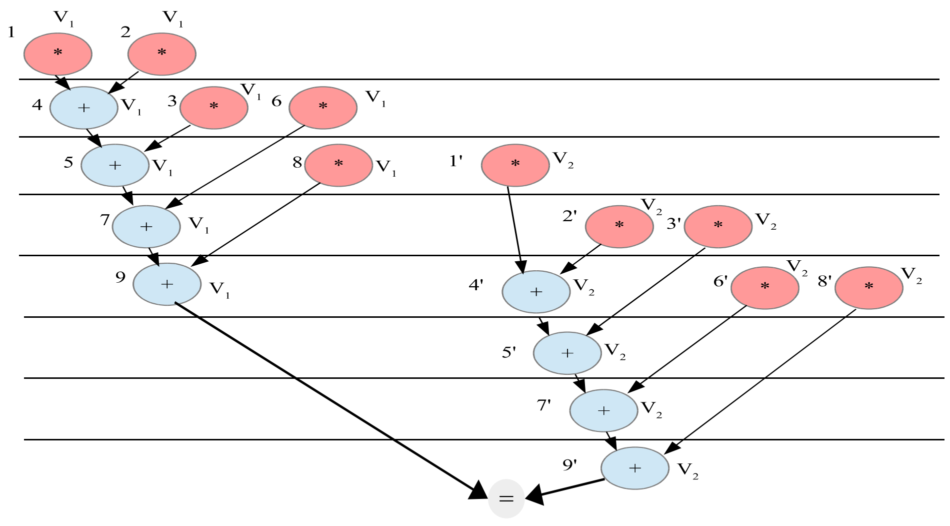


Fig. 6: IIR filter for A_v = 1; x_i = 2(+), 2(*) indicating each entire unit strictly assigned to same vendor type (U^{OG} to 'V_1' and $U^{DP}$to 'V_2').

VI. Experimental Results

A. Experimental Setup and Benchmarks

The experimental set up comprises of Intel Core-i5-3210M CPU with 3MB L3 cache memory, 4GB DDR3 primary memory and 2.5GHz frequency. Java has been used for implementation of proposed approach and [4]. As evident from experiments executed, the proposed approach is scalable and found capable to handle problems of any size (for instance, benchmarks higher than 100 nodes were tested which yielded low cost optimal solution with reasonable runtime of $\approx$ 91 s). For experimental results, we designed a HLS tool comprising of BFOA driven exploration module. This is because most of the available tools perform GA based exploration which incurs lot of runtime overhead. We note here that since our HLS tool comprises of BFOA driven exploration module as an independent package, hence it is possible for its integration with open source or commercial HLS tool (providing Trojan security and area-delay optimization). Note: The module library considered is adopted from [30], [31], [32]. Further, the values of area and delay of final HLS solution reported in this section are through model proposed earlier.

B. Analysis of Quality of Results

The exploration parameters considered in our proposed approach during experimentation are the following: number of chemotaxis steps (N_c), swarm size/population (p), terminating criterion (s) and step size ($C(i)$). To evaluate the effectiveness of our multi objective optimization algorithm, the following

TABLE II: Results Of Proposed Approach

Note: N_c = 120, p = 3, terminating criteria $s1$ or $s2$, $C(i)$=2 Δ = [-1, 1]

Benchmark [36], [37], [38]	L_{cons} (ns)	L_T^{DMR} (ns)	A_{cons} (a.u)	A_T^{DMR} (a.u)	**Cost**
IIR Butterworth (2^{nd} order)	39845	23080	19744	19744	-0.150
MPEG Motion Vectors	88307	36240	49188	34900	-0.245
ARF	132810	89890	23747	16198	-0.235
FFT	99390	66270	30627	29152	-0.135
FIR (6-tap)	50000	45890	26000	23322	-0.034
MESA Interpolate	237600	99350	117972	60254	-0.292
IDCT	119160	77080	42858	31582	-0.224
WDF	115600	111695	16547	13064	-0.090
DCT	170810	88810	32836	16610	-0.218
GLRT	170000	109800	50000	24962	-0.230

goals are considered: (1) discover solutions close enough to the Pareto front(P^*) and (2) explore solutions as diverse as possible in the non-dominated set (Q). The GD metric measures convergence of obtained solutions. Further, metrics which quantify the diversity among obtained non-dominated solutions are spacing (S) and spreading (Δ). In addition, W_m provides a combined qualitative measure of both closeness and diversity of the solutions (lower W_m is better) [39].

Table II indicates that for all benchmarks, the proposed approach is able to attain solutions which lie within the user provided constraints of area and delay (while simultaneously minimizing the cost in Eqn. 3). The possibility of false negative outcome which indicates an existing Trojan being undetected does not arise. This is because the allocation step meets the sufficient condition of detection by allocation to hardware from distinct vendors for every similar operation of original and duplicate unit. Moreover it has been assumed that the designed security aware DMR schedule solution is only susceptible to those Trojans which have potential of upsetting the computational output. This indicates that the security aware solution is assumed free of any other faults affecting final output. Therefore false positive outcome does not exist for proposed approach.

Table III highlights the impact of population size p on the runtime of proposed approach. It indicates that for all benchmarks with the growth in population size, the runtime of the proposed approach to find the non-dominated solutions escalates due to increase in computational complexity per iteration.

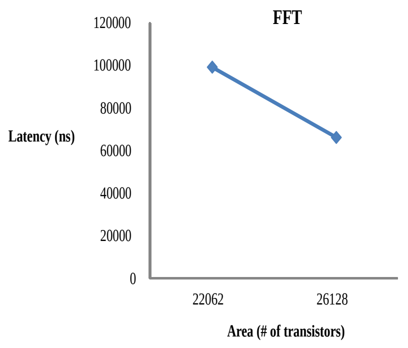
Additionally, Table IV, V and VI present the results of proposed approach on quality metrics for various population sizes. As seen, for IIR Butterworth, MPEG MV, ARF, WDF, IDCT and FFT, the GD and S for p = 3, 5 and 7 are same, however, the Δ at p = 3 is better compared to p = 5 and 7. This indicates that with p =3, the non-dominated solutions found through proposed approach not only

TABLE III: Comparison of Exploration Time For Finding a Low Cost Trojan Secured Datapath Solution with respect to Bacterium size p for Proposed Approach

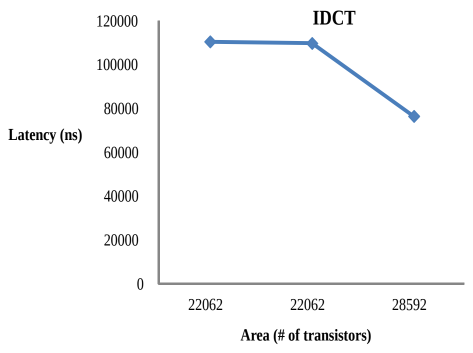
Benchmark [36], [37], [38]	Bacterium Size	Convergence Time (ms)	Exploration Time (ms)
IIR Butterworth (2^{nd} order)	3	304	688
	5	498	719
	7	837	1609
MPEG Motion Vectors	3	5018	9175
	5	8917	10003
	7	10981	16889
ARF	3	235	594
	5	675	828
	7	892	1844
WDF	3	1809	2453
	5	3568	5175
	7	5687	7344
IDCT	3	3112	8679
	5	11110	14719
	7	15628	20680
FFT	3	5261	7306
	5	7589	9160
	7	8978	12025
MESA Interpolate	3	21128	32522
	5	34560	57550
	7	56318	91818
FIR (6-tap)	3	1672	2509
	5	2491	3611
	7	1913	5335
DCT	3	3586	7375
	5	5234	8670
	7	7819	10235
GLRT	3	5634	8045
	5	8754	12415
	7	10327	17933

lies on true pareto front with better S but also nearly reaches (or reaches) the extremes of true pareto front curve. Fig. 7 shows the pareto optimal set of non dominated solutions obtained through proposed approach which is found to be exactly converging on true pareto front curve (thereby indicating that generational distance (GD) = 0). More explicitly as shown in Table IV, V and VI, GD has been found to be zero, which indicates that solution found by the proposed approach accurately converges on the true pareto front.

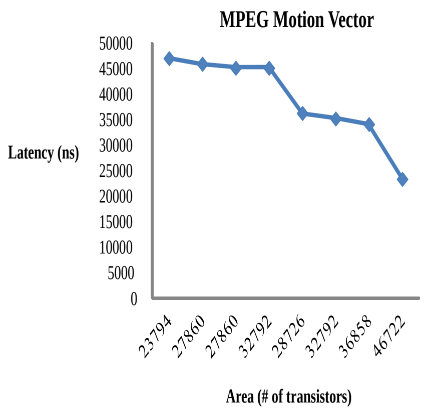
Therefore, the W_m value (combination of GD and Δ) is better for p = 3 than p = 5 or 7 for the aforesaid benchmarks. Hence the W_m value for p = 3 is underlined. On the other hand for large size



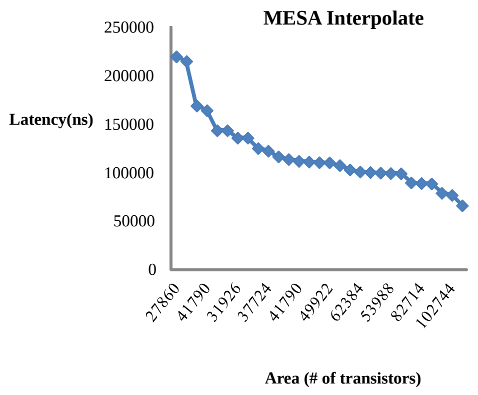
(a)



(b)



(c)



(d)

Fig. 7: Obtained Pareto optimal set of non dominated solutions through proposed approach

benchmarks such as MESA interpolation, the GD is found to be same for all population sizes, p =3, 5 or 7. However, the S and Δ is found to be lower for p = 7 than for p = 3 or 5. This indicates that at p = 7 the W_m has lowest values compared to p = 3 and 5. Since a lower value is considered better, hence a lower value of W_m at p = 7 is an indication of the fact that the quality of final solution found by the proposed approach at this bacterium population is better than solutions found at p = 3 or 5.

C. Validation of the Methodology

The proposed DSE framework generates solution for Trojan secured DMR datapath. A DMR datapath is generated for every explored solution (before fitness evaluation) which needs validation in its ability to detect Trojan present in the 3PIP. In case of HLS, the 3PIPs are the IPs/modules present in the HLS module library, which may have been imported from an outside (third party) vendor and may contain malicious logic inserted by a rogue in the third party design house. Typically, the RTL files of the modules/IPs of the library are provided by the HLS Company which it may have imported from third party vendors

TABLE IV: Quality Of Solution Explored Through Proposed Approach For p= 3.

Benchmark [36], [37], [38]	**GD**	**Spacing (S)**	**Spread (Δ)**	**Weighted Metric (W_m)**
IIR Butterworth (2^{nd} order)	0.00	0.00	0.00	0.00
MPEG Motion Vectors	0.00	0.05	0.88	0.44
ARF	0.00	0.04	0.28	0.14
WDF	0.00	0.00	0.00	0.00
IDCT	0.00	0.11	0.84	0.42
FFT	0.00	0.00	0.00	0.00
MESA Interpolate	0.00	0.08	0.93	0.46
FIR (6-tap)	0.00	0.00	0.00	0.00
DCT	0.00	0.09	0.84	0.42
GLRT	0.00	0.03	0.74	0.40

TABLE V: Quality Of Solution Explored Through Proposed Approach For p= 5

Benchmark [36], [37], [38]	**GD**	**Spacing (S)**	**Spread (Δ)**	**Weighted Metric (W_m)**
IIR Butterworth (2^{nd} order)	0.00	0.00	0.66	0.33
MPEG Motion Vectors	0.00	0.00	0.81	0.40
ARF	0.00	0.12	0.37	0.18
WDF	0.00	0.00	0.00	0.00
IDCT	0.00	0.10	0.87	0.43
FFT	0.00	0.00	0.66	0.33
MESA Interpolate	0.00	0.05	0.98	0.49
FIR (6-tap)	0.00	0.00	0.00	0.00
DCT	0.00	0.14	0.96	0.42
GLRT	0.00	0.05	0.34	0.42

as RTL files that may contain malicious/Trojan logic. The proposed approach deals with Trojan in the modules/IP that affects the computational output value. However the key is that Trojan only becomes visible at runtime (after external triggering by an adversary), thereby remaining completely obscured before that, thus impossible to be detected during RTL simulation. For the checking the detection ability of the solution generated, the presence of Trojan in module of HLS library was emulated by inserting malicious logic in the module (such as adder/ subtractor etc) of the HLS library. Trojan logic is such that only on external activation, functional alteration of the module occurs. Large set of random test sequences for the inputs were fed into the DMR datapath to identify the status (Trojan Detected/ undetected). Results indicate that in the explored solution whenever a Trojan was inserted (in any module provided a vendor);

TABLE VI: Quality Of Solution Explored Through Proposed Approach For p= 7

Benchmark [36], [37], [38]	**GD**	**Spacing (S)**	**Spread (Δ)**	**Weighted Metric (W_m)**
IIR Butterworth (2^{nd} order)	0.00	0.00	0.66	0.33
MPEG Motion Vectors	0.00	0.05	0.82	0.41
ARF	0.00	0.03	0.36	0.18
WDF	0.00	0.00	0.00	0.00
IDCT	0.00	0.17	1.04	0.52
FFT	0.00	0.00	0.66	0.33
MESA Interpolate	0.00	0.03	0.86	0.43
FIR (6-tap)	0.00	0.00	0.00	0.00
DCT	0.00	0.14	0.96	0.48
GLRT	0.00	0.03	0.86	0.52

TABLE VII: Comparison of Proposed DSE Approach with [4].

Note: W_1 and W_2 = 0.5

Benchmark [36], [37], [38]	**Solution [4]**	**Proposed Solution**	**Latency [4] (ns)**	**Latency (proposed) (ns)**	**Exploration Time (in ms) [4]**	**Proposed Exploration Time (in ms)**	**Cost of final solution [4]**	**Cost of final solution (proposed)**
IIR Butterworth (2^{nd} order)	2(+), 3(*), 1	2(+), 5(*), 0	44540	23080	15	688	-0.04	-0.15
FIR (6-tap)	5(+), 5(*), 1	6(+), 4(*), 0	45350	45890	108	2509	0.00	-0.03
MPEG Motion Vectors	3(+), 8(*), 1	2(+), 10(*), 0	45890	36240	15	9175	-0.22	-0.24
ARF	5(+), 3(*), 1	2(+), 4(*), 0	121540	152540	31	594	-0.06	-0.23
DCT	5(+), 3(*), 1	4(+), 2(*), 0	121810	88810	32	7375	-0.12	-0.21
FFT	5(+), 3(*), 1	8(+), 4(*), 0	89080	66270	31	7306	-0.07	-0.13
MESA Interpolate	10(+), 17(*), 1	3(+), 16(*), 0	74330	99350	109	32522	-0.29	-0.29
IDCT	5(+), 3(*), 1	6(+), 4(*), 0	98890	77080	31	8679	-0.09	-0.22
WDF	3(+), 3(*), 1	3(+), 2(*), 0	100970	111695	32	2453	-0.01	-0.09
GLRT	4(+), 10(*), 1	4(+), 6(*), 0	78080	109800	110	8045	-0.22	-0.23

difference in output was noticed from original and duplicate unit, indicating 100 % Trojan detection. Hence full coverage of Trojan detection was obtained through the proposed DMR allocation rules for both 'A_v' = 0, 'A_v' = 1.

D. Comparison with Related Prior Research

As seen from Table VII, the proposed approach generates substantially better quality (lower cost) results in comparison to [4]. This is because, in [4], there is no optimization performed during determination of security aware HLS solution (based on user area-delay constraint). Further, in [4] there is no supplementary option for exploration of an efficient hardware allocation procedure of vendors '(A_v)'. Furthermore, the latency of the final solution obtained through the proposed approach is lesser in most of

TABLE VIII: Comparison of Proposed DSE approach with baseline approach

Benchmark [36], [37], [38]	Area baseline approach	Area proposed approach	Latency baseline approach (ns)	Latency proposed approach	Area overhead of proposed approach compared to baseline (a.u)	Latency overhead of proposed approach compared to baseline (ns)	Cost of final solution (baseline)	Cost of final solution (proposed)
IIR Butterworth (2^{nd} order)	11472	19744	40795	23080	8272	0	-0.13	-0.15
FIR (6-tap)	22510	23322	41590	45890	812	4300	-0.05	-0.03
MPEG Motion Vectors	25846	34900	41855	36240	9054	0	-0.28	-0.24
ARF	13506	16198	120530	89890	2692	0	-0.19	-0.23
DCT	17574	16610	120530	153540	0	33010	-0.26	-0.21
FFT	17574	29152	90530	66270	11578	0	-0.15	-0.13
MESA Interpolate	62296	60254	81590	99350	0	17760	-0.31	-0.29
IDCT	17574	31582	100795	77080	14008	0	-0.29	-0.22
WDF	11038	13064	102650	111695	2026	9045	-0.17	-0.09
GLRT	32816	24962	91060	109800	0	18740	-0.22	-0.23

the cases compared to [4] because of datapath and vendor allocation optimization, except in case of WDF and MESA where the final solution in [4] has been scheduled with higher resource count. However, the runtime consumed by [4] is lower than proposed as we perform multi-dimensional (datapath resources and vendor allocation) exploration within HLS to find optimal final solution, which [4] is unable to due to lack of multi-dimensional optimization feature. Further, Table VIII shows the comparative results of the proposed approach with the baseline approach, in terms of cost, hardware area, delay and respective overheads. The proposed approach generates solution with minimal overhead (with no overhead in some cases also) besides being Trojan secured (due to allocation of two distinct vendor in the DMR design). However, the baseline DMR occupies lower area as it does not impose any security allocation rules and uses only one type of vendor (thereby incapable of providing Trojan detection/security) unlike proposed approach.

VII. Conclusions and Future Directions of Research

This paper presented a novel low cost Trojan security aware HLS methodology. The proposed DSE approach provides a significant reduction in the cost of security aware HLS solution in comparison to similar prior work [4]. Our future research aims to extend the proposed work by developing methodology that can explore optimized Trojan secured solution for multi-loop based applications. Several intricacies requires to be addressed such as optimization of high level transformation along with optimization of DMR structure.

Acknowledgments

This archival journal paper is based on our preliminary conference presentation of this research [40].

References

[1] R. Karri, J. Rajendran, K. Rosenfeld, and M. Tehranipoor, "Trustworthy Hardware: Identifying and Classifying Hardware Trojans," *Computer*, vol. 43(10), pp. 39–46, 2010.

[2] M. Banga and M. Hsiao, "A Novel Sustained Vector Technique for the Detection of Hardware Trojans," in *Proceedings of the 22nd VLSI Design Conference*, 2009, pp. 327–332.

[3] X. Zhang and M. Tehranipoor, "Case Study: Detecting Hardware Trojans in third-party Digital IP Cores," in *Proceedings of the IEEE Symposium on Hardware-Oriented Security and Trust (HOST)*, 2011, pp. 67–70.

[4] J. Rajendran, H. Zhang, O. Sinanoglu, and R. Karri, "High-level synthesis for security and trust," in *Proceedings of the IEEE 19th International On-Line Testing Symposium (IOLTS)*, 2013, pp. 232–233.

[5] Y. Jin and Y. Makris, "Hardware Trojan detection using path delay fingerprint," in *Proceedings of the IEEE Symposium on Hardware-Oriented Security and Trust*, 2008, pp. 51–57.

[6] D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi, and B. Sunar, "Trojan Detection using IC Fingerprinting," in *Proceedings of the IEEE Symposium on Security and Privacy*, 2007, pp. 296–310.

[7] X. Cui, K. Ma, L. Shi, and K. Wu, "High-level synthesis for run-time hardware Trojan detection and recovery," in *Proceedings of the 51st ACM/IEEE on Design Automation Conference (DAC)*, 2014, pp. 1–6.

[8] X. Wang, H. Salmani, M. Tehranipoor, and J. Plusquellic, "Hardware Trojan Detection and Isolation Using Current Integration and Localized Current Analysis," in *Proceedings of the IEEE International Symposium on Defect and Fault Tolerance of VLSI Systems (DFTVS)*, 2008, pp. 87–95.

[9] S. Narasimhan, D. Du, R. Chakraborty, S. Paul, F. Wolff, C. Papachristou, K. Roy, and S. Bhunia, "Multiple-parameter side-channel analysis: A non-invasive hardware Trojan detection approach," in *Proceedings of the IEEE Symposium on HOST*, 2010, pp. 13–18.

[10] S. Saha, R. S. Chakraborty, S. S. Nuthakki, Anshul, and D. Mukhopadhyay, *Cryptographic Hardware and Embedded Systems – CHES 2015: 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, ch. Improved Test Pattern Generation for Hardware Trojan Detection Using Genetic Algorithm and Boolean Satisfiability, pp. 577–596.

[11] S. S. Ali, R. S. Chakraborty, D. Mukhopadhyay, and S. Bhunia, "Multi-level attacks: An emerging security concern for cryptographic hardware," in *Design, Automation Test in Europe*, March 2011, pp. 1–4.

[12] L. Lin, W. Burleson, and C. Paar, "MOLES: Malicious off-chip leakage enabled by side-channels," in *IEEE/ACM International Conference on Computer-Aided Design - Digest of Technical Papers*, Nov 2009, pp. 117–122.

[13] B. Schafer and K. Wakabayashi, "Design Space Exploration Acceleration Through Operation Clustering," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29(1), pp. 153–157, Jan 2010.

[14] S. Xydis, K. Pekmestzi, D. Soudris, and G. Economakos, "Compiler-in-the-loop Exploration During Datapath Synthesis for Higher Quality Delay-area Trade-offs," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 18(1), pp. 11:1–11:35, 2013.

[15] S. Gupta, N. Savoiu, N. Dutt, R. Gupta, and A. Nicolau, "Using global code motions to improve the quality of results for high-level synthesis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 23(2), pp. 302–312, 2004.

[16] S. Xydis, G. Palermo, V. Zaccaria, and C. Silvano, “SPIRIT: Spectral-Aware Pareto Iterative Refinement Optimization for Supervised High-Level Synthesis,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34(1), pp. 155–159, 2015.

[17] H.-Y. Liu and L. Carloni, “On learning-based methods for design-space exploration with High-Level Synthesis,” in *Proceedings of the 50th ACM/IEEE DAC*, 2013, pp. 1–7.

[18] B. Schafer, “Hierarchical High-Level Synthesis Design Space Exploration with Incremental Exploration Support,” *IEEE Embedded Systems Letters*, vol. 7(2), pp. 51–54, 2015.

[19] 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(3), pp. 213–229, 2006.

[20] S. P. Mohanty, N. Ranganathan, and V. Krishna, “Datapath Scheduling Using Dynamic Frequency Clocking,” in *Proceedings of the IEEE Computer Society Annual Symposium on VLSI*, 2002, pp. 58–63.

[21] P. Coussy, D. Gajski, M. Meredith, and A. Takach, “An Introduction to High-Level Synthesis,” *IEEE Design Test of Computers*, vol. 26(4), pp. 8–17, 2009.

[22] S. P. Mohanty, *Nanoelectronic Mixed-Signal System Design*. McGraw-Hill Education, 2015.

[23] S. P. Mohanty, N. Ranganathan, and S. K. Chappidi, “Peak Power Minimization Through Datapath Scheduling,” in *Proceedings of the IEEE Computer Society Annual Symposium on VLSI*, 2003, pp. 121–126.

[24] S. Bhadauria and A. Sengupta, “Adaptive bacterial foraging driven datapath optimization: Exploring power-performance tradeoff in high level synthesis,” *Applied Mathematics and Computation*, vol. 269, pp. 265 – 278, 2015.

[25] G. D. Micheli, *Synthesis and Optimization of Digital Circuits*, 1st ed. McGraw-Hill Higher Education, 1994.

[26] F. Ferrandi, P. L. Lanzi, D. Loiacono, C. Pilato, and D. Sciuto, “A multi-objective genetic algorithm for design space exploration in high-level synthesis,” in *Proceedings on IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 2008, pp. 417–422.

[27] Z. Zeng, R. Sedaghat, and A. Sengupta, “A framework for fast design space exploration using fuzzy search for VLSI computing architectures,” in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, 2010, pp. 3176–3179.

[28] D. S. H. Ram, M. C. Bhuvaneswari, and S. S. Prabhu, “A Novel Framework for Applying Multiobjective GA and PSO Based Approaches for Simultaneous Area, Delay, and Power Optimization in High Level Synthesis of Datapaths,” *VLSI Des.*, vol. 2012, 2012.

[29] J. Gallagher, S. Vigraham, and G. Kramer, “A family of compact genetic algorithms for intrinsic evolvable hardware,” *IEEE Transactions on Evolutionary Computation*, vol. 8(2), pp. 111–126, 2004.

[30] N. Reynders and W. Dehaene, “A 190mV supply, 10MHz, 90nm CMOS, pipelined sub-threshold adder using variation-resilient circuit techniques,” in *Proceedings of the IEEE Asian Solid State Circuits Conference (A-SSCC)*, 2011, pp. 113–116.

[31] J. Crop, S. Fairbanks, R. Pawlowski, and P. Chiang, “150mV sub-threshold Asynchronous multiplier for low-power sensor applications,” in *Proceedings of the International Symposium on VLSI Design Automation and Test (VLSI-DAT)*, 2010, pp. 254–257.

[32] A. K. Kumar, D. Somasundareswari, V. Duraisamy, and M. Pradeepkumar, “Low power multiplier design using complementary pass-transistor asynchronous adiabatic logic,” *International Journal on Computer Science and Engineering*, vol. 2(7), pp. 2291–2297, 2010.

[33] D. D. Gajski, N. D. Dutt, A. C.-H. Wu, and S. Y.-L. Lin, *High-level Synthesis: Introduction to Chip and System Design*. USA: Kluwer Academic Publishers, 1992.

[34] K. Passino, “Biomimicry of bacterial foraging for distributed optimization and control,” *IEEE Control Systems*, vol. 22(3), pp. 52–67, 2002.

[35] Y. Liu and K. Passino, “Biomimicry of social foraging bacteria for distributed optimization: models, principles, and emergent behaviors,” *Journal of Optimization Theory and Applications*, vol. 115(3), pp. 603–628, 2002.

[36] S. P. Mohanty, N. Ranganathan, E. Kougianos, and P. Patra, *Low-power high-level synthesis for nanoscale CMOS circuits*. Springer Science & Business Media, 2008.

[37] A. Sengupta, R. Sedaghat, and Z. Zeng, “A high level synthesis design flow with a novel approach for efficient design space exploration in case of multi-parametric optimization objective,” *Microelectronics Reliability*, vol. 50(3), pp. 424 – 437, 2010.

[38] [Online]. Available: http://express.ece.ucsb.edu/benchmark/

[39] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*. USA: John Wiley & Sons, Inc., 2001.

[40] A. Sengupta and S. Bhadauria, “Untrusted Third Party Digital IP Cores: Power-Delay Trade-off Driven Exploration of Hardware Trojan Secured Datapath During High Level Synthesis,” in *Proceedings of the 25th Great Lakes Symposium on VLSI*, 2015, pp. 167–172.