TL-HLS: Methodology for Low Cost Hardware Trojan Security Aware Scheduling with Optimal Loop Unrolling Factor during High Level Synthesis

Anirban Sengupta, Saumya Bhadauria, and Saraju P. Mohanty, *Senior Member, IEEE*

***Abstract*—Security against hardware Trojan that is capable to change the computational output value is accomplished by employing Dual Modular Redundant (DMR) schedule during High Level Synthesis (HLS). However, building a DMR for Trojan security is non-trivial and incurs extra delay and hardware. This paper proposes a novel HLS methodology for constraint driven low cost hardware Trojan secured DMR schedule design for loop based Control Data Flow Graphs (CDFGs). Proposed approach simultaneously explores an optimal schedule and optimal loop unrolling factor (U) combination for a low cost Trojan security aware DMR schedule. As a specific example, proposed low cost Trojan secured high level synthesis approach (TL-HLS) relies on Particle Swarm Optimization (PSO) algorithm to explore optimized Trojan secured schedule with optimal unrolling that provides security against specific Trojan (causing change in computational output) within user provided area and delay constraints. The novel contributions of the paper are, firstly an exploration of a low cost Trojan security aware HLS solution for loop based CDFGs; secondly, proposed encoding scheme for representing design solution comprising candidate schedule resources, candidate loop unrolling factor and candidate vendor allocation information; thirdly, a process for exploring the a low cost vendor assignment that provides Trojan security; finally, experimental results over the standard benchmark that indicates an average reduction in final cost of $\sim$ 12 % compared to recent approach.**

***Index Terms*—Trojan, Intellectual Property, High-Level Synthesis, Loop Unrolling, Particle-Swarm Optimization**

I. Introduction and Motivation

HARDWARE Trojan's are malicious logic/hardware components embedded by a rogue in order to induce malfunctioning of Integrated Circuits (ICs). Globalization of System on Chips (SoC) design and manufacturing has raised serious concerns on the security and trustworthiness of the embedded third party intellectual property (3PIPs) [1] [2], [3] [4]. Fig. 1 shows the typical design cycle involving third party vendors and in house system design. In this figure, the first block indicating datapath components or third party Register Transfer Level (RTL) library is considered untrustworthy, while the remaining blocks(in house system designer and foundry) are considered trustworthy. The SoCs may involve analog and digital components at the same time for cost and performance trade-offs [5]. However, the focus of this current paper is the digital components when their design exploration are performed at the architecture level.

A. Sengupta is with the Department of Computer Science and Engineering, Indian Institute of Technology Indore, Email: asengupt@iiti.ac.in. S. Bhadauria is with the Department of Computer Science and Engineering, Indian Institute of Technology Indore. S. P. Mohanty is with the Department of Computer Science and Engineering, University of North Texas, Email: saraju.mohanty@unt.edu.

During designing, an IP may be corrupted (by an adversary), by inserting hardware Trojan into it. During HLS, it should be ensured that any possible infection of 3PIP is detectable, thereby generating a trustworthy design. 3PIPs are the IPs present in the module library of a HLS tool, which may have been imported from an outside (third party) vendor and is considered untrustworthy. The key is that any third party module may contain malicious Trojan logic inserted by an adversary in the third party design house. This unique possibility of Trojan insertion in third party module of a HLS library was highlighted and discussed in recent literature [6]. In a typical case, the HLS company provides the RTL files of the modules/IPs of the library, which might have been imported from third party vendors. These RTL files of a module may contain Trojan logic as discussed above. Therefore, detection strategy during HLS for possible Trojan in 3PIP/module (present in the library) requires attention [6]. However, the detection process of hardware Trojan during HLS mandates additional hardware, which upon deployment may not abide by the user area constraint provided. Further, incorporating additional logic for Trojan detection during HLS also results in extra delay for processing output, which again may not abide by the user delay constraints specified. It therefore becomes mandatory to consider the effect of extra delay and hardware cost during Design Space Exploration (DSE) in HLS.

The aforesaid process is applicable for Data Flow Graphs (DFGs) in HLS. However, in the context of loop based CDFGs, the exploration of a low cost hardware Trojan security aware schedule does not suffice alone, due to involvement of an additional variable called "loop unrolling factor". Loop unrolling plays an important significance in dictating the final area and delay of a design. Therefore, during the design of a Trojan security aware schedule for CDFGs, simultaneously

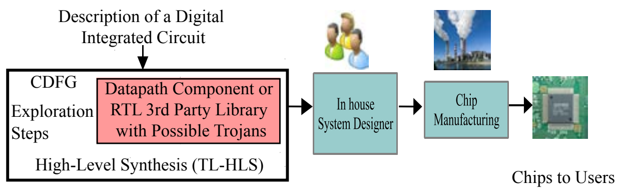

Description of a Digital Integrated Circuit
CDFG Exploration Steps
Datapath Component or RTL 3rd Party Library with Possible Trojans
High-Level Synthesis (TL-HLS)
In house System Designer
Chip Manufacturing
Chips to Users

Fig. 1: Proposed TL-HLS in the Chip Development Flow.

considering the effects of loop unrolling on its area-delay tradeoff is equally critical. This paper resolves the aforesaid problem by proposing a methodology for simultaneous DSE of low cost Trojan security aware DMR schedule and optimal loop unrolling factor, that satisfies the user area-delay constraints provided. DMR 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 resource constraints, which is iteratively obtained through PSO process. Further, the hardware overhead of proposed Trojan secured DMR (i.e. steering logic and internal buffers) is also optimized in our approach.

The rest of the paper is organized in the following manner: Section II discusses the novelty aspects of the current paper. Section III describes a scenario of Hardware Trojan in 3PIP. Further, section IV discusses the approaches developed so far, to handle hardware Trojans. Section V and VI presents the evaluation models and proposed methodology. The results are further described in section VII.

II. Novel contributions of the paper

The novel contributions of the current paper for advancement of state-of-art in HLS for trusted hardware are the followings:

1) Approach for simultaneous exploration of **low cost Trojan security aware DMR schedule and optimal loop unrolling factor** during HLS that abides by the user area-delay constraints provided. This work aims at providing low cost Trojan detection and not recovery (unlike in [7], [8]). Therefore, the proposed technique in this paper is not extended for Triple Modular Redundant (TMR) designs to keep a focused scope.
2) Vendor allocation exploration procedure during proposed PSO driven DSE process. This encoding explores the most efficient vendor assignment that provides Trojan detection.
3) Encoding scheme for representing design solutions that comprises of candidate schedule resources, candidate loop unrolling factor and candidate vendor allocation information.
4) Technique for area-delay tradeoff using PSO during exploration of a low cost Trojan security aware DMR schedule. DMR 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 (unlike in [6]) as it is designed based on resource constraints (fed through PSO process). Further, the overhead of proposed secured DMR (i.e. steering logic and internal buffers) is also optimized in our approach.
5) Model for delay estimation of a Trojan secured DMR CDFG.

Why Security Aware HLS? Digital ICs go through design exploration at various stages of abstraction in consistence with divide and conquer philosophy, to handle the complexity at the same time to control the non-recurrent design cost. There is no doubt that, HLS is way matured starting from performance optimization, power optimization, to process variation optimization, conducted at the architecture level [9], [10]. In the current era of smart mobile computing, IP based designs are the need of the hour to meet the time to market demand and HLS techniques can play more crucial role for the design engineers [5]. So, a natural progression of HLS research is to explore security awareness along the other challenges at the architecture level through HLS. The key idea of low cost Trojan security aware HLS solution for loop based CDFGs has been depicted in the Fig. 2.

Threat Model for Proposed Research: The paper targets Trojans in 3PIPs that only change computational output value (and produces no other impact). This paper doesn't target class of Trojans that steal information/data. 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. In proposed approach the foundry and the in-house system designer are considered to be trustworthy in the IC design flow. In other words, in our proposed approach only the third party vendor (design house) is considered untrustworthy. This indicates an adversary or rogue designer only in the third party house can manipulate the IP (as the entire IPs (examples shown in Fig. 3) are received from the third party for subsequent design integration).

III. Hardware Trojan in 3PIP Module: An Example

Consider the following scenario, which explains the hardware Trojan problem in 3PIPs and reason for hardware security during HLS: an untrusted IP vendor/supplier may deliberately insert Trojan logic into a module/IP (used as a component in the HLS library). The Trojan logic remains ineffective, until triggered by a rogue and therefore is difficult to be detected during security review by in-house design team. This is due to reason that, during normal conditions (when not

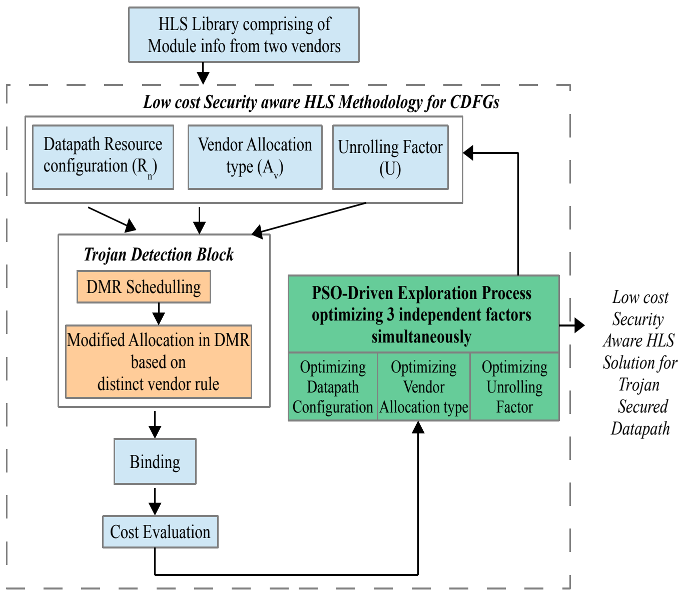

HLS Library comprising of Module info from two vendors
Low cost Security aware HLS Methodology for CDFGs
Datapath Resource configuration (R_n)
Vendor Allocation type (A_v)
Unrolling Factor (U)
Trojan Detection Block
DMR Schedulling
Modified Allocation in DMR based on distinct vendor rule
PSO-Driven Exploration Process optimizing 3 independent factors simultaneously
Optimizing Datapath Configuration
Optimizing Vendor Allocation type
Optimizing Unrolling Factor
Binding
Cost Evaluation
Low cost Security Aware HLS Solution for Trojan Secured Datapath

Fig. 2: Key idea behind low cost Trojan secured schedule during HLS

Note: A possible case of an infected IP present within the module library of a HLS tool. When select (S) = 1 is triggered by a rogue (controlled externally), then, Trojan blocks get activated. Until triggered, Trojan remains dormant in the system and circuit behaves normally. Red colored wires/blocks indicate malicious Trojan logic secretly inserted by an adversary in the third party design house.

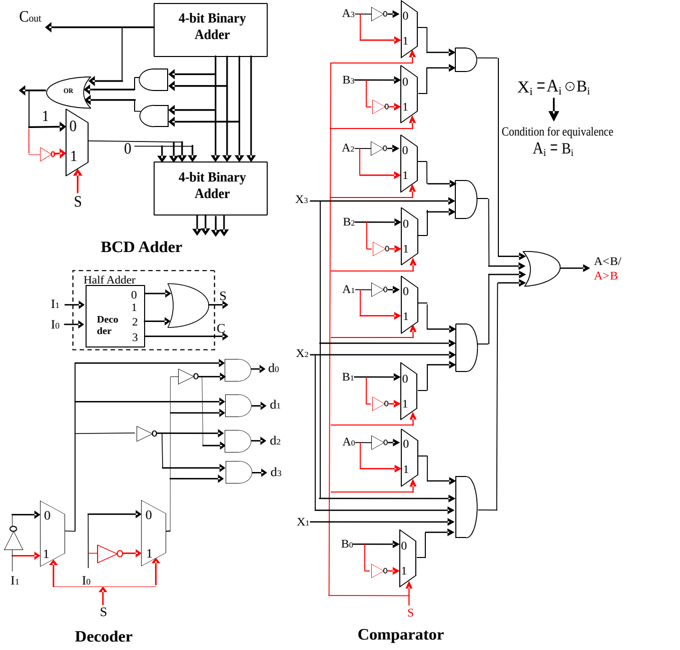


Fig. 3: Infected IP present as module in library of a HLS tool.

triggered), the logic behaves like a functionally correct IP. Typically, the HLS company provides the RTL files of the modules/IPs of the library that may contain malicious/Trojan logic, which might have been imported from third party vendors. The proposed approach deals with hardware Trojan in the modules/IP that changes its computational value. However the key is that Trojan only becomes visible at runtime (after triggering by an adversary), thereby remaining completely ineffective before, thus being difficult to detect during RTL simulation/other lower level tests. Additionally, for the case when the Trojan is 'always on' in an IP, it is likely that most of the outputs reflect no change during RTL simulation (except a few). Therefore, without detailed examination, the Trojan may sometimes go unnoticed. This is applicable for both small and large size modules present in HLS library. This is because, even small modules are activated by an adversary when deployed in real time situation to perform computational error. Even an exhaustive examination (through RTL simulation, physical inspection etc.) for small modules makes it difficult for detection as in offline situation it behaves like a functionally correct IP. The next paragraph outlines in details the reasons why this type of Trojan in IP is difficult to detect through normal examination. Fig. 3 shows an instance of Trojan within a 3PIP/module present inside the library of a HLS tool.

Here, few examples of possible Trojan logic in components (BCD adder, decoder, digital comparator, multiplier etc.) of a HLS tool library are shown, which on triggering changes the normal functional output. The triggering of the Trojan can be achieved through numerous possibilities such as, activation through Trojan time bomb (FSM counter), sensing a specific design signal, external antenna etc. Regardless of where the component is used, activation of the Trojan logic is bound to impact the computational output value of the entire system. This is because, the Trojan that we handle in our work only intends to affect the computational output value of the system (that is, change the digital value), and not steal/leak any secret information. Therefore, knowledge of the architecture of the system is not required by the attacker to exploit hardware Trojan and affect computational output value. Such Trojans are difficult to be detected with the detection techniques applied at lower levels of abstraction such as side channel analysis [11] and RTL simulation as described below. This is because, firstly, Trojan logics by design are typically triggered under very specific predefined conditions (such as through FSM counter value, sensing a specific design signal, external antenna, etc.) which makes them unlikely to be activated and detected during normal functional verification by test vectors [12]. Normally, in the system RTL code, only the complete design is functionally verified; secondly, in spite of functional verification check (RTL simulation), the IP will perform normally/correctly due to being dormant at that stage. It is only activated by an adversary when deployed in real time situation to perform functional failure. Thirdly, because there is no trustworthy golden IP model, therefore, the detection of Trojan in a 3PIP during HLS is not possible. Moreover, detection by physical inspection and reverse engineering are very difficult and costly owing to the complexity involved and nanometer IC feature size. Since Trojans are intelligently inserted in a particular portion of the design and may be activated by a single triggering signal, therefore, reverse engineering usually doesn't guarantee detection of Trojan. This is because, detection of Trojans during reverse engineering depends on its accuracy and efficiency, thus making the detection difficult. Additionally, tests used for detecting manufacturing faults (such as, stuck at faults, delay faults and bridging faults) cannot guarantee detection of such Trojans. Finally, detection of such Trojan using analysis of parametric signals is considered ineffective [12] due to decrease in physical feature size because of evolution in technology.

Direct conversion into a register transfer level structure from its corresponding behavioral description (CDFG) is accomplished through HLS which involves the process of DSE that includes evaluation of alternative candidate design solutions based on objectives such as area and delay [13]–[23]. During this process, multiple sub-processes participate such as scheduling, allocation and binding [16]. The process of DSE gets convoluted with the involvement of auxiliary variable called loop unrolling factor for CDFG applications as it adds an extra dimension to explore based on conflicting user constraints of area and delay. Moreover, the consideration of the aforesaid variables during DSE of a trusted hardware is extremely complex as discussed in Section I.

IV. Related Prior Research

No effort has been made on designing a low cost Trojan security aware HLS that considers optimization of schedule and loop unrolling (based on user constraints). There have been few promising work that is based on code-coverage analysis, property checker which checks whether an IP satisfies those

TABLE I: Proposed Nomenclature of the Current Paper

Notations	Definitions	Notations	Definitions
p	Population size	D	Total available resource types
X_i	Resource set of a particular particle solution with unrolling and allocation procedure information	$\overrightarrow{R_n}$	Resource array
X_i^+	New particle position of i^{th} particle	P_v	Vendor allocation procedure type
X_{gb}	Global best particle positions	U	unrolling factor
X_{lb_i}	Local best particle position for i^{th} particle	V_j	Type of vendor
N(R_1),..N(R_d)	Number of instances of resource type '1',..'d'	$A(R_i^{V_j})$	Area of a resource type (R_i) corresponding to vendor (V_j)
$C_f(X_i)$	Cost of particle with resource set X_i	A_T^{DMR}	Total area of a DMR design
n and n'	Maximum value of node	A_{cons}	User specified area constraint
V_{d_i}	Velocity of i^{th} particle in d^{th} dimension	A_{max}^{DMR}	DMR solution with maximum area in the design
$V_{d_i}^+$	New velocity of i^{th} particle in d^{th} dimension	T_E^{DMR}	Total execution time of a DMR design
$V_{d_i}^{max}$	maximum velocity of i^{th} particle in d^{th} dimension	T_{cons}	User specified execution time constraint
ω	Inertia weight	T_{max}^{DMR}	DMR solution with maximum time in the design space
R_{d_i}	Resource value or 'U' value of particle X_i in d^{th} dimension of i^{th} particle	$R_{d_{gb}}$	resource Value of X_{gb} in d^{th} dimension
$R_{d_i}^+$	New Resource value or 'U' value of particle X_i in d^{th} dimension of i^{th} particle	$R_{d_{lb_i}}$	Resource value of X_{lb_i} (local best position) in d^{th} dimension of i^{th},particle
b_1, b_2	Acceleration coefficients which balances the effect of cognitive and social factor during exploration	$R_i^{V_j}$	Number of instances utilized from Vendor V_j for a resource type R_i
r_1, r_2	Random numbers providing stochasticity	C_{first}^{DMR}	Number of CS required to execute first iteration of the $CDFG^{DMR}$
C_{body}^{DMR}	Number of CS required to execute loop body of $CDFG^{DMR}$ once	C_T^{DMR}	Total CS required to execute the loop of $CDFG^{DMR}$
I	Maximum number of iteration (loop count)	Δ	Delay of one Control Step (CS) in nanoseconds

properties. In code-coverage analysis [24], list of suspicious signals are identified in the RTL design. Those signals are identified as suspicious signals which remain stable during coverage analysis. The motivation behind this concept is that Trojans normally do not change states until triggered. Then the approach adds test vectors to activate uncovered parts of the design. Though this is beneficial however, it results in huge verification time. Additionally, redundant circuit removal is applied since these tend to stay at the same logic level during verification. This is obtained by techniques where scan chains are inserted into the post synthesis results (gate level netlist) and analysed for stuck-at-faults through test patterns generated by Automatic Test Pattern Generation (ATPG) process. An untestable stuck-at-fault is likely to be a redundant logic. Thus when an ATPG identifies a stuck-at- 1/0 fault as untestable, the faulty net can be replaced by logic 1/0 in the gate level netlist. All circuits driving fault nets will be removed as well. Equivalence analysis [25] (like done in case of faults) may be applied on the suspicious signal list in order to reduce the number of signals, however, it incurs runtime overhead. Further, in the lists of suspicious signals not all signals are Trojans. Moreover, in this approach the quantity of suspicious signals keeps increasing with the increase in the complexity of Trojan inserted. Finally, it is difficult to achieve 100% code-coverage for all IPs.

In [26], authors developed an efficient dummy flip-flop insertion procedure to increase the transition probability of nets when it is lower than a specific probability threshold. The transition probability was modeled using geometric distribution. Furthermore, a number of approaches have been proposed for Trojan detection at lower levels of chip design [12], [27]–[30]. Besides, [20], which has dealt with Trojan detection at system level however with no effort on exploring an optimized Trojan secured schedule with optimal loop unrolling (based on user constraints) that is capable of providing system level protection. In [30], authors detect Trojans by generating a multiple supply transient current integration methodology. Once the detection of Trojan in a chip is done, an isolation process is initiated to isolate the Trojans within the IC. In [27], the detection process uses chip characteristics like path delay and power consumption (of the manufactured chips) and compares them with the expected values to identify Trojan's. Moreover, the approach, in [28] is capable of detecting malicious hardware modifications in the presence of large process variation induced noise. The detection strategy relies on the fact that, Trojan's once inserted change the design parameters of the circuit.

Authors in [6], [31] have dealt with Trojan detection at system level. Authors in [6], [31], adopted a concurrent error detection (CED) approach for Trojan detection. The approaches use a diverse set of 3PIP vendors for Trojan identification. Authors have only targeted DFGs and therefore do not handle loop based CDFG. Moreover, the approaches [6], [31] do not have techniques for exploration of efficient vendor allocation procedure for hardware in DMR system since it affects the final area-delay of the solution [32]. As a result of which an inferior quality solution (higher cost) is generated. However, the proposed approach only focuses on Trojan detection and not on preventing its activation (as in [6]). Therefore, the cost comparisons reported for both approaches are with respect to Trojan detection only. The proposed Trojan secured HLS is different than fault secured HLS approach [33] in terms of the threat model, number of vendors required and the type of security constraints imposed. For example, the threat model considered in Trojan secured HLS is malicious logic inserted by an adversary secretly which remains dormant until triggered. However, fault secured HLS considers resiliency against transient faults. Further, Trojan secured HLS requires at least 2 vendors to provide distinctness in the DMR output, however, its counterpart only requires one vendor for security. Finally, the Trojan secured HLS demands distinct vendor allocation to hardwares of sister operations in DMR while its counterpart requires distinct hardware assignments to sister operations in DMR only (where even both distinct hardware units may be from same vendor).

V. SECURITY AWARE HLS: FORMULATION AND MODELS

The aim of proposed approach is to explore the design space of a Trojan Secured DMR schedule comprising of candidate solutions for DMR schedule resource configurations (architecture), candidate loop unrolling factor and candidate vendor assignment procedure, all hybrid encoded (as discussed in the upcoming sections).

A. Problem Definition

Determine a solution for Trojan secured DMR schedule, optimal $\{X_i\}$ = {N(R_1), N(R_2),..N(R_D), U P_v}, while exploring the design space of a given CDFG and satisfying conflicting user constraints and minimizing the overall cost. The problem can be formulated as follows:

Minimize: Hybrid *Cost(A_T^{DMR}, T_E^{DMR})*, for Optimal $\{X_i\}$, *Subjected to:* $A_T^{DMR} \leq A_{cons}$ and $T_E^{DMR} \leq T_{cons}$ and hardware Trojan security. The variables are explained in Table. I. 'P_v' is the vendor allocation procedure capable of holding only binary value (where 'P_v' = '1' indicates all operations of a specific unit being strictly assigned to resources of same vendor type e.g. all operations of original unit strictly assigned to same vendor 'V_1' and all operations of duplication to same vendor 'V_2'; while 'P_v' = '0' indicates alternate vendor assignment to operations in a CS of a unit). Hence, the variable 'P_v' is crucial for Trojan secured schedule optimization, as both 'P_v' = 0 and 'P_v' = 1, provide vendor distinctness in DMR design resulting in Trojan security. However, they have different impact on final delay of the design.

B. Proposed Evaluation Models

In the proposed work, each particle position represents a hybrid encoding of candidate solutions for schedule resource configurations (X_i), loop unrolling factor (U) and vendor assignment procedure information (P_v) in the design space.

1) Proposed area model: Total area consumed (A_T^{DMR}) by a resource set is given by:

$$A_T^{DMR} = \sum_{j=1}^{2}\sum_{i=1}^{m}\left(A\left(R_i^{V_j}\right)\times R_i^{V_j}\right), \tag{1}$$

where, the variables are explained in Table I. Note: The area component includes area due to functional resources, interconnect units (mux and demux), comparator (for error detection) as well as overhead incurred from internal buffering (during temporary storage of operation output in DMR scheduling). The area evaluated is with respect to module library generated with CMOS 90nm technology node [34], [35].

2) Proposed Delay Model: The delay is evaluated after derivation of the delay model using the following two cases:

Case 1: When 'U' is equal to one (indicates no unrolling) then: Total # of CS = # of CS required executing loop body once * number of duplicate iterations of loop body:

$$C_T^{DMR} = C_{body}^{DMR} * \alpha = \left(C_{body}^{DMR} * \left[\frac{I}{U}\right]_{floor}\right) \tag{2}$$

where, $\alpha = [I/U]^{floor}$ Since U = I,

$$C_T^{DMR} = C_{body}^{DMR} * I \tag{3}$$

where, C_T^{DMR} the variables are explained in Table I. *Note: eqn. 2 is valid when 'U' evenly divides the loop count(I) (i.e. I % U == 0).*

Case 2: When 'U' unevenly divides I: In such a case, $ImodU$ iterations will be executed sequentially, therefore, the total no. of CSs is:

$$C_T^{DMR} = \left(C_{body}^{DMR} * \left[\frac{I}{U}\right]^{floor}\right) + (ImodU) * C_{first}^{DMR} \tag{4}$$

{CSs for unrolled loop} {CSs for sequential loop},

The variables are explained in Table I. Hence the execution time for the system is calculated as:

$$T_E^{DMR} = \Delta * C_T^{DMR} \tag{5}$$

where, 'Δ' is the delay of one CS in nanoseconds. The delay model is based on the latency values of each functional hardware described at 90nm technology scale [36].

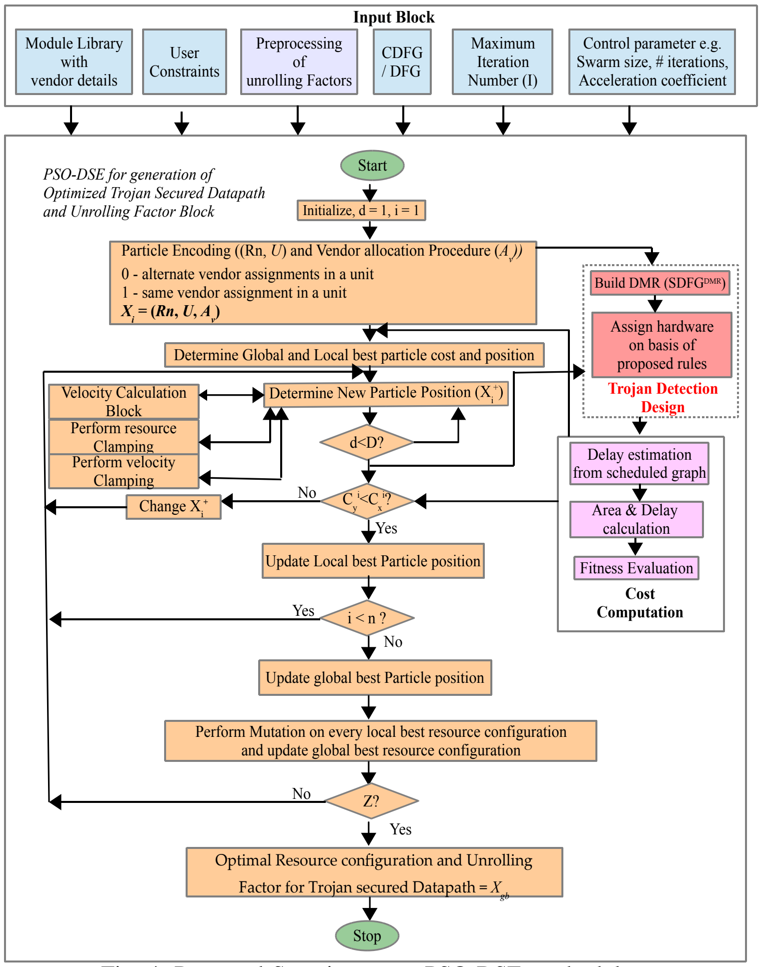
3) Fitness function: The fitness function (considering execution time and area consumption of a solution) is inspired from [37] [38] and is formulated as:

$$C_f(X_i) = W_1\left(\frac{A_T^{DMR} - A_{cons}}{A_{max}^{DMR}}\right) + W_2\left(\frac{T_E^{DMR} - T_{cons}}{T_{max}^{DMR}}\right). \tag{6}$$

The variables are explained in Table I. W_1 and W_2 are the user defined weights both kept at 0.5 during exploration to provide equal preference. The equation above is a penalty graded cost function which incorporates the effect of user constraints (A_{cons} and T_{cons}) during fitness evaluation of DMR system. Further, it considers the maximum values of area and execution time (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 (specified in section V-A), hence a higher negative value indicates a more desirable solution.

VI. PROPOSED SECURITY AWARE DSE IN HLS

For detection of hardware Trojan in 3PIPs that only change computational output value (and produces no other impact), minimum two distinct third party vendors are required. The concept of this was introduced in [6]. Even if the IPs from two different vendors have dissimilar timing, but functional similarity of two distinct IPs, allow for comparison at the DMR output. In the proposed approach we only require two distinct vendors for generating a Trojan secured schedule. We do not require optimization of number of vendors, however we optimize the cost of solution by regulating internal allocation process of two distinct vendor within DMR schedule through a variable 'P_v' during exploration. Imposing diverse set of 3PIP vendors as security constraints during allocation step for similar operations in DMR design during HLS provides



Input Block
Module Library with vendor details
User Constraints
Preprocessing of unrolling Factors
CDFG / DFG
Maximum Iteration Number (I)
Control parameter e.g. Swarm size, # iterations, Acceleration coefficient
PSO-DSE for generation of Optimized Trojan Secured Datapath and Unrolling Factor Block
Start
Initialize, d = 1, i = 1
Particle Encoding ((Rn, U) and Vendor allocation Procedure (A_v))
0 - alternate vendor assignments in a unit
1 - same vendor assignment in a unit
X_i = (Rn, U, A_v)
Build DMR (SDFG^DMR)
Assign hardware on basis of proposed rules
Trojan Detection Design
Determine Global and Local best particle cost and position
Velocity Calculation Block
Perform resource Clamping
Perform velocity Clamping
Determine New Particle Position (X_i^+)
d<D?
Change X_i^+
No
C_y^i<C_x^i?
Yes
Delay estimation from scheduled graph
Area & Delay calculation
Fitness Evaluation
Cost Computation
Update Local best Particle position
Yes
i < n ?
No
Update global best Particle position
Perform Mutation on every local best resource configuration and update global best resource configuration
No
Z?
Yes
Optimal Resource configuration and Unrolling Factor for Trojan secured Datapath = X_gb
Stop

Fig. 4: Proposed Security aware PSO-DSE methodology.

detection of malicious output. A simple demonstrative example is shown in Fig. 5 where y is the primary input. Let us consider a scenario: Multiplier of vendor V1 is malicious: In such case assuming the multiplier maliciously performs division. The subsequent output from original unit is $1/y$, however, the duplicate output is y^3, indicating Trojan detection (due to difference in computation value). As discussed in [6] for cases where no sub-IP exists, it is highly unlikely that different Trojans in different 3PIPs will produce identical wrong outputs. In other words, when no sub-IP exists, the chance of IP from both third party vendors being Trojan infected or carrying same Trojan logic is low. In this approach, we do not provide solution for cases when a 3PIP instantiates a Trojan infected sub-IP from another IP vendor, as it falls beyond the target scope of our paper. If both vendors are untrusted, the chance of their hidden Trojan logic being same is very rare (as both third party vendors are mutually exclusive of each other). Just performing equivalence check on each IP from the different vendors does not help in Trojan detection, since the Trojan in an IP remains dormant (inactive) in normal condition and gets activated only on external triggering (by an adversary) during real time deployment. Therefore, performing equivalence check on both the IP will not reflect a difference (or Trojan detection).

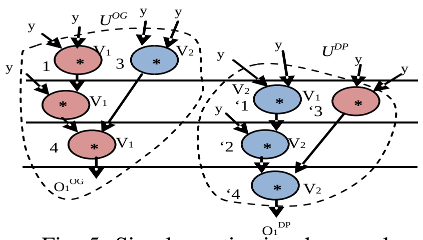


Fig. 5: Simple motivational example

A. Background on PSO Algorithm

PSO is a a metaheuristic search methodology where the particles move through a multidimensional search space. Each particle is attracted towards the position of the current global best x^{gb} and its own best location x^{lb} in history. When a particle finds a location that is better than any previously found locations, then it updates it as the new current best for the particle. The position of i^{th} particle is changed by adding the velocity to the current position as follows:

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{7}$$

wher, x_i and v_i be the position and velocity vector for particle i. While, the velocity is updated with the following rule:

$$v_i(t+1) = \omega v_i(t) + b_1 r_1[x_i^{lb} - x_i(t)] + b_2 r_2[x^{gb} - x_i(t)] \tag{8}$$

where, ω is called the inertia weight, b_1 is the cognitive learning factor, b_2 is the social learning factor, r_1, r_2 are random numbers in the range [0, 1].

B. Proposed Methodology for PSO Driven Exploration of Trojan Secured DMR Datapath

Module library, behavioral description of CDFG and pre-defined user parametric constraints for area and delay, control parameters of PSO (such as inertia weight, acceleration coefficients and swarm size 'p'), maximum iteration count for the CDFG and the pre-processing algorithm for unrolling factors are provided as inputs to the proposed DSE process (as indicated in Fig. 4). The details of the major steps of PSO driven DSE framework are described in section VI-E to VI-H later. However, for the sake of immediate reference, a brief explanation for Fig. 4 is provided below: The PSO-DSE initiates with novel particle encoding process representing a candidate design solution X_i in the design space. The encoding scheme for a candidate design solution comprises of DMR schedule architecture ($\overrightarrow{R_n}$), unrolling factor (U) and distinct vendor allocation procedure type (P_v). Therefore, a particle position which is a candidate design solution in PSO-DSE framework is labeled as X_i :

$$X_i = (\overrightarrow{R_n}, U, P_v) \tag{9}$$

where, $\overrightarrow{R_n}$ indicates the resource array (resource configuration e.g. number of adders, multipliers etc). The reason behind addition of last dimension (vendor allocation procedure type) 'P_v' is discussed in upcoming section. Once the initial encoding phase (initialization of particles) is complete, then the next step is to build a Trojan secured DMR design (discussed in upcoming section) which is subjected to fitness evaluation through the cost computation block. Determination of local and global best particle position is done after this, followed with calculation of particle velocity and new particle position (X_i^+). Determination of new particle position is accomplished by evaluating new value for every dimension (d) for position X_i^+ until, the final dimension 'D' is checked (*note:* in our work, the final dimension 'D' is equal to 'U' as indicated in eqn. 9. This is because the last dimension 'P_v' can only hold Boolean value as discussed in the upcoming paragraph). Interaction with velocity and resource clamping block is performed if required to avoid boundary outreach. Then similarly as discussed before, the next step is to build a Trojan secured DMR design for the new particle position X_i^+ followed by evaluation of its cost computation. If the cost of the new particle position (X_i^+) is found higher than the previous position (X_i), then change in X_i^+ is made by evaluating another new particle position. Else, the local best particle position is updated. This process is repeated for the entire swarm population size (p) and finally the global best position is updated. Mutation process is followed after this to arrive at a further better solution if possible. The aforesaid process continues until the terminating criteria (Z) is reached.

Mutation: Mutation operation is performed on all local best resource configurations with probability P_m=0.25. The algorithm uses two basic operations for mutation: a) Rotation operation, and b) Increment or decrement operation. To perform these, the total population is divided into two groups: a) even group, in which algorithm performs left rotation operation and b) odd group, in which algorithm performs increment or decrement with a random number.

```
Input –value of 'I'(Total no. of loop iteration)
Output –screened set of unrolling factor (U)
1.    Begin
      // Screening of U//
2.    int k = 0
3.    For U=2 to I Do
3.1         IF ((I mod U < (U/2) ) && (U <= I/2)) Then
            //Add Uinto the accepted U list//
3.2         Accepted U [k] = U
3.3         k++
3.4         End IF
3.5   End For
4.    End
```

Fig. 6: Pre-processing of unrolling factor

C. Pre-processing of unrolling factor candidates

Pre-processing (screening) of unrolling factor candidates that do not form part of an optimal solution during exploration is extremely crucial. Some unrolling factors such as the ones which yield large trailer loops are potential sources for greater delay due to multiple sequential loops involved. Further, it is established in [10], [37], that performance is not a monotonically increasing function of unrolling factor value i.e. for large unrolling factor values; therefore, the performance improvement is found marginal. The corresponding algorithm is shown in Fig. 6.

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

In the proposed approach for building Trojan secured schedule, DMR logic with specific vendor allocation rule (discussed in upcoming paragraph) is employed. In order to obtain a Trojan secured DMR schedule, its corresponding DMR schedule first needs to be obtained. In proposed DMR logic, complete duplication is done for all the unrolled operations of the CDFG (based on the unrolling information (U) of the candidate design solution X_i, described in eqn. 9). This indicates that double modular redundancy of the CDFG is done at the behavioral level itself before scheduling. Once operations of the CDFG are unrolled, then both the operations of original unit and duplicate unit are concurrently scheduled using list scheduling algorithm based on the information of the schedule architecture (N(R_1), N(R_2),..N(R_D)) in the candidate design solution (X_i). This enables to extract information of C_{first}^{DMR} and C_{body}^{DMR} to determine the final delay C_T^{DMR} (as discussed in section V-B).

In the proposed approach, as described above DMR logic is employed with some specific vendor allocation rule to design a Trojan secured schedule. 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 (W^{OG}) and v' of duplicate unit (W^{DP}) are assigned to distinct vendor [6]. This enables Trojan security (detection) as for cases where no sub-IP exists, it is highly unlikely that different Trojans in different 3PIP's will produce identical wrong outputs. In this approach, we do not provide solution for cases when a 3PIP instantiates a Trojan infected Sub-IP from another IP vendor, as it falls beyond the target scope of our paper. However it is important to note that distinct vendor assignment rule for Trojan security can be implemented in more than one ways (discussed later), 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 within the DMR schedule (i.e. assignment of each vendor IPs within the system while allocation) controls the final delay and area of the design. This is because the same resource type/IP from two different vendors have different area and delay. Note: Further it is assumed 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' (au = area unit; 1 au = 1 Transistor) 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. Therefore, in the proposed approach components (IPs) with parametric values (i.e., different cost values) are considered during exploration of an optimized Trojan secured schedule. Since the components serve as modules in the HLS library, the final solution of the proposed approach changes if the components from two different vendors have different parameter values.

Let us now discuss the two different ways (denoted by P_v = 1 and P_v = 0) how two distinct vendor allocation can be made inside a DMR scheduling in order to provide Trojan security. The value of 'P_v' as '0'/'1' is characterized as follows:

1) Vendor allocation procedure (Type 1): P_v = 1:

- All operations of a specific unit (original/duplicate), are strictly assigned to same vendor type (i.e., all operations of W^{OG} strictly assigned to vendor 'V_1' and all operations of W^{DP} to vendor 'V_2').
- Similar operations of both W^{OG} and W^{DP} being assigned to two different vendors.

2) Vendor allocation procedure (Type 2): P_v = 0:

- Alternate vendor assignment to operations of a resource type in a CS of a unit. Moreover, in a CS if an opeartion with another resorce type is encountered then independantly alternate vendor assignment is performed. (Example, in Fig. 8, alternate vendor assignments of V_1, V_2, V_1 and V_2 are provided to operations 1, 2, 3 and 10 respectively in a CS. Same assignment is followed for each remaining CS. Moreover in CS#4, separately alternate vendor assignments have been made for each resource types adder and multiplier i.e. 7 & 15 alternatively assigned to V_1 & V_2 separately while, 22 & 21 are also alternately assigned to V_1 & V_2 separately).
- Similar operations of both W^{OG} and W^{DP} being assigned to two different vendors.

As described before in this Section, consider two cases as illustrated in Fig. 7 and 8; which shows list scheduling based CDFG for Diffeq benchmark unrolled thrice with:

- resource constraint of $\overrightarrow{R_n}$ = 2(+), 4(*), 2(-), 1(<); U = 3; iteration count (I) =4 P_v = 0. The corresponding T_E^{DMR} = 176350ns and A_T^{DMR} = 28380 au.
- $\overrightarrow{R_n}$ = 2(+), 4(*), 2(-), 1(<); U =3; iteration count (I)=4; P_v = 1. The corresponding T_E^{DMR} = 173350ns and A_T^{DMR}= 38908 au.

It can be seen that there is a difference in the area and delay values of the two generated scheduling solutions, which have distinct vendor assignment to similar operations for detectability. The DMR schedules with vendor allocation details generated in Fig. 7 and 8 are both hardware Trojan secured (with two vendors required in both cases) with duplication of all the operations of the original unrolled CDFG. For the sake of demonstration let us assume we have a candidate design solution X_i = (2(+), 4(*), 2(-), 2(<), U= 3), P_v for building a Trojan secured DMR schedule. This indicates that the CDFG needs to be unrolled thrice (I_1, I_2 and I_3) and duplication must be done for all the operations of the three iterations ('I_1, 'I_2 and 'I_3) to create an unrolled DMR CDFG. Once unrolled DMR CDFG is created, it (both original unit and duplicate unit) will be concurrently scheduled based on the schedule resource information in X_i: 2 adders, 4 multipliers, 2 subtractors and 2 comparators. However as evident in Fig. 7 and 8, one Trojan secured DMR schedule is better than its counterpart in delay or area based on the value of P_v, regardless of same schedule resources and unrolling information. Therefore, in context of DSE exploration of P_v (indicating allocation procedure of IPs from different vendor type) which can either be '0' or '1' dimension, along with resource array (as shown in eqn. 9) is important.

The cost of obtained Trojan secured DMR schedule is further evaluated. The cost function inudes includes area component due to functional resources, interconnect units (mux and demux), comparator as well as overhead incurred from internal buffering (temporary storage of operation output). The area due to internal buffering involves overhead incurred in a DMR similar operation (from both original duplicate) at different times. The system needs to keep the outputs from both units stored in some internal buffer to compare only when both outputs are ready. This process of evaluating design solutions (particle positions) evolves through the proposed DSE using PSO to generate an optimal hardware Trojan fault secured DMR system that satisfies A_{cons}, T_{cons}, as well as minimizes hybrid cost.

From the above discussion, it is clear that the proposed approach considers three factors:

- Efficient distinct vendor allocation inside DMR scheduling
- Optimized combination of schedule resources
- Screening of unwanted loop unrolling factors to achieve low cost optimal solution to Trojan secured schedule.

All the aforesaid factors were not considered in past approaches so far.

E. Initialization of Particle

In this current proposed approach, a candidate design solution is represented through a particle. The position of a particle in PSO as defined earlier in eqn. 9 can be expanded as follows. The particle position 'X_i' is given as follows:

$$X_i = (N(R_1), N(R_2), ..N(R_D), U, P_v)\,. \tag{10}$$

The particles are uniformly distributed over the design space and are initialized as follows:

$$X_1 = \quad (min(R_1), min(R_2), ..min(R_D), min(U), 0) \tag{11}$$

$$X_2 = \quad (max(R_1), max(R_2), ..max(R_D), max(U), 1) \tag{12}$$

$$X_3 = \left(\frac{min(R_1)+max(R_1)}{2}, \ldots \frac{min(R_D)+max(R_D)}{2}, \frac{min(U)+max(U)}{2}, 0\right). \tag{13}$$

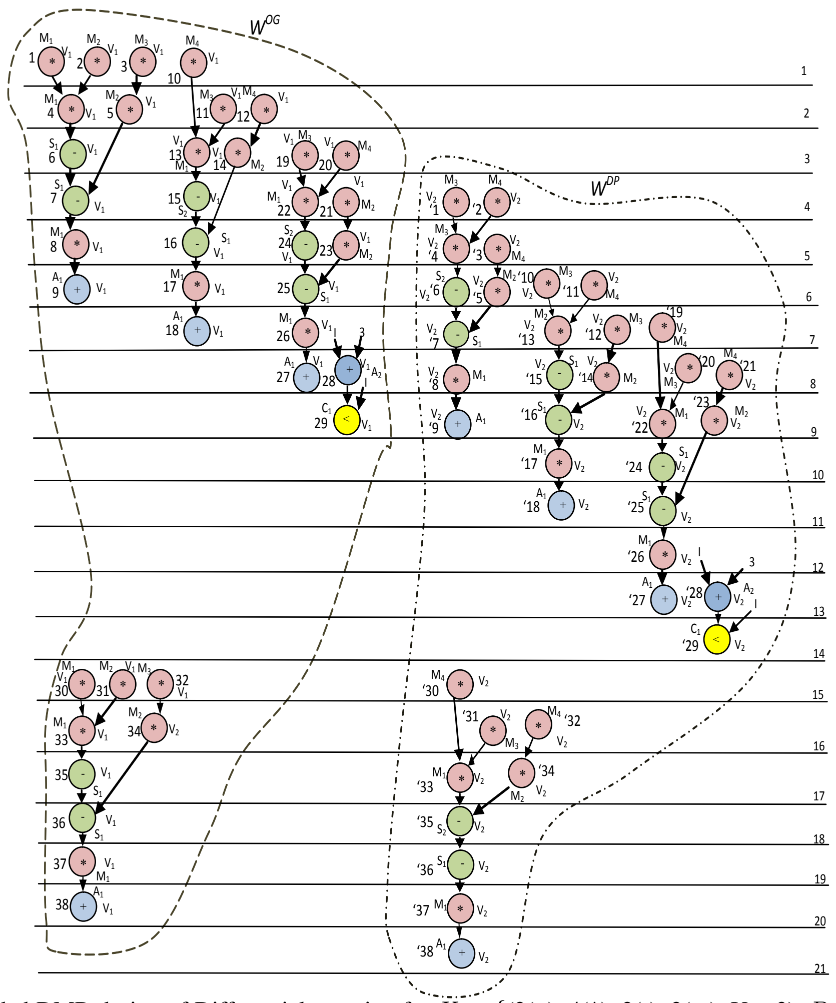


Fig. 7: Scheduled and binded DMR design of Differential equation for X_i = {(2(+), 4(*), 2(-), 2(<), U = 3), P_v} with P_v = 1; for achieving vendor distinctness for Trojan security; T_E^{DMR} = 173350ns and Area = 38908au.

However, rest of the particles ($X_4 ... X_n$) are initialized by the following:

$$X_n = \Big(\frac{min(R_1)+max(R_1)}{2} \pm \alpha, \frac{min(R_2)+max(R_2)}{2} \pm \alpha, \\ ... \frac{min(R_d)+max(R_d)}{2} \pm \alpha, \frac{min(U)+max(U)}{2} \pm \alpha, \\ random(P_v)\Big). \quad (14)$$

where, α is a random integer between *min* and *max* of particular resource type or unrolling factor.

F. Particle Movement using Velocity

PSO-DSE [39], each dimension (d) of a particle position X_i (except the last dimension) is updated using the following function [37]:

$$R_{d_i}^{+} = R_{d_i} + V_{d_i}^{+}. \quad (15)$$

The variable $V_{d_i}^{+}$ is updated by eqn. 16 as follows:

$$V_{d_i}^{+} = \omega V_{d_i} + b_1 r_1 [R_{d_{lb_i}} - R_{d_i}] + b_2 r_2 [R_{d_{gb}} - R_{d_i}] \quad (16)$$

The local best (X_{lb_i}) and global best (X_{gb}) particle positions are updated using eqn. 17 and 18:

$$X_{lb_i} = R_{1_{lb_i}}, ... R_{D-1_{lb_i}}, U \quad (17)$$

$$X_{gb} = R_{1_{gb}}, ... R_{D-1_{gb}}, U \quad (18)$$

The variables have been explained in Table I. A DMR schedule ($SDFG^{DMR}$) (with distinct vendor assignment rule to detect the hardware Trojan), is generated corresponding to a particle positions/configurations.

G. Velocity clamping

The velocity clamping adopted from [37] is performed, when a particle's exploration drift ($V_{d_i}^{+}$) crosses the $\pm V_{d_i}^{max}$ as follows:

$$V_{d_i}^{+} = \begin{cases} +V_{d_i}^{max} & if \quad V_{d_i}^{+} > +V_{d_i}^{max} \\ -V_{d_i}^{max} & if \quad V_{d_i}^{+} < -V_{d_i}^{max} \\ V_{d_i}^{+} & else \end{cases} \quad (19)$$

In the above expression, the value of $\pm V_{d_i}^{max}$ is the following:

$$V_{d_i}^{+} = \left(\pm \frac{max(N(R_d)) - min(N(R_d))}{2}\right). \quad (20)$$

H. Terminating condition

The proposed methodology terminates when a) the maximum number of iterations exceeds 100 (S_1), or when no improvement is visible in X_{gb} over 'δ' number of iterations. (δ =10) (S_2).

VII. Experimental Results

This section shows the results of proposed methodology discussed in following sub-sections: (a) experimental setup and benchmark, (b) analysis of results, (c) comparison with related prior research and (d) detection ability of design solutions generated during DSE.

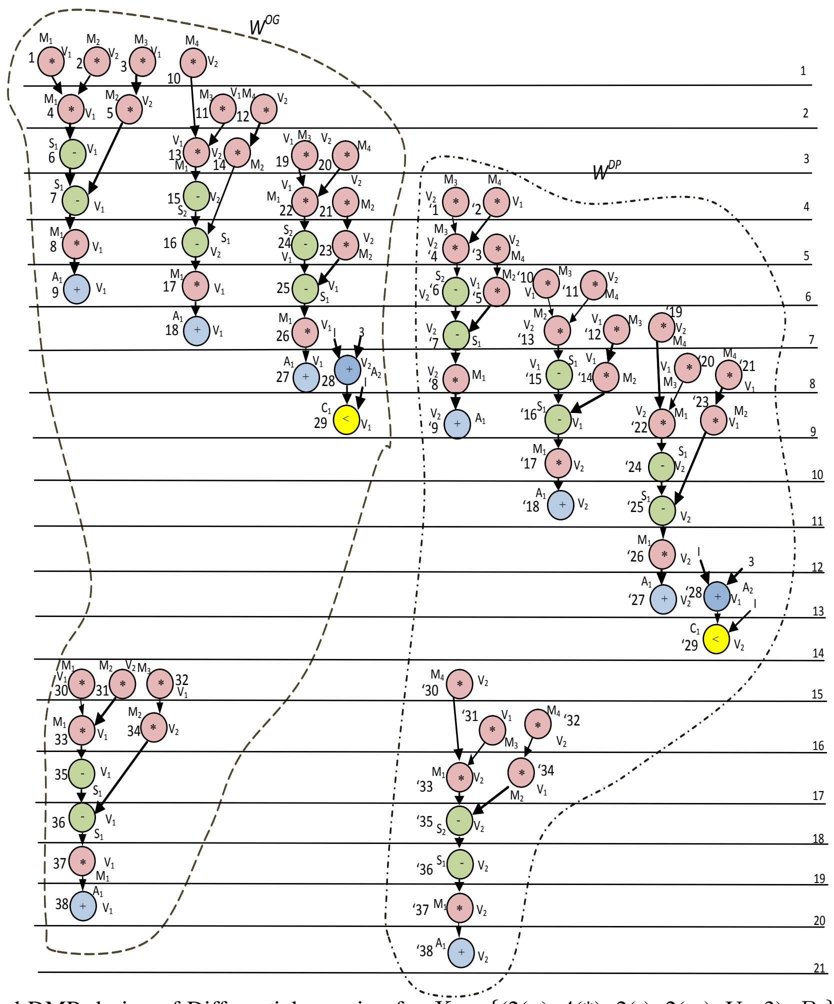


Fig. 8: Scheduled and binded DMR design of Differential equation for X_i = {(2(+), 4(*), 2(-), 2(<), U= 3), P_v} with P_v = 0; for achieving vendor distinctness for Trojan security; T_E^{DMR} = 176350ns and Area = 28380au.

A. Experimental Setup and Benchmarks

The proposed approach and [6], have been implemented in java. The proposed flow when implemented run on Intel Core-i5-3210M CPU with 3MB L3 cache memory, 4GB DDR3 primary memory and processor frequency of 2.5 GHz. During experimentation, 15 runs were executed for proposed PSO-DSE with equal weightage to both user objectives of area and delay (W_1= W_2 =0.5). During experiments, it was found that the proposed approach is scalable and is able to handle large size problems (for instance, mutiple CDFG applications with nodes greater than 200 have been tested, which also yielded final optimal solution within acceptable exploration runtime). Further, during experiments, following optimal settings from [40] for PSO framework were fixed: ω (inertia weight) = linearly decreasing between 0.9 to 0.1; b_1 and b_2 (acceleration coefficient) = 2; r_1 and r_2 (random numbers) = 1; stopping criterion = S_1 or S_2; p = 3 or 5 or 7.

B. Analysis of Results

Table II shows the effect of swarm size 'p' on the exploration time of the proposed methodology. As seen from Table II (for all benchmarks), the exploration time of the proposed approach to find the non- dominated solutions increases with the increase in swarm size, however with no improvement in the quality of the solution found. This is due to the

TABLE II: Comparison of Exploration Time with respect to swarm size 'p' for proposed approach

Benchmark	**Exploration Time (ms) for Swarm Size, p**		
	p=3	p=5	p=7
Differential Equation	370	550	678
FIR	446	769	978
FFT	1797	2680	3847
Test Case	571	715	1345
Mesa Interpolate	32522	57550	91818
ARF	594	828	1844
IDCT	8679	14719	20680
WDF	2453	5175	7344
GLRT	3045	6415	7933
DHMC	21836	43218	62345

fact that, total number of positions evaluated in a run increases with the increase in 'p'. Additionally, for all the benchmarks viz. DFGs, single and nested loop applications tested, the proposed approach generates, solutions that lie within the user provided are-delay constraints and minimizes the hybrid cost in eqn. 6, as shown in Table III. For example, for FIR benchmark, the proposed approach generates the final solution (X_i) : (N(R_{adder}), N($R_{multiplier}$), N($R_{comparator}$), U, P_v) = (2(+), 6(*), 1(<), U= 3, P_v = 0) with A_T^{DMR} = 23936 au and

TABLE III: Results of Proposed Approach
Note: 1 a.u. = 1 Transistor; us = microseconds

Benchmark	T_{cons} (us)	T_E^{DMR} (us)	A_{cons} (au)	A_T^{DMR}	Cost
Differential Equation	182.2	120.81	43301	29136	-0.21
FIR	40.00	23.62	27000	23936	-0.16
FFT	200.0	163.18	41000	26632	-0.21
Test Case	130.0	99.00	29000	23070	-0.15
Mesa Interpolate	237.6	99.35	117972	60254	-0.29
ARF	132.8	89.89	23747	16198	-0.23
IDCT	119.1	77.08	42858	31582	-0.22
WDF	115.60	111.69	22154	18356	-0.08
GLRT	170.00	109.80	50000	24962	-0.23
DHMC	420.00	262.33	35000	33524	-0.14

TABLE IV: Results of Approach [6]
Note: 1 a.u. = 1 Transistor; us = microseconds

Benchmark	T_E^{DMR} (us)	T_E^{DMR} (us)	A_{cons} (au)	A_T^{DMR} (au)	Cost
Differential Equation	182.2	131.27	43301	29640	-0.19
FIR	40.00	34.08	27000	24692	-0.07
FFT	200.0	179.24	41000	33974	-0.10
Test Case	130.0	99.27	29000	23826	-0.14
Mesa Interpolate	237.6	74.33	117972	75050	-0.29
ARF	132.8	121.54	23747	21398	-0.06
IDCT	119.1	98.89	42858	38746	-0.09
WDF	115.60	111.69	22154	24422	-0.01
GLRT	170.0	78.08	50000	37580	-0.23
DHMC	420.00	286.35	35000	52824	0.05

T_E^{DMR}= 23.62 us, which completely satisfies the area-delay constraints (A_{cons} = 27000 au and T_{cons} = 40us). *(NOTE: A_{cons} and T_{cons} could be any value between the minimum (A_{min}, T_{min}) and maximum value (A_{max}, T_{max})).* Similar behavior is observed for the other benchmarks. Additionally, the result for approach [6] is shown in Table IV, which shows that the execution time (T_E^{DMR}) for [6] is higher than proposed T_E^{DMR}. Further, the process of screening the unfit unrolling factor values through algorithm in Fig. 6 is described in Table V, VI and VII (for sake of brevity additional tables have not been added). The first column indicates the unrolling factor values, second represents the number of sequential loops, third indicates the number of pipelined loops and the fourth column indicates accept/reject status. The results produces accept/reject of unrolling factor values by screening unroll values that results in large sequential loops as well as ignoring very large unroll values that provide only marginal performance improvement (as performance enhancement is not a monotonically increasing function of unroll factor values).

C. Comparison with Related Prior Research

This sub-section reports the comparative results with a state of the art approach [6] with respect to the final solution.

The cost function, for both [6] and proposed approach, also considers the area of single comparator/error detection block responsible to runtime Trojan detection at the final output. Table. VIII shows the comparative results of the proposed approach with [6], in terms of cost, hardware area, delay, respective overheads and final architectural solution found. Since both [6] and proposed approach uses the concept of diverse 3PIP vendors, hence both approaches provide equally robust security. As evident, the proposed approach generates

TABLE V: Pre-processing of Unrolling factors for Differential Equation

I = 16

U	Sequential loop (I mod U)	Pipelined loop (I-I mod U)	Accepted(I)
2	0	16	1
3	1	15	1
4	0	16	1
5	1	15	1
6	4	12	0
7	2	14	1
8	0	16	1
9	7	9	0
10	6	10	0
11	5	11	0
12	4	12	0
13	3	13	0
14	2	14	0
15	1	15	0
16	0	16	1

TABLE VI: Pre-processing of unrolling factors for FIR

I = 18

U	Sequential loop (I mod U)	Pipelined loop (I-I mod U)	Accepted(I)
2	0	18	1
3	0	18	1
4	2	16	1
5	3	15	0
6	0	18	1
7	4	14	0
8	2	16	1
9	0	18	1
10	8	10	0
11	7	11	0
12	6	12	0
13	5	13	0
14	4	14	0
15	3	15	0
16	2	16	0
17	1	17	0
18	0	18	0

lower cost solutions with no area and delay overheads (in most cases) in comparison to [6]. This is because of the following: (a) The proposed approach comprises of a supplementary option in encoding for exploration of an optimal allocation procedure of vendors P_v, which bears an impact on the final optimization quality; (b) The proposed approach comprises of another supplementary option in encoding for exploration of an optimal unrolling factor; (c) The proposed approach provides exploration of schedule resources based on user constraints, which also bears an effect on the final design quality. Therefore, calculations show that, the proposed approach provides an average reduction of 11.4 % in cost of the final Trojan secured schedule compared to [6]. Since there is no feature for exploring an optimal unrolling factor and distinct vendor allocation in [6], it uses the maximum unroll factor value as well as same vendor allocation to each unit rule (P_v = 1) during DMR design thereby resulting in higher cost. Therefore, distinct vendor allocation type, P_v = 0, always yields lower cost final solution than P_v = 1.

D. Detection ability of design solutions generated during DSE

Type of Trojan inserted: We insert Trojans in components of HLS library that can only change the output computational value (and has no other effect). Therefore, this paper targets only this specific class of Trojan during detection analysis. Few examples of this specific class of Trojan inserted in our approach and its payload are shown in Fig. 3 .

TABLE VII: Pre-processing of unrolling factors for FFT

I = 24			
U	**Sequential loop (I mod U)**	**Pipelined loop (I-I mod U)**	**Accepted(I)**
2	0	24	1
3	0	24	1
4	0	24	1
5	4	20	0
6	0	24	1
7	3	21	1
8	0	24	1
9	6	18	0
10	4	20	1
11	2	22	1
12	0	24	1
13	11	13	0
14	10	14	1
15	9	15	0
16	8	16	0
17	7	17	0
18	6	18	0
19	5	19	0
20	4	20	0
21	3	21	0
22	2	22	0
23	1	23	0
24	0	24	0

The proposed DSE framework generates solution for Trojan secured DMR schedule. A DMR schedule is generated for every explored solution (before fitness evaluation) which needs validation in its ability to detect hardware 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 an adversary in the third party design house. Typically, the HLS company provides the RTL files of the modules/IPs of the library that may contain malicious/Trojan logic, which might have been imported from third party vendors. The proposed approach deals with hardware Trojan in the modules/IP that changes its computational value. However the key is that Trojan only becomes visible at runtime (after triggering by an adversary), thereby remaining completely ineffective before that, thus difficult to be detected during RTL simulation/other lower level tests. For checking the detection ability of the solution generated, the presence of hardware Trojan in module of HLS library was emulated by inserting malicious logic in the module (such as adder/ subtractor etc.) of the HLS library. Hardware Trojan were implemented in the modules of the HLS library as maliciously altered RTL codes. Some examples of the schematic equivalent of the malicious alterations made in the RTL codes of the HLS module library are shown in Fig. 3. Test vectors generated through 8^{th} order polynomial Linear-Feedback Shift Register (LFSR) as ATPG is fed into the DMR schedule for comprehensive coverage of Trojan detection in the final explored solution. Results indicate that in the explored solution whenever a Trojan was inserted (in any module provided by a vendor); difference in the computational value in the output was noticed from original and duplicate unit, indicating Trojan detection. 100% Trojan detection has been possible when handling such Trojans, because these Trojans when activated produce no other effect, but only change in computational value as payload. Hence, through the proposed DMR allocation rules for both P_v = 0 and P_v = 1, a complete coverage of Trojan detection was obtained. Therefore, it can be said that, it is safe to allow interleaving of IPs (P_v = 0) resulting from different vendors. The time taken for the final Trojan secured schedule depends on the initial population selected during exploration. For example, time taken for Trojan security is $\sim$ 5.9 sec, $\sim$ 10.3 sec and $\sim$ 16.0 sec for p = 3, 5 and 7 respectively. For successful detection at least one comparison point is needed, which is at the primary output of DMR scheduling. However, additional checkpoints can be inserted at the output of each unrolled loop body in DMR scheduling. This indicates that the additional checkpoints are equal to the unrolling factor value. In the proposed approach, security can be provided against any Trojan that has ability to affect functional output. This is because insertion of Trojan in a 3PIP by a rogue induces a different output in one of the units than the other due to usage of a distinct vendor in original and duplicate units. This results in comprehensive detection in all cases. However Trojan which affects output in terms of only disabling components (with no change in output magnitude) cannot be detected in our approach.

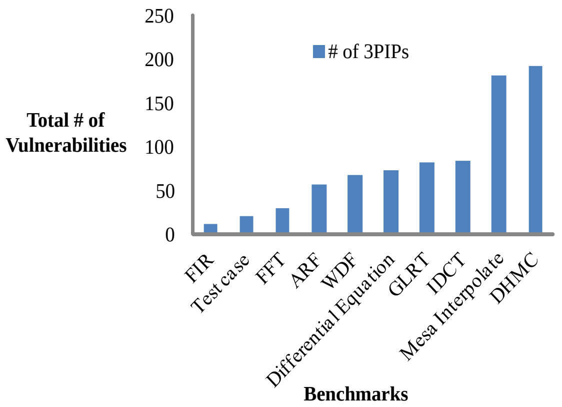


Fig. 9: Number of potential untrustworthy 3PIPs as vulnerabilities detected (secured) by proposed approach

Security Analysis: We further provide security analysis by presenting how robust our approach is. This is a direct indicative of the number of vulnerabilities detected by proposed approach. Each potentially untrustworthy 3PIPs used in the design is considered as a vulnerability. Further, the number of 3PIPs used is directly proportional to the number of operations in the DMR design. In the proposed approach we detect each such potential vulnerability by applying the concept of distinct vendor to both original and duplicate operations in DMR design. Thus the extent of security is the capability of proposed approach to detect potential vulnerabilities. Fig.9 shows the number of potential vulnerabilities detectable by proposed approach for each benchmark (comprising of a mix of DFGs, single loop CDFGs and nested loop CDFGs). As seen in Fig. 9, the number of vulnerabilities is largest for DHMC nested loop CDFG, which are all detected by proposed approach.

VIII. Conclusions and Future Research

This paper presented a novel work on simultaneous exploration of an optimized Trojan secured schedule and optimal loop unrolling factor for CDFGs. More explicitly, the paper contributed in the following novel aspects: (a) a model for execution delay determination of a DMR system for Trojan secured CDFG (b) particle encoding scheme that concurrently

TABLE VIII: Comparison of Proposed DSE Approach with [6]

Benchmark	Final architecture solution for Trojan Secured schedule (proposed)	Final architecture solution for Trojan Secured schedule [6]	Cost of final solution (proposed)	Cost of final solution [6]	Area overhead of [6] compared to proposed (a.u.)	Delay overhead of [6] compared to proposed (us)
Differential Equation	3(+), 3(-), 5(*), 2(<), U = 4, P_v = 0	2(+), 2(-), 5(*), 2(<), U = 4, P_v = 1	-0.21	-0.19	504	10.46
FIR	2(+), 6(*), 2(<), U = 3, P_v = 0	3(+), 5(*), 2(<), U = 6, P_v = 1	-0.16	-0.07	756	10.46
FFT	4(+), 2(-), 4(*), 2(<), U = 1, P_v = 0	5(+), 2(-), 5(*), 2(<), U = 4, P_v = 1	-0.21	-0.10	7342	16.06
Test Case	4(+), 3(*), 2(<), U = 2, P_v = 0	3(+), 3(*), 2(<), U = 6,P_v = 1	-0.15	-0.14	756	0.27
Mesa Interpolate	3(+), 16(*), P_v = 0	10(+), 17(*), P_v = 1	-0.29	-0.29	14796	-
ARF	2(+), 4(*), P_v = 0	5(+), 3(*), P_v = 1	-0.23	-0.06	5200	31.65
IDCT	6(+), 4(*), P_v = 0	5(+), 3(*), P_v = 1	-0.22	-0.09	7164	21.81
WDF	3(+), 2(*), P_v = 0	3(+), 3(*), P_v = 1	-0.08	-0.01	6066	-
GLRT	4(+), 6(*), P_v = 0	4(+), 10(*), P_v = 1	-0.23	-0.23	12618	-
DHMC	4(+), 6(*), 2(<), P_v = 0	3(+), 5(*), 2(<), P_v = 1	-0.14	0.05	19300	20.02

explores schedule configuration, unrolling factor and vendor allocation procedure (c) methodology for area - execution delay trade-off using PSO during optimization of secured schedule.

Our future research directions includes nanoelectronic technology based trusted HLS as the hardware eventually implemented using such a technology [10]. For example, we intend to work on architecture-level synthesis based obfuscation technique, IP trust, process variation awareness, and fault tolerance.

REFERENCES

[1] X. Zhang and M. Tehranipoor, "Case study: Detecting hardware Trojans in third-party digital IP cores," in *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, June 2011, pp. 67–70.

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

[3] R. Karri, J. Rajendran, K. Rosenfeld, and M. Tehranipoor, "Trustworthy Hardware: Identifying and Classifying Hardware Trojans," *Computer*, vol. 43, no. 10, pp. 39–46, Oct 2010.

[4] S. Bhunia, M. Abramovici, D. Agrawal, P. Bradley, M. Hsiao, J. Plusquellic, and M. Tehranipoor, "Protection Against Hardware Trojan Attacks: Towards a Comprehensive Solution," *IEEE Design Test*, vol. 30, no. 3, pp. 6–17, June 2013.

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

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

[7] X. Cui, K. Ma, L. Shi, and K. Wu, "High-Level Synthesis for Run-Time Hardware Trojan Detection and Recovery," in *Proceedings of 51st Annual Design Automation Conference*. USA: ACM, 2014, pp. 157:1–157:6.

[8] D. McIntyre, F. Wolff, C. Papachristou, S. Bhunia, and D. Weyer, "Dynamic Evaluation of Hardware Trust," in *Proceedings of 2009 IEEE International Workshop on HOST*. USA: IEEE Computer Society, 2009, pp. 108–111.

[9] 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.

[10] S. P. Mohanty, M. Gomathisankaran, and E. Kougianos, "Variability-aware Architecture Level Optimization Techniques for Robust Nanoscale Chip Design," *Comput. Electr. Eng.*, vol. 40, no. 1, pp. 168–193, Jan. 2014.

[11] X. Wang, M. Tehranipoor, and J. Plusquellic, "Detecting malicious inclusions in secure hardware: Challenges and solutions," in *IEEE International Workshop on HOST*, June 2008, pp. 15–19.

[12] M. Tehranipoor and F. Koushanfar, "A Survey of Hardware Trojan Taxonomy and Detection," *IEEE Design Test of Computers*, vol. 27, no. 1, pp. 10–25, Jan 2010.

[13] A. Sengupta and S. Bhadauria, "Automated exploration of datapath in high level synthesis using temperature dependent bacterial foraging optimization algorithm," in *IEEE 27th Canadian Conference on Electrical and Computer Engineering (CCECE)*, May 2014, pp. 1–5.

[14] C. Haubelt, T. Schlichter, J. Keinert, and M. Meredith, "SystemCoDesigner: Automatic design space exploration and rapid prototyping from behavioral models," in *45th ACM/IEEE Design Automation Conference (DAC)*, June 2008, pp. 580–585.

[15] A. Sengupta and R. Sedaghat, "Integrated scheduling, allocation and binding in High Level Synthesis using multi structure genetic algorithm based design space exploration," in *12th International Symposium on Quality Electronic Design (ISQED)*, March 2011, pp. 1–9.

[16] 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, 2006.

[17] 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 *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2008, pp. 417–422.

[18] Z. Zeng, R. Sedaghat, and A. Sengupta, "A framework for fast design space exploration using fuzzy search for VLSI computing Architectures," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2010, pp. 3176–3179.

[19] C. Mandal, P. Chakrabarti, and S. Ghose, "GABIND: a GA approach to allocation and binding for the high-level synthesis of data paths," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 8, no. 6, pp. 747–750, Dec 2000.

[20] 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 Design*, vol. 2012, pp. 2:2–2:2, Jan. 2012.

[21] 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, no. 3, pp. 603–628, 2002.

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

[23] S. Das, A. Biswas, S. Dasgupta, and A. Abraham, "Bacterial Foraging Optimization Algorithm: Theoretical Foundations, Analysis, and Applications," in *Foundations of Computational Intelligence Volume 3*. Springer Berlin Heidelberg, 2009, vol. 203, pp. 23–55.

[24] A. Hu, "Formal hardware verification with BDDs: an introduction," in *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, vol. 2, Aug 1997, pp. 677–682.

[25] M. Bushnell and V. Agrawal, *Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits*. Springer Publishing Company, Incorporated, 2013.

[26] H. Salmani, M. Tehranipoor, and J. Plusquellic, "New design strategy

for improving hardware Trojan detection and reducing Trojan activation time," in *IEEE International Workshop on HOST*, July 2009, pp. 66–73.

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

[28] 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 *IEEE International Symposium on HOST*, June 2010, pp. 13–18.

[29] Y. Jin and Y. Makris, "Hardware Trojan detection using path delay fingerprint," in *IEEE International Workshop on HOST*, June 2008, pp. 51–57.

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

[31] X. Cui, K. Ma, L. Shi, and K. Wu, "High-Level Synthesis for Run-Time Hardware Trojan Detection and Recovery," in *51st Annual Design Automation Conference (DAC)*. USA: ACM, 2014, pp. 157:1–157:6.

[32] 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 *25th Edition on Great Lakes Symposium on VLSI (GLSVLSI)*. USA: ACM, 2015, pp. 167–172.

[33] A. Sengupta and R. Sedaghat, "Swarm intelligence driven design space exploration of optimal k-cycle transient fault secured datapath during high level synthesis based on user powerdelay budget," *Microelectronics Reliability*, vol. 55, no. 6, pp. 990 – 1004, 2015.

[34] 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.

[35] J. Crop, S. Fairbanks, R. Pawlowski, and P. Chiang, "150mV subthreshold 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.

[36] 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.

[37] A. Sengupta and V. Mishra, "Swarm Intelligence Driven Simultaneous Adaptive Exploration of Datapath and Loop Unrolling Factor during Area-Performance Tradeoff," in *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, July 2014, pp. 106–111.

[38] A. Sengupta and S. Bhadauria, "Exploration of Multi-objective Tradeoff during High Level Synthesis Using Bacterial Chemotaxis and Dispersal," *Procedia Computer Science*, vol. 35, pp. 63 – 72, 2014.

[39] I. C. Trelea, "The particle swarm optimization algorithm: convergence analysis and parameter selection," *Information Processing Letters*, vol. 85, no. 6, pp. 317 – 325, 2003.

[40] A. Sengupta and V. K. Mishra, "Automated exploration of datapath and unrolling factor during powerperformance tradeoff in architectural synthesis using multi-dimensional PSO algorithm," *Expert Systems with Applications*, vol. 41, no. 10, pp. 4691 – 4703, 2014.

Anirban Sengupta is currently an Assistant Professor in Discipline of Computer Science and Engineering at Indian Institute of Technology (I.I.T) Indore, where he directs the research lab on Behavioural Synthesis of Digital IP core . He holds a Ph.D. and M.A.Sc. in Electrical and Computer Engineering from Ryerson University, Toronto (Canada) and is a registered Professional Engineer of Ontario (P.Eng.). He also holds a B.Tech degree from West Bengal University of Technology, India. In the past, he was also affiliated with Indian Institute of Science (IISc) Bangalore as a visiting research scholar. His research interest includes Optimization during Design Space Exploration for Hardware Accelerators, High Level Synthesis, Fault Secured High Level Synthesis, Trojan Security Aware HLS, Hardware Trust in High Level Synthesis, IP core Protection during HLS, Evolutionary Computing during HLS as well as Physical Design using CAD. His research/sponsored projects are funded by Department of Science and Technology (Science and Engineering Research Board), Govt. of India as well as supported by Intel Corporation and Department of Electronics and IT (DEITY), Govt. of India. He has more than 100 Publications and Patents (major contributions in IEEE Transactions, IEEE Journals/Magazines, Elsevier Journals, IET Journals, US Patents, Canadian Patents etc.) in reputed venues. He has been awarded by IEEE for my research contributions to IEEE Access Journal as Associate Editor and Author (The award certificate is attached). He is owner 11 Patents (granted/published/pending). In the past, his Patents generated funding from Ontario Center of Excellence (OCE), Canada. He had performed industry interactive research extensively for more than 2 years with Calypto, Bluespec, BEECube, Huawei Canada during development of his Ryerson Design Space Exploration Tool arising from his Patent. He currently serves in Editorial positions of 7 IEEE and IET Journals as Associate Editor, Columnist and Guest Editor. He is currently serving as Associate Editor of IET Journal on Computer and Digital Techniques, Associate Editor and Columnist of IEEE Consumer Electronics(M-CE), Associate Editor of IEEE VLSI Circuits and Systems Letter (VCAL) and Associate Editor of IEEE Access Journal. He further serves as Guest Editor of IEEE Transactions on Consumer Electronics, IEEE Transactions on VLSI Systems and IEEE Access Journals. He is a regular reviewer of IET Journal on Computers and Digital Techniques, IEEE Transactions on VLSI, Elsevier Journal on Swarm and Evolutionary Computation, Elsevier Journal on Applied Soft Computing and Elsevier Journal on Expert Systems. He regularly serves as a member of the Technical Program Committee of IEEE-CS ISVLSI, ACM GLVLSI, IEEE CCECE and ICIT. He has supervised 4 Ph.D. candidates (2 completed and 2 pursuing)

Saumya Bhadauria holds a Ph.D. in Computer Science and Engineering from Indian Institute of Technology (I.I.T) Indore. Her area of research is data path optimization using bio inspired algorithms, fault security during HLS, Hardware security during HLS, IP core protection during HLS etc. She has number of publications in IEEE, Elsevier and IET. She received her M. Tech. Degree in Information Security from Indian Institute of information Technology and Management, M.P., India.

Saraju P. Mohanty (SM'08) is a Professor at the Department of Computer Science and Engineering (CSE), University of North Texas (UNT), where he directs the NanoSystem Design Laboratory (NSDL). He obtained a Ph.D. in Computer Engineering from the University of South Florida (USF) in 2003, a Masters degree in Systems Science and Automation (SSA) from the Indian Institute of Science (IISc), Bangalore, India in 1999, and a Bachelor's degree (Honors) in Electrical Engineering from Orissa University of Agriculture and Technology (OUAT), Bhubaneswar, India in 1995. Prof. Mohanty's research is in "Low-Power High-Performance Secure Electronic Systems". Prof. Mohanty's research has been funded by National Science Foundation (NSF), Semiconductor Research Corporation (SRC), and Air Force. Dr. Mohanty is an inventor of 4 US patents. Prof. Mohanty is an author of 200 peer-reviewed journal and conference articles, and 3 books. The publications are well-received by the world-wide peers with a total of 2,400 citations leading to an h-index of 25 and i10-index of 63 (from Google Scholar). His latest book titled "Nanoelectronic Mixed-Signal System Design" is published by McGraw-Hill in 2015 is a best seller. This book received 2016 PROSE (Professional & Scholarly Excellence) Award for best Textbook in Physical Sciences & Mathematics from the Association of American Publishers (AAP). Prof. Mohanty has been serving on the editorial board of several peer-reviewed international journals or transactions. He currently serves on the editorial board of 6 peer-reviewed international journals, including IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), ACM Journal on Emerging Technologies in Computing Systems (JETC), and IET Circuits, Devices & Systems Journal (CDS). He is currently Editor in Chief (Designate) of the IEEE Consumer Electronics Magazine. He serves as a founding Editor in Chief (EiC) of the VLSI Circuits and Systems Letter (VCAL). He has been serving as a guest editor for many prestigious journals including ACM Journal on Emerging Technologies in Computing Systems (JETC) and IEEE Transactions on Emerging Topics in Computing (TETC). Prof. Mohanty currently serves as the Chair of Technical Committee on Very Large Scale Integration (TCVLSI), IEEE Computer Society (IEEE-CS) to oversee a dozen of IEEE conferences. He serves on the organizing and program committees of several international conferences. He serves on the steering committee of the IEEE-CS Symposium on VLSI (ISVLSI). He is the founder conference chair for the IEEE International Symposium on Nanoelectronic and Information Systems (IEEE-iNIS). He was a conference chair for the IEEE-CS Symposium on VLSI (ISVLSI) 2012 and 2014. Prof. Mohanty is a senior member of IEEE and ACM. Prof. Mohanty has supervised 7 Ph.D. dissertations and 24 M.S. theses; seven of these advisees have received outstanding student awards at UNT. He has received Honors Day recognition as an inspirational faculty at the UNT for multiple years. He has also received UNT Provosts Thank a Teacher recognition for multiple years. More about his biography, research, education, and outreach activities can be obtained from his website: http://www.smohanty.org.