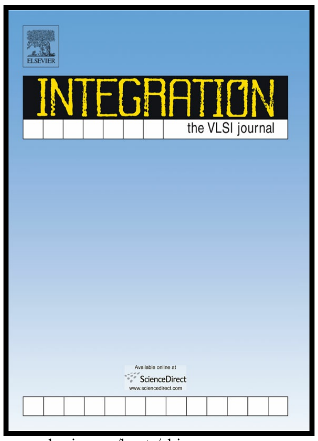
# Optimized Hardware Design for Trojan Security at Behavioral Level for Loop Based Applications

Dipanjan Roy
Institute for Development & Research in Banking Technology
**27** PUBLICATIONS **173** CITATIONS

Saumya Bhadauria
Indian Institute of Technology Indore
**33** PUBLICATIONS **243** CITATIONS


www.elsevier.com/locate/vlsi

# Low Cost Optimized Trojan Secured Schedule at Behavioral Level for Single & Nested Loop Control Data Flow Graphs *(Invited Paper)*

Anirban Sengupta, Dipanjan Roy, Saumya Bhadauria

PII: S0167-9260(16)30069-4
DOI: http://dx.doi.org/10.1016/j.vlsi.2016.09.007
Reference: VLSI1244

To appear in: *Integration, the VLSI Journal*

Received date: 14 March 2016
Revised date: 19 September 2016
Accepted date: 22 September 2016

# Low Cost Optimized Trojan Secured Schedule at Behavioral Level for Single & Nested Loop Control Data Flow Graphs
*(Invited Paper)*

Anirban Sengupta[a], Dipanjan Roy[a], Saumya Bhadauria[b]
[a]Computer Science & Engineering, Indian Institute of Technology, Indore, India
[b]Computer Science & Engineering, Indian Institute of Information Technology, Kota, India
asengupt@iiti.ac.in

***Abstract***
**Internet of Things (IoT) powered by high level synthesis (HLS) provides huge opportunity of progress in the area of hardware design. However, the present era of hardware design involves globalization which poses security threat to the design integrators that rely on third party intellectual property (IP) cores for increasing design productivity at reduced design time. This paper presents for the first time in the literature, a low cost optimized hardware Trojan secured HLS approach for hardware (or application specific core) designs that is based on single or nested loop control data flow graphs (CDFG) applications. The paper presents multiple novel vendor allocation schemes (each with its own attribute of delay and area) as security constraint that yield Trojan secured schedules at behavioral level. Demonstration of the proposed approach on a nested loop case study asserts our proposed theory. Results on standard benchmarks indicate significant reduction in final solution cost compared to a similar approach.**

***Keywords—IP; HLS; low cost; loop, CDFGs; Trojan; Security***

## I. INTRODUCTION

IoT is a system of interconnected electronic/computing devices such as application specific processor, microcontrollers, macro IPs, sensors etc. Many of these applications are algorithms which are data/control intensive driven with requirements of low latency, low power, high throughput etc. which mandates hardware realization. IoT thus presents opportunity for growth in the area of hardware design enabled by high level synthesis. However in high level synthesis, design library comprises of lot of third party IP (3PIP) cores as modules. This is because in order to develop complex designs, third party IPs are used to increase the design productivity. But there are serious security concerns for design integrators due to involvement of third party IPs owing to globalization involved [1,2,3]. In a 3PIP design, an adversary (an untrustworthy vendor) can deliberately infuse a Trojan logic resulting in erroneous computation of a digital design. Such Trojan logic may corrupt the IP by either external activation (such as antennas or sensors) or internal activation (such as internal states of finite state machine (FSM) or counters), resulting in malfunctioning. Therefore, to have a Trojan secured design it should be ensured during HLS [3-7] that any possible infection of 3PIP is detectable. Thus a paradigm shift in 'high level synthesis of IoT' is needed to incorporate trust/security objective besides power, area, energy etc. The designs generated should be equipped with low cost detection capability to fight against Trojan threats that targets alteration of computational value [3, 8, 9]. Typically, the hardware design with Trojan detection (security) capability generated through HLS methodology contains additional register transfer level (RTL) blocks as detection logic resulting from imposed security constraints. Trojan security constraints in the form of dual modular redundancy include duplication of functional modules which incur additional RTL logic, resulting in both delay and area overhead [10, 11]. As providing Trojan detection capability to HLS designs will in most cases require additional digital circuitry hence exploring a low cost solution is crucial besides providing Trojan security.

The novel contributions of this paper are as follows: (*Note: seminal version of this work is presented in iNIS 2015* [11])
1) Presents a novel low cost security aware HLS for optimized Trojan secured schedule within user constraints.
2) Presents a novel encoding to handle single and nested loop CDFGs using Particle Swarm Optimization (PSO) that explores an optimal vendor allocation procedure besides schedule configuration.
3) Provides lower cost Trojan security aware solutions that are of better quality than solutions obtained through existing approach [10].

**Threat Models**

By default a Trojan in a circuit is ineffective as its trigger is not enabled, which indicates there is no payload effect. Under certain rare specific criteria or events the Trojan gets triggered and then its payload makes the circuit output erroneous. Trojan triggering process can be classified into three categories [12]: 1) rare value triggered, 2) time-triggered and 3) both time and value triggered. Payload effect on a triggered Trojan depends on the application [13]. It may disable the system or leak the vital information or change the output value of the circuit.

Class of Trojan which only changes the computational output value of an IP, is targeted in this paper. Following threat model is considered in this paper: An IP which is malicious in nature designed either by an adversary/rogue in the third party house may be present within the library of a HLS tool. This IP behaves normally until triggered by a rogue or adversary and therefore detection of such Trojan becomes difficult during normal validation and manufacturing test.

There could also be other Trojans in 3PIPs which produce other effects such as Trojan that disable functionality i.e. does not allow generating any output, Trojan that leak information such as encrypted data/secret key etc and Trojan that inserts unwanted processing delay/creates reliability problem into the IP. However, these ones are not handled in this current paper.

## II. Related Prior Approach

No effort has been done on generating an optimized Trojan security aware schedule for single loop and nested loops CDFGs (based on user constraints) during system level which is capable of providing runtime detection. There are different types of Trojan detection mechanisms available in the literature. Trojans may be inserted in a foundry, therefore different types of detection methods exist: (a) side channel analysis method [14] [15] [16], (b) logic testing method [10] [17] [18] [19] [22] [23]. Moreover, Trojans can also be inserted in an IP, 3PIP or in a design for which detection methods exist such as [20] [21], [10], [18]. Authors in [10], [18] duplicate the IPs in order to detect the presence of a Trojan, however none of the aforesaid approaches handle single and nested CDFG during Trojan detection.

In [14], authors presented a way to construct the fingerprint for all delay paths to detect Trojan circuits. However authors in [15], used the chip characteristics like path delay and power consumption (of the manufactured chips) to detect Trojan within. Authors in [16], proposed a side channel analysis approach to detect the Trojans. The approach is capable of detecting malicious hardware modifications where the Trojan detection is done based on path delay and power of the manufactured chips. Moreover, the approach mentioned in [17], generated a multiple supply transient current integration methodology to detect Trojan within the circuits. In [19] authors worked on code-coverage analysis where list of suspicious signals is identified to detect Trojans. The signals which stay stable during the coverage analysis are considered suspicious since Trojans do not change the state. The analysis includes line, statement, finite state machine (FSM) and path coverage at the RTL. In this approach, multiple test vectors are added to activate the uncovered parts of the design, but this requires a huge verification time. Moreover, redundant circuits are removed from the list of suspicious signals and components since these remain at the same logic level during verification. A sequential automatic test pattern generation (ATPG) is used to generate special patterns to change the signal values during simulation. An equivalence analysis is on the suspicious signal list in order to reduce the number of signals but the runtime overhead incurred is very high. Also, in this approach the quantity of suspicious signals keeps increasing with the increase in the complexity of Trojan inserted. Moreover, it is also difficult to achieve 100% code-coverage for all IPs. Also Trojan detection cannot be accomplished by tests used for detecting manufacturing faults, by analysis of parametric signals and reverse engineering. Moreover, test vectors generation [12] specific to Trojan detection also exists which depend on logic and fault simulator as well as on ATPG tool; however it incurs high runtime overhead/complexity for large number of circuit inputs during testing phase. Further, multiple excitation of rare occurrences (MERO) [22] based compact test pattern generation technique also exists for Trojan detection which starts with a golden netlist, random patterns, rare nodes and number of times to activate a node to rare value. However this technique is sensitive to Trojan sample size and number of times a rare condition is satisfied, in order to create a balance between estimate coverage and simulation time as well as coverage and test vector set size respectively. Without carefully selecting the sensitivity parameters, the results may be imprecise. Further, in case of a 3PIP in HLS, no golden model exists. Additionally in [23], procedure to identify circuit sites where Trojan may be inserted in untrusted foundry is also proposed, however it is not beneficial for detection of Trojan inserted in 3PIP module of HLS library at architectural level. None of the approaches mentioned above considers Trojan security of single and nested loop CDFGs during HLS.

Furthermore, authors in [10, 18] adopted a concurrent error detection (CED) approach for Trojan detection of DFGs. The approaches use a diverse set of 3PIP vendors for Trojan identification. However, the approaches do not have mechanism to explore efficient vendor allocation procedure for Trojan secured hardware. Efficient vendor allocation procedure for secured hardware is critical as it affects the final area-latency of the solution. Not considering exploration of an efficient vendor allocation procedure results in inferior quality solutions (as discussed later in results section). Further in [10, 18], the authors have only targeted DFGs and no efforts have been done to handle single and nested loop based CDFGs. In [20], a simulation based Trojan detection approach is proposed where the statistical correlation of Trojan logic compared to rest of the circuit is weak. A weighted circuit graph is generated based on the input and output of the circuit and their interconnection. The reachability plots identify the weak relation, thus identify the Trojan. In [21] the Trojan detection process is divided into tracer and checker. The tracer identifies those signals which contain un-activated entries. Out of these signals, the checker analyzes to determine the signals which contain redundant inputs and hence potentially affected by Trojan. None of the above techniques focus on solving the design space exploration (DSE) problem of generating an optimized Trojan secured schedule for single and nested loop CDFGs during HLS based on area-delay constraints capable of providing run-time detection. This paper aims to provide a low cost Trojan security aware HLS solution for single loop and nested loop CDFGs that affects computational output based on user specified constraints.

Fig.1. Trojan logic inserted by an adversary in a BCD Adder IP (inputs A & B); Note: When Enable (En) = 0 (rare event), then Trojan blocks get activated and produces incorrect computational output ($O_3O_2O_1O_0$)

## III. Motivational Example: Trojan In 3PIP Module

Let us assume a case during HLS which explains the Trojans in 3PIP module and their effect. Fig. 1 shows an IP of BCD adder designed by an untrusted third party. This IP is included in a HLS tool library. In the given figure, the area marked in red is a possible location where insertion of Trojan is made by malevolent alteration of the IP by an untrusted vendor. Under normal conditions, the module behaves like a functionally correct IP i.e. the upper mux passes '1' and lower mux passes '0' to finally form bits '0110' (decimal of '6'). This value is added to correct any invalid BCD summation produced and yield a result between valid decimal values 0 to 9. However if an adversary triggers the select (S) of the circuit through external or internal activation the Trojan logic will impact the system and change the computational output value. This is the only impact that this class of Trojan makes, as it does not attempt to steal/leak secret information (or cause any other harm). *Note: The existence of Trojan in a 3PIP in a HLS library typically cannot be prevented as it depends on the presence of a rogue element in the third party design house, which a design integrator is unaware of. The only way to minimize the possibility of a Trojan existence during high level synthesis is using IPs from trusted third party sources in tool library. As there is no guarantee of a Trojan free 3PIP in HLS library, therefore low cost Trojan security methodology is critical. Another alternative to prevent Trojan is designing all IPs within in-house design team (i.e. without outsourcing to third party vendors). However, this drastically increases design cycle time and affects design productivity.*

To exploit Trojan for affecting computational output value, an attacker does not need to know the architecture of the system. 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. Activation of the Trojan logic, would directly impact the computational output value of the entire system. Lower level techniques like side channel analysis and RTL simulation cannot detect such Trojans because Trojan logics can only be triggered through specific predefined conditions like FSM counter value, sensing a specific design signal, external antenna etc. Also, Trojans are only activated by a rogue when deployed in real time situation for change in computational output value. Therefore, even after a functional verification check (RTL simulation), the Trojan within IP might get undetected since the IP will perform normally/ correctly due to it being dormant at that stage. This is because of two reasons: (a) the IP cores are generally considered pre-verified functional blocks thus many design integrators do not rigorously perform verification. This provides opportunity to escape any Trojan detection; (b) During functional verification/RTL simulation/other lower level tests (during pre-integration phase), the Trojan logic does not get triggered easily as it gets activated only under rare condition. It normally remains dormant and escapes the detection process performed by in-house design integrator team. For example, consider a scenario for Fig.1 where the BCD adder IP black box provides three input signals Enable (En), A (4-bit) and B (4-bit) and output signal $O_3O_2O_1O_0$. During functional verification/RTL simulation test, En is always be kept 'on' (by feeding logic '1' in test bench) by the design verification team as they falsely believe it to be an IP enable signal (without being aware that it is designed in disguise to act as a trigger signal for Trojan). Under En = 1, the BCD adder IP would behave normally, thus passing RTL simulation/functional verification test. This is because Trojan would have only got activated under rare event (enable reset i.e. En = 0 which normally the design integrator would never make during verification test). Thus after integration, whenever the IP 'En' signal remains 'off', it actually continues to work maliciously (due to Trojan logic activation) thereby impacting the remaining system through wrong output value. Moreover, detection of Trojan in an IP during HLS is very difficult since there is no golden IPs available.

TABLE 1 PROPOSED NOMENCLATURE

| | | | |
|---|---|---|---|
| $p$ | Population size | $V_j$ | Type of vendor |
| $X_i$ | Resource set of a design solution | $C_T^{DMR}$ | Total CS required to execute the loop of CDFG DMR completely |
| $D$ | Total available resource types | $C_{body}^{DMR}$ | Number of CS required to execute loop body of CDFG DMR once |
| $N(R_D)$ | Number of instances of a resource type '*D*' | $I$ | Maximum number of iteration (loop count) |
| $U$ | Candidate unrolling factor value | $C_{first}^{DMR}$ | Number of CS required to execute first iteration of the $CDFG^{DMR}$ |
| $I_1, I_2$ | Corresponding iteration counts of loops | $\Delta$ | Delay of one CS in nanoseconds |
| $A_v$ | Vendor allocation Procedure | $D(op_i^{Vj})$ | Delay of operation *i*, assigned to vendor $V_j$, |
| $A_T^{DMR}$ | Total area of a dual modular redundant (DMR) design | $c.s$ | Control steps |
| $A_{cons}$ | User specified area constraint | $T_{max}^{DMR}$ | DMR solution with maximum time in the design space |
| $A_{max}^{DMR}$ | DMR solution with maximum area in the design space | $C_f(X_i)$ | Cost of solution with resource set $X_i$ |
| $T_E^{DMR}$ | Total execution time of a dual modular redundant (DMR) design | $\overrightarrow{R_n}$ | Resource array |
| $T_{cons}$ | User specified execution time constraint | $R_{d_i}^{+}$ | New resource value or '*U'* value of particle $X_i$ in dth dimension |
| $R_i^{Vj}$ | Number of instances utilized from vendor $V_j$ for a resource type $R_i$ | $V_{d_i}^{+}$ | New velocity of *i*th particle in *d*th dimension |
| $n$ | Maximum number of instances of resource type $R_i$ for vendor $V_j$ | *k* and *'k* | Maximum value of node |
| $A(R_i^{Vj})$ | Area of a resource type ($R_i$) corresponding to vendor ($V_j$) | $D(op_i^{Vj})$ | Delay of operation *i*, assigned to vendor $V_j$ |
| $R_{d_{lbi}}$ | Resource value of $X_{lbi}$ in *d*th dimension | $R_{d_i}$ | Resource value or '*U*' value of particle $X_i$ in *d*th dimension |
| $\omega$ | Inertia weight | $R_{d_{gb}}$ | Resource value of $X_{gb}$ in *d*th dimension |

| | | | |
|---|---|---|---|
| $b_1, b_2$ | Acceleration coefficients which balances the effect of cognitive and social factor during exploration | $r_1, r_2$ | Random numbers providing stochasticity |
| $X_{lbi}^{SL}$, $X_{lbi}^{NL}$ | Local best particle of Single loop and Nested loop | $X_{gb}^{SL}$, $X_{gb}^{NL}$ | Global best particle of Single loop and Nested loop |

## IV. Security Aware HLS: Formulation and Models

The aim of proposed approach is to explore the design space, comprising of candidate solutions for scheduling and vendor assignment, for single and nested loop CDFGs (as discussed in the upcoming sections). The output of the exploration results in a user constraint optimized Trojan detectable schedule for loop based CDFGs. To instantiate the proposed approach, example of area and delay constraints have been considered in the current paper.

### A. Problem Definition

*1) Encoding for Handling Single Loop Based CDFGs:*

Determine an optimal solution,

$X_i = \{N(R_1), N(R_2),..N(R_D), U, A_v\}$,

while exploring the design space of a given single loop based CDFG and satisfying conflicting user constraints and minimizing the overall cost.

*2) Encoding for Handling Nested Loop Based CDFGs:*

Determine an optimal solution,

$X_i = \{N(R_1), N(R_2),..N(R_D), I_1, I_2, A_v\}$,

while exploring the design space of a given nested loop based CDFG and satisfying conflicting user constraints and minimizing the overall cost.

The problem can be formulated as:

Find: *Optimal* $(X_i)$,

with minimum hybrid $\mathrm{Cost}(A_T^{DMR}, T_E^{DMR})$,

subjected to: $A_T^{DMR} \le A_{cons} \, and \, T_E^{DMR} \le T_{cons}$ and Trojan security,

The variables have been defined in Table 1. The details and impact of vendor allocation procedure are explained in later sections.

### B. Evaluation Models

In the proposed work, each particle position represents a hybrid encoding of candidate solutions for scheduling resource configurations ($X_i$), loop unrolling factor ($U$) and vendor assignment procedure information ($A_V$) in the design space [11].

Fig.2. Proposed Security aware PSO-DSE methodology

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

$$A_T^{DMR} = \sum_{j=1}^{2} \sum_{i=1}^{n} (A(R_i^{V_j}) * R_i^{Vj}) \qquad (1)$$

The variables have been defined in Table 1. 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).

*2) Execution Time (Delay) Model:*

*a) For Handling Single Loop Based CDFGs*

The execution delay is evaluated after derivation of the delay model using the following two cases [11]:

**Case 1:** When '*U*' is equal to one (indicates no unrolling) then:

Total # of control steps (CS) = # of CS required executing loop body once * number of duplicate iterations of loop body:

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

Where α= [I/*U*] $^{floor}$

Since *U=1*, $\quad C_T^{DMR} = \left(C_{body}^{DMR} * I\right) \qquad (3)$

The variables have been defined in Table 1. *Note: Eqn. (2) is also 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, *I mod U* iterations will be executed sequentially, therefore, the total no. of CSs is:

$$C_T^{DMR} = \underbrace{\left( C_{body}^{DMR} * \left[ \frac{I}{U} \right]_{floor} \right)}_{\{CSs\ for\ unrolled\ loop\}} + \underbrace{(I \bmod U) * C_{first}^{DMR}}_{\{CSs\ for\ sequential\ loop\}} \qquad (4)$$

The variables have been defined in Table 1. Hence the execution time for the system is calculated as:

$$T_E^{\ DMR} = \Delta * C_T^{\ DMR} \qquad (5)$$

*b) For Handling Nested Loop Based CDFGs:*

The delay for nested loop based CDFG is evaluated by fully unrolling the CDFG on the basis of iteration counts ($I_1$, $I_2$). Further, for given 'D' functional resources the delay is:

$$T_E^{\ DMR} = \sum_{c.s=1}^{c.s(\max)} \sum_{j=1}^{2} Max(D(op_i^{\ Vj}),....D(op_k^{\ Vj}), D(op_{'i}^{\ Vj}),......D(op_{'k}^{\ Vj})) \qquad (6)$$

Where, $1 \le i \le k$ and $'1 \le {}'i \le {}'k$. (Here, operations in original and duplicate is labeled as *i* and *'i* respectively). The variables have been defined in Table 1.

*3) Fitness function:* The fitness function (considering execution time and area consumption of a solution) is inspired from [24] and is formulated as:

$$C_f(X_i) = W_1 \frac{A_T^{\ DMR} - A_{cons}}{A_{\max}^{\ DMR}} + W_2 \frac{T_E^{\ DMR} - T_{cons}}{T_{\max}^{\ DMR}} \qquad (7)$$

The variables have been defined in Table 1. $W_1$ and $W_2$ are the user defined weights both kept at ½ during exploration to provide equal preference.

## V. Proposed Security aware PSO-DSE Methodology

The proposed DSE methodology is driven through PSO process for generation of optimal Trojan secured schedule based on area-delay constraint (shown in Fig. 2). A Trojan secured schedule is obtained as follows: a) duplicate all the operations of original CDFG to form a duplicate one, which together is called '*dual modular redundant system'* b) Perform scheduling of operations of DMR system to obtained scheduled DMR c) Finally, perform hardware allocation of operations of DMR schedule based on Trojan security allocation rules (i.e. assigning hardware from distinct/multiple vendors to similar operations of original and duplicate unit). The proposed methodology presents a novel encoding scheme for the particle which comprises of resource configuration ($\overrightarrow{R_n}$), unrolling factor (*U)* in case of single loop applications, iteration counts ($I_1$, $I_2$) in case of nested loop applications and vendor allocation procedure type ($A_V$). Therefore, a particle position (candidate design solution) is labeled as $X_i$ for :

$$X_i = (\overrightarrow{R_n}, U, A_v) \qquad (8)$$

in case of single loop based CDFGs, while

$$X_i = (\overrightarrow{R_n}, I_1, I_2, A_v) \qquad (9)$$

for nested loop based CDFGs.
Where, $\overrightarrow{R_n}$ indicates the resource array (resource configuration e.g. number of adders, multipliers etc.). The reason behind incorporating the last dimension with vendor allocation procedure type '$A_V$' is elaborated in section V.E.

### *A. Motivation of proposed work*

The proposed work is motivated from the following three uncovered aspects of the literature: **(a)** during detection of Trojans, exploration to optimize the Trojan secured schedule based on user constraint is not performed. **(b)** recent approaches [10] have only attempted to keep distinct hardware (from different vendor) assignment to similar operations in original and duplicate units of a dual modular redundant system to achieve detection but have not probed into a detailed vendor allocation procedure of hardware units inside a DMR schedule (as there can be more than one ways to implement distinct hardware assignment condition). **(c)** Exploration of simultaneous Trojan secured schedule and unrolling factor for single loop based control data flow graphs have not been addressed. Further, integrating the exploration of an efficient vendor allocation procedure concurrently with the aforesaid task has not been performed.

### *B. Pre-processing of unrolling factor candidates*

A pre-processing algorithm for unrolling factors has been deployed to screen the large unfit unrolling factor (which are potential sources for greater delay due to multiple sequential loops involved) candidates before exploration initiates.

### C. Initialization of Particles

#### 1) For Single Loop Based CDFGs

The position of particle as defined earlier in eqn. (8) & (9) can be expanded as follows. The particle position '$X_i$' is given as:

$X_i = (N(R_1), (N(R_2),..(N(R_d).. (N(R_D), U, A_V)$ (10)

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

$X_1 = (min(R_1), min(R_2),.. min(R_D), min (U), A_V)$ (11)

$X_2 = (max(R_1), max(R_2),.. max(R_D), max(U), A_V)$ (12)

$X_3 = ((min(R_1)+max(R_1))/2,......,((min(R_D)+max(R_D))/2, (min(U)+max(U))/2, A_V)$ (13)

While rest of the particles ($X_4...X_n$) is initialized by:-

$X_n = ((min(R_1) + max(R_1))/2 \pm \alpha , (min(R_2) + max(R_2))/2 \pm \alpha, ...(min(R_d) + max(R_d))/2 \pm \alpha ), ...(min(U) + max(U))/2 \pm \alpha, random (A_V))$ (14)

Where, $\alpha$ is a random integer between *min* and *max* of particular resource type or unrolling factor.

#### 2) For Nested Loop Based CDFGs

The position of particle as defined earlier in eqn. (8) & (9) can be expanded as follows. Particle position '$X_i$' is given as:

$X_i = (N(R_1), (N(R_2),..(N(R_d).. (N(R_D), I_1, I_2, A_V)$ (15)

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

$X_1 = (min(R_1), min(R_2),.. min(R_D), I_1, I_2, A_V))$ (16)

$X_2 = (max(R_1), max(R_2),.. max(R_D), max(U), A_V)$ (17)

$X_3 = ((min(R_1)+max(R_1))/2,......,((min(R_D)+max(R_D))/2, (min(U)+max(U))/2, A_V)$ (18)

While rest of the particles ($X_4...X_n$) is initialized by:-

$X_n = ((min(R_1) + max(R_1))/2 \pm \alpha , (min(R_2) + max(R_2))/2 \pm \alpha, ...(min(R_d) + max(R_d))/2 \pm \alpha ), ...(min(U) + max(U))/2 \pm \alpha, random (A_V))$ (19)

Where, $\alpha$ is a random integer between *min* and *max* of particular resource type or unrolling factor.

*Note: The initialization process selects the best positions through all combinations of $A_V$ value. Further, these selected positions are used as initial solutions to begin the exploration.*

### D. Particle Movement using Velocity

In PSO-DSE, each dimension ($d$) of a particle position $X_i$ (except the last dimension) is updated using the following function [24]:

$$R_{d_i}^{+} = R_{d_i} + V_{d_i}^{+} \quad (20)$$

The variable $V_{d_i}^{+}$ is updated by eqn. (21) as follows:

$$V_{d_i}^{+} = \omega V_{d_i} + b_1 r_1 \left[ R_{d_{lbi}} - R_{d_i} \right] + b_2 r_2 \left[ R_{d_{gb}} - R_{d_i} \right] \quad (21)$$

The local best ($X_{lbi}^{SL}$ and $X_{lbi}^{NL}$ for single and nested loop) and global best ($X_{gb}^{SL}$ *and* $X_{gb}^{NL}$ for single and nested loop) particle positions are updated using eqn. (22) & (23) for single loop based applications while using eqn. (24) & (25) for nested loop based applications:

$$X_{lbi}^{SL} = \{R_{1_{lbi}}, ...R_{D-1_{lbi,}} U\} \quad (22)$$

$$X_{gb}^{SL} = \{R_{1_{gb}}, ...R_{D-1_{gb,}} U\} \quad (23)$$

$$X_{lbi}^{NL} = \{R_{1_{lbi}}, ...R_{D-1_{lbi}}, I_1, I_2\} \quad (24)$$

$$X_{gb}^{NL} = \{R_{1_{gb}}, ...R_{D-1_{gb,}} I_1, I_2\} \quad (25)$$

The variables have been defined in Table 1. The particle positions are used to generate a DMR schedule ($SDFG^{DMR}$) with distinct vendor assignment rule to detect the presence of Trojan.

### E. Motivation of using Vendor Type '$A_V$' in Particle Encoding during DSE

The proposed approach uses two distinct/multiple vendors for assigning to similar operations of original and duplicate units. The goal is to perform the similar operations of DMR not only through distinct hardware but also distinct hardware from distinct vendors. This enables original and duplicate version to produce computationally different output in case hardware of one vendor is Trojan infected. However, distinct vendor assignment rule for Trojan detection can be implemented in more than one ways; therefore the most efficient allocation procedure of hardware units from distinct vendor to similar operations (in original and duplicate) is explored through proposed scheme. The schedule with allocation generated indicates a Trojan secured system. The obtained Trojan secured DMR schedule is evaluated on metrics of area and delay. The cost of the Trojan secured scheduling solution explored 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). This internal buffering contributes to the total area since in a DMR similar operation is being done on the two copies (original and 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 Trojan fault secured DMR system that satisfies $A_{cons}$, $T_{cons}$ as well as minimizes hybrid cost.

Fig. 3: Scheduling and Binding of FIR for $X_i$ = 2(+), 2(*), 2(<), $U$=2, I=4 based on Vendor Allocation Mode $A_v$ = 00; $T_E^{DMR}$ = 45080 ns and $A_T^{DMR}$ = 13064 au

Fig. 4: Scheduling and Binding of FIR for $X_i$ = 2(+), 2(*), 2(<), $U$=2, I=4 based on Vendor Allocation Mode $A_v$ = 01; $T_E^{DMR}$ = 43080 ns and $A_T^{DMR}$ = 17996 au

Fig. 5: Scheduling and Binding of FIR for $X_i$ = 2(+), 2(*), 2(<), $U$=2, I=4 based on Vendor Allocation Mode $A_v$ = 10; $T_E^{DMR}$ = 45080 ns and $A_T^{DMR}$ = 13064 au

Fig. 6: Scheduling and Binding of FIR for $X_i$ = 2(+), 2(*), 2(<), $U$=2, I=4 based on Vendor Allocation Mode $A_v$ = 11; $T_E^{DMR}$ = 45070 ns and $A_T^{DMR}$ = 15096 au

Fig. 7(a) ‘C’ code for original loop of Autocorrelation benchmark (adopted from [30])

Fig 7(b): Scheduling and Binding of Auto Correlation Filter for $X_i$ = 2(+), 3(*), 2(<), $I_1$ = 3, $I_2$ = 2, based on vendor allocation mode $A_v$= *00*. $T_E^{DMR}$ = 45350.0ns and $A_T^{DMR}$ = 17996.0au

In order to detect Trojans a minimum of two vendors is always needed to provide distinctness (Note: cases where Trojans lead to disabling hardware units are ignored in this paper, as it falls outside the scope of the proposed work). However, technique of usage of the two vendors during allocation inside the DMR scheduling (i.e. assignment process of each vendor IPs inside the system during allocation) dictates the final latency and area of entire system. This is because 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’ & ‘2034au’, latency = ‘10000ns’ & ‘265ns’ (where values have been extracted from [24]; assuming 1 au = 1 transistor) while multiplier and adder provided by vendor $V_2$ has area = ‘2464au’ & ‘2032au’, latency = ‘11000ns’ & ‘270ns’ respectively. The values of area and delay of modules assumed is with respect to 90nm technology scale adopted from [25], [26], [27]. Hence, merely using distinctive vendor assignment for detection without probing into the procedure of allocation (assignment) of vendor type in DMR system may lead to skipping of an alternate better solution in context of DSE of Trojan secured schedule. Therefore, exploration of an additional dimension ‘$A_v$’ (indicating allocation procedure of IP’s from different vendor type) which can either be in one of the following distinct vendor allocation modes, mode 1:‘00’ or mode 2: ‘01’ or Type 3: ‘10’ or Type 4: ‘11’ is incorporated in the bacterium encoding along with resource array. The value of ‘$A_v$’ as ‘00’ or ‘01’ or ‘10’ or ‘11’ is interpreted as follows:

*Rule 1: Vendor allocation procedure (Type 1):* $A_v = V_xV_y = 00$

- Alternate vendor assignment to operations in control step of a unit. (Example in Fig. 3, operation 1 & 3 assigned alternatively to ‘$V_1$’ and ‘$V_2$’. Next multiplication if any would have been assigned to ‘$V_1$’ alternately).
- Similar operations of both $U^{OG}$ and $U^{DP}$ being assigned to different vendors.

*Rule 2: Vendor allocation procedure (Type 2):* $A_v = V_xV_y = 01$

- All operations of a specific unit being strictly assigned to resources of same vendor type (say: all operations of original unit strictly assigned to vendor ‘$V_1$’ and all operations of duplication to vendor ‘$V_2$’).
- Similar operations of both original unit ($U^{OG}$) and duplicate unit ($U^{DP}$) being assigned to different vendors.

*Rule 3: Vendor allocation procedure (Type 3):* $A_v = V_xV_y = 10$

- All operations within critical path ($p^{cri}$) of a specific unit being strictly assigned to a vendor type while all operations of non critical path through **alternate** vendor type (say all operations of original unit belonging to $p^{cri}$ strictly assigned to vendor ‘$V_1$’ while all operations of non $p^{cri}$ to alternate vendor ‘$V_2$’).
- Operations of critical path of $U^{OG}$ and $U^{DP}$ are assigned to distinct vendors.
- Similar operations of non critical path in both $U^{OG}$ and $U^{DP}$ being assigned to different vendors.

*Rule 4: Vendor allocation procedure (Type 4):* $A_v = V_xV_y = 11$

- **Alternate** vendor assignment to operations belonging to subsequent unrolled nested loop iterations within a unit.
- Similar operations of unrolled nested loop iteration in both $U^{OG}$ and $U^{DP}$ assigned to different vendors.

Fig 8: Scheduling and Binding of Auto Correlation Filter for $X_i$ = 2(+), 3(*), 2(<), $I_1$ = 3, $I_2$ = 2, based on vendor allocation mode $A_v$ = 01. $T_E^{DMR}$ = 43350.0ns and $A_T^{DMR}$ = 31060.0au

Fig 9: Scheduling and Binding of Auto Correlation Filter for $X_i$ = 2(+), 3(*), 2(<), $I_1$ = 3, $I_2$ = 2, based on vendor allocation mode $A_v$ = *10;*$T_E^{DMR}$ = 43340.0ns and $A_T^{DMR}$= 31060.0au

## F. Handling of Single Loop CDFGs: An Example

As described before in section *V.E*, consider cases which use a list scheduled CDFG$^{DMR}$ for *FIR* benchmark unrolled twice with: **(i)** resource constraint of $X_i$ = 2(+), 2(*), 1(<); $U$ = 2; iteration count (I) = 4 $\boldsymbol{A_v}$ **= 00 (**illustrated in Fig. 3). The corresponding $T_E^{DMR}$ = 45080ns and $A_T^{DMR}$ = 13064au **(ii)** $X_i$ = 2(+), 2(*), 1(<); $U$ = 2; iteration count (I) = 4 $\boldsymbol{A_v}$ **= 01(**illustrated in Fig. 4). The corresponding $T_E^{DMR}$ = 43080ns and $A_T^{DMR}$ = 17996 au **(iii)** $X_i$ = 2(+), 2(*), 1(<); $U$ = 2; iteration count (I) = 4, $\boldsymbol{A_v}$ **= 10 (**illustrated in Fig. 5). The corresponding $T_E^{DMR}$ = 45080ns and $A_T^{DMR}$ = 13064au **(iv)** $X_i$ = 2(+), 2(*), 1(<); $U$ = 2; iteration count (I) = 4, $\boldsymbol{A_v}$ **= 11(**illustrated in Fig. 6). The corresponding $T_E^{DMR}$ = 45070ns and $A_T^{DMR}$ = 15096 au.

Clearly a difference is observed in the delay and area of the generated scheduling solutions where all abide the rule of hardware from distinct vendor assignment to similar operations for detectability. The schedules with allocation generated in Fig 3, 4, 5 and 6 are Trojan secured (with two distinct vendors needed), however one is better than the other in different parameter. Only using distinct vendor assignment for hardware without probing into the procedure of allocation of vendor type in DMR system may lead to missing of better solutions in context of DSE. Therefore in context of DSE, it is worth to explore the additional dimension incorporated in the proposed particle encoding in eqn. (8) and (9).

Fig 10: Scheduling and Binding of Auto Correlation Filter for $X_i$ = 2(+), 3(*), 2(<), $I_1$ = 3, $I_2$ = 2, based on vendor allocation mode $A_v$ = *11*; $T_E^{DMR}$ = 45350.0ns and $A_T^{DMR}$ = 26128.0au

## G. Handling of Nested Loop Based CDFGs : An Example

On the basis of discussion in section *IV.E*, consider the four cases as illustrated in Fig. 7(b), 8, 9 and 10; which show the list scheduled nested loop based CDFG for Auto Correlation Filter shown in Fig. 7(a) with $I_1$=3, $I_2$=2 iteration counts respectively with:

- Resource constraint of $X_i$ = 2(+), 3(*), 2(<), $I_1$ = 3, $I_2$ = 2, $A_v$ = 00. The corresponding $T_E^{DMR}$ = 45350.0ns and $A_T^{DMR}$ = 17996.0 au. As seen from Fig. 7(b) operations of $U^{OG}$ are assigned through alternate vendor assignment in a control step, for example, operation 1 and 5 are assigned alternatively to '$V_1$' and '$V_2$'. For Trojan detection distinct vendor assignment is given to $U^{DP}$ based on rule 1.
- Resource constraint of $X_i$ = 2(+), 3(*), 2(<), $I_1$ = 3, $I_2$ = 2, $A_v$ = 01. The corresponding $T_E^{DMR}$ = 43350.0ns and $A_T^{DMR}$ = 31060.0au. For example, in Fig. 8 all operations of $U^{OG}$ are assigned to vendor type '$V_1$'. For Trojan detection distinct vendor assignment is given to $U^{DP}$ based on rule 2.
- Resource constraint of $X_i$ = 2(+), 3(*), 2(<), $I_1$ = 3, $I_2$ = 2, $A_v$ = 10. The corresponding $T_E^{DMR}$ = 43340.0ns and $A_T^{DMR}$ = 31060.0au. From Fig. 9 it can be seen that, all operations belonging to $p^{cri}$ (1, 2, 4, 5, 6, 8, 9, 10, 12) of $U^{OG}$ are assigned to vendor type $V_1$, while all operations belonging to $p^{non\text{-}cri}$ (3, 7, 11, 15, 16, 13, 14) are assigned through alternate vendor type. For Trojan detection distinct vendor assignment is given to $U^{DP}$ based on rule 3.
- Resource constraint of $X_i$ =2(+), 3(*), 2(<), $I_1$ = 3, $I_2$ = 2, $A_v$ = 11. The corresponding $T_E^{DMR}$ = 45350.0ns and $A_T^{DMR}$ = 26128.0au. $G_n$ is the unrolled nested loop iterations; where, $G_1$, $G_2$, $G_3$ are unrolled nested loop with respect to first, second and third outer loop iteration respectively. In Fig. 10, operations belonging to subsequent unrolled nested loop iterations are assigned through alternate vendor procedure. For example, operations 1, 2, 3 and 4 belonging to $G_1$ are assigned to $V_1$, while operations 5, 6, 7 and 8 belonging to $G_2$ are assigned to $V_2$ respectively. For Trojan detection distinct vendor assignment is given to $U^{DP}$ based on rule 4.

It can be seen from that there is a difference in the area and delay values of the generated Trojan secured scheduling solutions, though all have distinct vendor assignment to similar operations of original and duplicate unit for detectability. The CDFG$^{DMR}$ schedules with vendor allocation details generated in Fig. 7(b), 8, 9 and 10 are Trojan secured (with two vendors required in all cases) with duplication of all the operations of the original. For example, suppose we have a candidate design solution $X_i$ = 2(+), 3(*), 2(<), $I_1$ = 3, $I_2$ = 2, $A_v$ for building a Trojan secured DMR schedule. This indicates that the nested CDFG has two loops ($I_1$

and $I_2$, with $I_1$ as outer loop and $I_2$ as the inner loop, see Fig. 7(a)), where 2 iterations of $I_2$ are carried out with 3 iterations of $I_1$ and duplication is done for all the operations corresponding to the iteration count to create a complete unrolled CDFG$^{DMR}$. Once the CDFG$^{DMR}$ is created both the original and duplicate unit will be scheduled based on the resource constraint information available in $X_i$: 2(+), 3(*), 2(<). However, it can be seen that one schedule is better than its counterpart in either area or execution delay depending on the mode $A_v$, irrespective of the fact that the schedule resources and iteration counts are same. Therefore in context of low cost scheduling, exploration of $A_v$ (which can either be Mode 1: '00' or Mode 2: '01' or Mode 3: '10' or Mode 4: '11') dimension along with resource array is important to design a Trojan secured schedule for nested loop applications.

Once the schedules and the corresponding $T_E^{DMR}$ and $A_T^{DMR}$ are obtained, the cost of obtained Trojan secured DMR schedule is further evaluated using eqn. (7). This process of evaluating design solutions (particle positions) evolves through the proposed DSE using PSO to generate an optimal Trojan fault secured DMR system that satisfies $A_{cons}$, $T_{cons}$ as well as minimizes hybrid cost.

TABLE 2 RESULTS OF PROPOSED APPROACH

| **Benchmark [29]** | $T_{cons}$ **(us)** | $T_E^{DMR}$ **(us)** | $A_{cons}$ **(au)** | $A_T^{DMR}$ **(au)** |
|---|---|---|---|---|
| AUTOCORRELATION FILTER ($I_1$= 2, $I_2$=3) | 50.04 | 34.62 | 19000 | 17996 |
| AUTOCORRELATION FILTER ($I_1$= 3, $I_2$=3) | 65.00 | 35.43 | 23000 | 22928 |
| AUTOCORRELATION FILTER ($I_1$= 4, $I_2$=2) | 55.00 | 24.43 | 28000 | 27860 |
| DHMC ($I_1$= 2, $I_2$=3) | 420.00 | 262.33 | 35000 | 33524 |
| DHMC ($I_1$= 3, $I_2$=3) | 620.00 | 507.35 | 35000 | 26994 |
| DHMC ($I_1$= 4, $I_2$=2) | 550.00 | 327.80 | 36000 | 35992 |
| DIFFERENTIAL EQUATION (U = 4) | 182.21 | 120.81 | 43301 | 26994 |
| FFT (U = 2) | 200.00 | 156.85 | 41000 | 26128 |
| FIR (U = 6) | 40.00 | 23.62 | 27000 | 22928 |
| TEST CASE (U = 6) | 130.00 | 99.00 | 29000 | 22062 |

Note: 1a.u. = 1 Transistor; us = microseconds

## *H. Velocity clamping , Resource Clamping and Stopping Criterion*

The velocity clamping is necessary to control unwarranted drift control. Additionally, resource clamping is performed to control 'swarm explosion'. Both clampings have been adopted from [24]. 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 '£' number of iteration. (£=10) ($S_2$).

## VI. EXPERIMENTAL RESULTS

This section shows the results of the proposed methodology, divided into following sub-sections: (a) Experimental setup and benchmarks which discusses the machine setup, control parameters and benchmarks used (b) Analysis of results which presents the proposed results for single and nested loop based CDFGs based on user constraints and (c) Comparison with related prior research in terms of implementation cost and final architectural solution explored.

TABLE 3 RESULTS OF APPROACH [10]

| Benchmark [29] | $T_{cons}$ (us) | $T_E^{DMR}$ (us) | $A_{cons}$ (au) | $A_T^{DMR}$ (au) |
|---|---|---|---|---|
| AUTOCORRELATION FILTER ($I_1$= 2, $I_2$=3) | 50.04 | 33.81 | 19000 | 30396 |
| AUTOCORRELATION FILTER ($I_1$= 3, $I_2$=3) | 65.00 | 45.08 | 23000 | 31278 |
| AUTOCORRELATION FILTER ($I_1$= 4, $I_2$=2) | 55.00 | 45.08 | 28000 | 31404 |
| DHMC ($I_1$= 2, $I_2$=3) | 420.00 | 286.35 | 35000 | 52824 |
| DHMC ($I_1$= 3, $I_2$=3) | 620.00 | 440.89 | 35000 | 66054 |
| DHMC ($I_1$= 4, $I_2$=2) | 550.00 | 327.80 | 32000 | 35992 |
| DIFFERENTIAL EQUATION (U = 4) | 182.21 | 131.27 | 43301 | 29640 |
| FFT (U = 4) | 200.00 | 179.24 | 41000 | 33974 |
| FIR (U = 6) | 40.00 | 34.08 | 27000 | 24692 |
| TEST CASE (U = 6) | 130.00 | 99.27 | 29000 | 23826 |

Note: 1a.u. = 1 Transistor; us = microseconds

## *A. Experimental Setup and Benchmarks*

The proposed approach as well as [10] both have been implemented in java and run on Intel Core-i5-3210M CPU with 3MB L3 cache memory, 4GB DDR3 primary memory and processor frequency of 2.5 GHz. During experiment, 15 runs were executed for proposed PSO-DSE. During the experiments conducted, it was found that the proposed approach is scalable and is able to handle problems of medium/large size. Further during the experiments, the following optimal settings from [24] for PSO framework were fixed: $\omega$ (inertia weight) = linearly decreasing between 0.9 to 0.1; *b1* & *b2* (acceleration coefficient) = 2; *r1* & *r2* (random numbers) = 1; stopping criterion = $S_1$ or $S_2$; *p* (swarm size) = *3* or *5* or *7*. The benchmarks used during experiments comprises of loop based control intensive applications derived from multimedia and signal processing domain. More explicitly, single and nested loop based control data flow graphs were used as standard benchmarks from [23, 25]. These benchmarks are typically data and control intensive applications which consume huge power and processing delay. The benchmark application is made up of operations that utilize third party micro IPs such as adder, subtractor, comparator etc. present in the tool library. For evaluating the Trojan detection capability (i.e. testing purpose) of the proposed approach, Trojan logic was inserted into these micro 3PIPs by altering the digital logic of the circuit through a combination of additional gates, multiplexers, flip-flops, control signals etc. A demonstrative example which shows the procedure for Trojan insertion in BCD adder present as 3PIP module in HLS tool library is shown in Fig. 1 (section III). This type of modelling is realistic as mostly in real life cases the Trojan gets triggered by activation of a control line (such as ‘En’ in Fig.1).

TABLE 4 COMPARISON OF PROPOSED DSE APPROACH WITH [10]

| Benchmark [29] [30] | Final design solution for Trojan Secured schedule (Proposed) | Final design solution for Trojan Secured schedule [10] | Cost of final solution (proposed) | Cost of final solution [10 ] |
|---|---|---|---|---|
| AUTOCORRELATION FILTER ($I_1$= 2, $I_2$=3) | 2(+), 4(*), 2(<), $I_1$= 2, $I_2$=3, $A_v$ =00 | 3(+), 5(*), 2(<), $I_1$= 2, $I_2$=3, $A_v$ = 01 | -0.134 | 0.096 |
| AUTOCORRELATION FILTER ($I_1$= 3, $I_2$=3) | 2(+), 8(*), 2(<), $I_1$= 3, $I_2$=3, $A_v$ =00 | 3(+), 5(*), 2(<), $I_1$=3, $I_2$=3, $A_v$ = 01 | -0.149 | 0.025 |

| | | | | |
|---|---|---|---|---|
| AUTOCORRELATION FILTER ($I_1$= 4, $I_2$=2) | 2(+), 8(*), 2(<), $I_1$= 4, $I_2$=2, $A_v$ =00 | 3(+), 5(*), 2(<), $I_1$= 4, $I_2$=2, $A_v$ = 01 | -0.173 | -0.017 |
| DHMC ($I_1$= 2, $I_2$=3) | 4(+), 6(*), 2(<), $I_1$=2, $I_2$=3, $A_v$ =00 | 3(+), 5(*), 2(<), $I_1$= 2, $I_2$=3, $A_v$ = 01 | -0.140 | 0.057 |
| DHMC ($I_1$= 3, $I_2$=3) | 3(+), 4(*), 2(<), $I_1$=3, $I_2$=3, $A_v$ =00 | 3(+), 5(*), 2(<), $I_1$=3, $I_2$=3, $A_v$ = 01 | -0.131 | 0.173 |
| DHMC ($I_1$= 4, $I_2$=2) | 4(+), 6(*), 2(<), $I_1$=4, $I_2$=2, $A_v$ =00 | 3(+), 5(*), 2(<), $I_1$= 4, $I_2$=2, $A_v$ = 01 | -0.136 | -0.103 |
| DIFFERENTIAL EQUATION (U = 4) | 2(+), 2(-), 5(*), 2(<),$U$ =4, $A_v$ =00 | 2(+), 2(-), 5(*), 2(<), U = 4, $A_v$ = 01 | -0.232 | -0.193 |
| FFT (U = 4) | 4(+), 2(-), 4(*), 1(<),$U$ =2, $A_v$ =00 | 5(+), 2(-), 5(*), 2(<), U = 4, $A_v$ = 01 | -0.185 | -0.088 |
| FIR (U = 6) | 4(+), 8(*), 2(<),$U$ =6, $A_v$ =00 | 3(+), 5(*), 2(<), U = 6, $A_v$ = 01 | -0.175 | -0.074 |
| TEST CASE (U = 6) | 4(+), 4(*), 2(<), $U$=3, $A_v$ =00 | 3(+), 3(*), 2(<), U = 6, $A_v$ = 01 | -0.191 | -0.166 |

## B. *Analysis of Results*

### 1) *For Single LoopBased CDFGs*

For single loop based applications, the proposed approach is able to attain final solutions which lie within the user provided constraints of area and execution delay (as well as minimizes the hybrid cost in eqn. (7)), as shown in Table 2. For example, for Differential Equation benchmark, the proposed approach generates the final solution, $(X_i)$*: N*$(R_{adder})$*, N*$(R_{multiplier})$*, N*$(R_{comparator})$*, U,* $A_v$ = 3(+), 3(-), 5(*), 2(<), U = 4, $A_v$ = 00, with $T_E^{DMR}$ = 120.81us and $A_T^{DMR}$ = 29136 au, which completely satisfies the area-delay constraints ($A_{cons}$ = 43301 au and $T_{cons}$ = 182.21us). (*NOTE:* $A_{cons}$ and $T_{cons}$ could be any value between the minimum ( $A_{\min}^{DMR}$ ,$T_{\min}^{DMR}$) and maximum value $\left(A_{\max}^{DMR},T_{\max}^{DMR}\right)$). Similar behavior is observed for the other benchmarks.

### 2) *For Nested Loop Based CDFGs*

In case of nested loop based CDFGs, the proposed approach is able to attain final solutions which lie within the user provided constraints of area and execution delay (as well as minimizes the hybrid cost in eqn. (7)), as shown in Table 2. For example, for Auto correlation Filter benchmark, the proposed approach generates the final solution, $(X_i)$*: N*$(R_{adder})$*, N*$(R_{multiplier})$*, N*$(R_{comparator})$, $I_1$, $I_2$, $A_v$ = 2(+), 4(*), 2(<), $I_1$= 2, $I_2$=3, $A_v$ =00, with $T_E^{DMR}$ = 34.620us and $A_T^{DMR}$ = 17996.00au, which completely satisfies the area-delay constraints ($A_{cons}$ = 19000.00 au and $T_{cons}$ = 50.04us).

## C. *Comparison with Related Prior Research*

It is important to note that there is no work in the literature that explores the low cost Trojan secured schedule either single-dimensionally (comprising of only scheduling resources) or multi-dimensionally (comprising of scheduling resources ($\overrightarrow{R_n}$), '*U' (for single loop)* or $I_1$, $I_2$, *(for nested loops)* with respect to user specified constraints.

TABLE 5 COMPARISON OF EXPLORATION TIME FOR FINDING A LOW COST TROJAN SECURED DATAPATH SOLUTION WITH RESPECT TO SWARM SIZE P FOR PROPOSED APPROACH

| **Benchmark [29] [30]** | **Swarm Size, *p*** | **Exploration Time (in ms)** |
|---|---|---|
| AUTOCORRELATION FILTER ($I_1$= 2, $I_2$=3) | 3 | 2144 |
| | 5 | 4305 |
| | 7 | 4893 |
| AUTOCORRELATION FILTER ($I_1$= 3, $I_2$=3) | 3 | 3297 |
| | 5 | 4091 |
| | 7 | 8080 |
| AUTOCORRELATION FILTER ($I_1$= 4, $I_2$=2) | 3 | 4928 |
| | 5 | 5536 |
| | 7 | 7860 |
| DHMC | 3 | 21836 |

| | | |
|---|---|---|
| ($I_1$= 2, $I_2$=3) | 5 | 43218 |
| | 7 | 62345 |
| DHMC ($I_1$= 3, $I_2$=3) | 3 | 29823 |
| | 5 | 49876 |
| | 7 | 68243 |
| DHMC ($I_1$= 4, $I_2$=2) | 3 | 29066 |
| | 5 | 45327 |
| | 7 | 65827 |
| DIFFERENTIAL EQUATION (U = 4) | 3 | 2084 |
| | 5 | 6359 |
| | 7 | 9243 |
| FFT (U = 4) | 3 | 6856 |
| | 5 | 9132 |
| | 7 | 9205 |
| FIR (U = 6) | 3 | 638 |
| | 5 | 1132 |
| | 7 | 1775 |
| TEST CASE (U = 6) | 3 | 1156 |
| | 5 | 3382 |
| | 7 | 4537 |

Note: ms = milliseconds

The costs function for both [10] and proposed approach, both considers the area of single comparator/error detection block responsible for runtime Trojan detection at the final output. However, the cost function does not have any impact of comparator/error detection block as it is used in both approaches. Table 3 shows the results of the approach [10], while Table 4 shows the comparative results of the proposed approach with [10], in terms of cost and final Trojan secured hardware solution found. Although [10] does not handle single and nested loop CDFGs during Trojan security, however for comparison purpose we have fully re-implemented [10] by feeding the completely unrolled version of each CDFG (both single and nested loop). Complete unrolling of the CDFG (by eliminating the loop converts it into a straightforward DFG) was required because [10] cannot handle loop unrolling factor concurrently with hardware resource configuration during exploration. As seen the proposed approach generates substantially better results in comparison to [10]. This is due to following reasons:

(a) The proposed approach comprises of a supplementary option in encoding for exploration of an optimal allocation procedure of vendors $A_v$, which bears an impact on the final optimization quality;

(b) In case of single loop based applications, 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.

It can be seen from the results, that even though for some benchmarks the number of resources in the final solution for [10] are lower than the proposed solution, it has no optimization of loop unrolling (for single loop CDFGs) and vendor allocation procedure. Since there is no feature for exploring an optimal unrolling factor and distinct vendor allocation in [10] hence it uses the maximum unroll factor value as well as same vendor allocation to each unit rule ($A_v$ = 01) during DMR design thereby resulting in higher cost. Therefore, distinct vendor allocation type $A_v$ =00 always yields lower cost final solution than $A_v$ = 01.

Further, Table 5 highlights the impact of population size *p* on the runtime of proposed approach. It can be seen that for all benchmarks, the runtime of the proposed approach to find the solutions escalates with the growth in population size. This is due to the increase in computational complexity per iteration. However, there is no improvement in the quality of the solution found.

## *D. Detection ability of solutions generated*

The proposed approach deals with Trojan in the IP that change its computational value. However, during real time deployment, the Trojan becomes effective when triggered (by an adversary) through a rare event value, thereby remaining inactive before that. Since, triggering through rare event value is performed through intelligent disguise hence it is very difficult to detect Trojan during RTL simulation/other lower level tests by a trustworthy in-house design integrator. Detailed explanation on how a sample Trojan gets triggered only through a rare event value is explained in section III.

*Type of Trojan inserted for evaluation:* The work presented in this paper targets only a specific class of Trojan. Trojans that on triggering only change the output computational value (and have no other effect) are inserted within the components of HLS library for testing purpose i.e. the IPs were modelled by inserting malevolent logic in the module (such as adder/subtractor etc) of the HLS library. Further for testing purpose during experiments, the inserted Trojan was triggered for evaluating the detection capability of the proposed approach. An example of such a Trojan and its payload is shown in Fig. 1. Additionally, a sequence of test vectors generated through 8th order polynomial Linear-Feedback Shift Register (LFSR) as ATPG are fed into the DMR schedule for comprehensive coverage of Trojan detection in the final explored solution. (*Note: there are no specific test sets that make detection harder or easier*). The results generated indicate that a detection rate of 100% was achieved while handling such Trojans. This is because these Trojans on activation only change the computational value in the output from original and duplicate unit as payload without producing any other effect.

## VII. Conclusion

This paper presented a novel low cost methodology for optimized Trojan secured scheduling at behavioural level for control data flow graphs (single and nested loop). It is capable of providing secured information processing during high level synthesis. Results indicated improvement in quality of final solution obtained compared to a similar related work.

References

[1] M. Tehranipoor, and F. Koushanfar, "A Survey of Hardware Trojan Taxonomy and Detection," *IEEE Design & Test of Computers,* 2010, pp.10-25.

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

[3] 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.

[4] 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.

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

[6] 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.

[7] 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.

[8] S. Bhunia, M. Abramovici, D. Agrawal, P. Bradley, M.S. Hsiao, J. Plusquellic and M. Tehranipoor, "Protection Against Hardware Trojan Attacks: Towards a Comprehensive Solution," in *IEEE Proc. Design & Test,* 2013, pp.6-17.

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

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

[11] S. Bhadauria, A. Sengupta, "Secure Information Processing during System level: Exploration of an Optimized Trojan Secured Datapath for CDFGs during HLS based on User Constraints", *Proceedings of IEEE iNIS,* pp. 1 - 6, Dec 2015.

[12] F. Wolf, C. Papachristou, S. Bhunia and R. S. Chakraborty, "Towards Trojan-Free Trusted ICs: Problem Analysis and Detection Scheme," *In Design, Automation and Test in Europe (DATE'08),* pp. 1362–1365, 2008.

[13] X. Wang, M. Tehranipoor and J. Plusquellic, "Detecting Malicious Inclusions in Secure Hardware: Challenges and Solutions," *In IEEE International Workshop on Hardware-Oriented Security and Trust (HOST'08),* pp.15–19, 2008.

[14] J. Yier and Y. Makris, "Hardware Trojan detection using path delay fingerprint," *IEEE International Workshop on Hardware-Oriented Security and Trust, 2008. HOST 2008.,* pp.51 - 57.

[15] D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi and B. Sunar, "Trojan Detection using IC Fingerprinting," *IEEE Symposium on Security and Privacy, 2007. SP '07.* 2007, pp.296-310.

[16] S. Narasimhan, Du. Dongdong, R.S. Chakraborty, S. Paul, F. Wolff, C. Papachristou, K. Roy, S. Bhunia, "Multiple-parameter side-channel analysis: A non-invasive hardware Trojan detection approach," *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), 2010,* pp.13-18.

[17] W. Xiaoxiao, H. Salmani, M. Tehranipoor and J. Plusquellic, "Hardware Trojan Detection and Isolation Using Current Integration and Localized Current Analysis," *IEEE International Symposium on Defect and Fault Tolerance of VLSI Systems, 2008,* pp.87-95.

[18] C. Xiaotong, M. Kun, S. Liang, K. Wu, "High-level synthesis for run-time hardware Trojan detection and recovery", *51st/IEEE Design Automation Conference (DAC),* California, 2014, pp. 1 – 6.

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

[20] B. Cakir, S. Malik, "Hardware Trojan Detection for Gate-level ICs Using Signal Correlation Based Clustering," *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition,* pp. 471-476, 2015.

[21] J. Zhang, F. Yuan, L. Wei, Z. Sun, and Q. Xu, "VeriTrust: Verification for Hardware Trust," *Proceedings of the 50th Annual Design Automation Conference*, pp. 61:1-61:8, 2013.

[22] R. S. Chakraborty, F. Wolff, S. Paul, C. Papachristou, and S. Bhunia,"MERO: A Statistical Approach for Hardware Trojan Detection," *In International Conference on Cryptographic Hardware and Embedded Systems (CHES'09),* pp. 396–410, 2009.

[23] S. Dupuis, P-S. Ba, M-L. Flottes, G. D. Natale and B. Rouzeyre, "New Testing Procedure for Finding Insertion Sites of Stealthy Hardware Trojans," *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*, pp.776-781, 2015.

[24] A Sengupta, VK Mishra, "Swarm Intelligence Driven Simultaneous Adaptive Exploration of Datapath and Loop Unrolling Factor during Area-Performance Tradeoff", *Proceedings of 13th IEEE Computer Society Annual International Symposium on VLSI (ISVLSI),* Florida, 2014, pp. 106 – 112.

[25] 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), 2010, pp. 2291–2297.

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

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

[28] 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.

[29] [Online]. Available: http://express.ece.ucsb.edu/benchmark/, 2015.

[30] Vipul Kumar Mishra, Anirban Sengupta, "Swarm Inspired Exploration of Architecture and Unrolling Factors for Nested Loop Based Application in Architectural Synthesis", *IET Electronics Letters*, Volume 51,Issue: 2, Jan 2015, pp. 157 - 159.

## Biography of Authors:

**Dipanjan Roy** is a Ph.D candidate in discipline of Computer Science & Engineering at Indian Institute of Technology (I.I.T). His research interests include high level synthesis, hardware security and optimization. He has prior work experience in industries such as Amazon.

**Dr. 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. & M.A.Sc. in Electrical & 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.He holds an external affiliation as 'Honorary Chief Scientist' at VividSparks IT Solutions Pvt Ltd, besides his regular affiliation.
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 & Technology (Science & Engineering Research Board), Govt. of India as well as supported by Intel Corporation and Department of Electronics & IT (DEITY), Govt. of India. He has more than 100 Publications & Patents which include Journals, Patents and Invited Book Chapters from IEEE, IET, Elsevier, Springer and USPTO/CIPO/IPO. 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. For his excellence in doctoral research, he has been awarded/nominated by Ministry of Training, Colleges and Universities, Ontario for multiple years through OGS as well as by Ryerson University through GREA, RGA and NSERC ICA for consecutive years.
He currently serves in Editorial positions of 8 IEEE, Elsevier, & IET Journals as Editorial Board Member, Associate Editor, Columnist and Guest Editor , He is currently serving as Editorial Board Member of Elsevier Microelectronics Journal,. Associate Editor of IET Journal on Computer & Digital Techniques, Executive Editor & Columnist of IEEE Consumer Electronics(M-CE), Associate Editor of IEEE VLSI Circuits & 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 IEEE ICIT. He has supervised 4 Ph.D. candidates (2 completed and 2 pursuing) and 6 RA/B.Eng. students, many of whom are/were placed in industries and academia.

**Dr. Saumya Bhadauria** is currently an Assistant Professor in Indian Institute of Information Technology Kota. She holds a Ph.D from Computer Science and Engineering at Indian Institute of Technology (I.I.T) Indore. Her thesis and research area includes high level synthesis, hardware security, optimization, digital design, evolutionary algorithms etc.

**Highlights**

- Low cost optimized Trojan secured HLS approach
- Handles Trojan security of single or nested loop control data flow graphs (CDFG) applications.
- Multiple novel vendor allocation schemes as security constraint

Figure 1

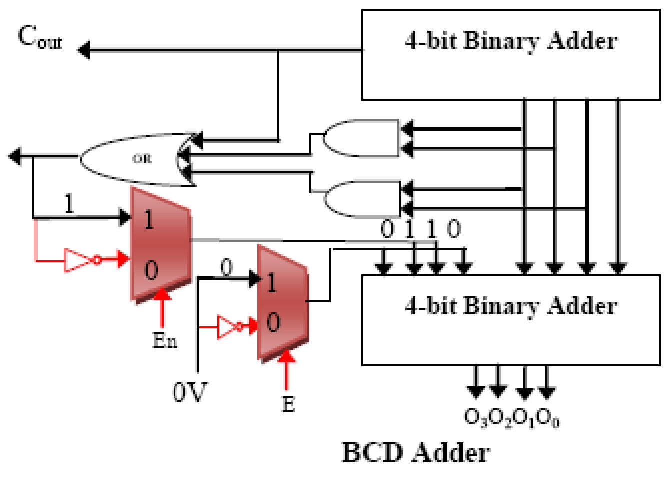
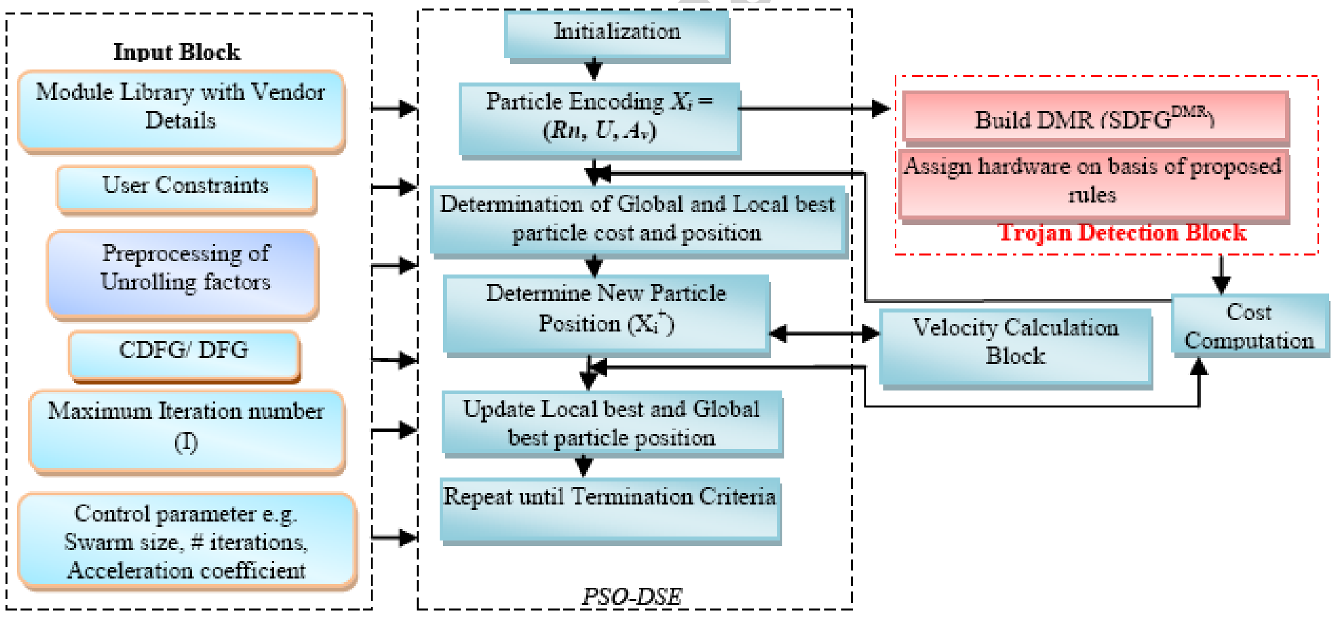


Figure 2



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
Initialization
Particle Encoding $X_i$ = ($Rn$, $U$, $A_v$)
Determination of Global and Local best particle cost and position
Determine New Particle Position ($X_i^+$)
Update Local best and Global best particle position
Repeat until Termination Criteria
PSO-DSE
Build DMR (SDFG$^{DMR}$)
Assign hardware on basis of proposed rules
Trojan Detection Block
Velocity Calculation Block
Cost Computation

Figure 3

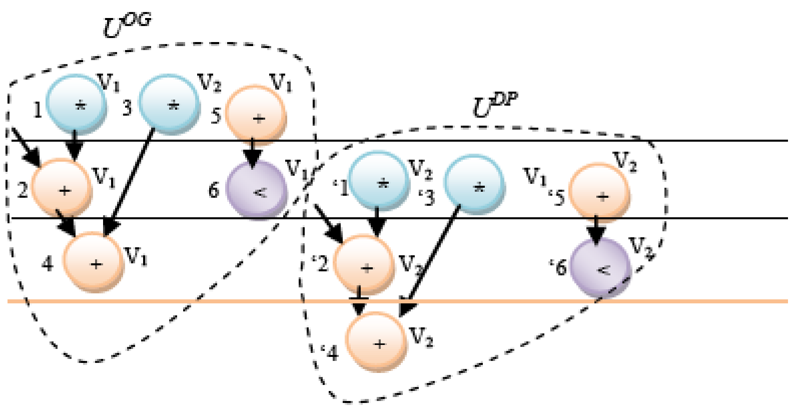


Figure 4

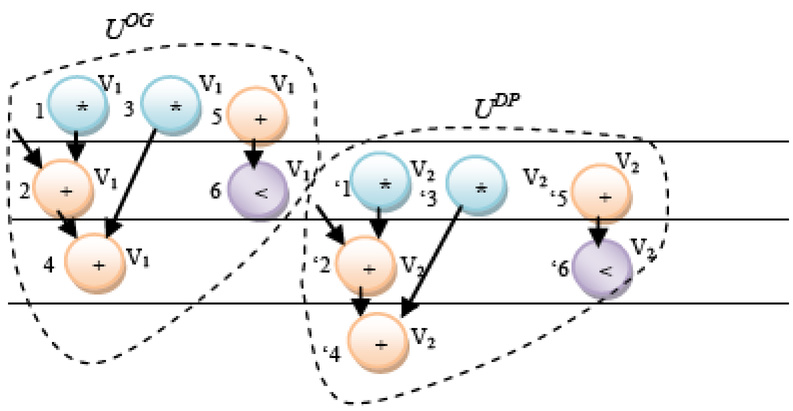


Figure 5

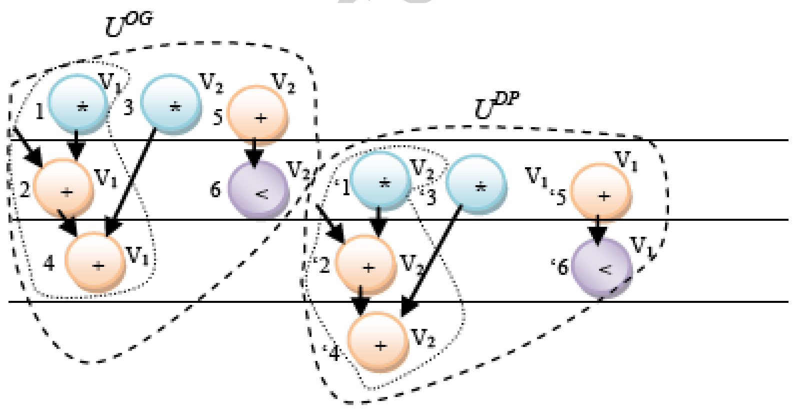


Figure 6

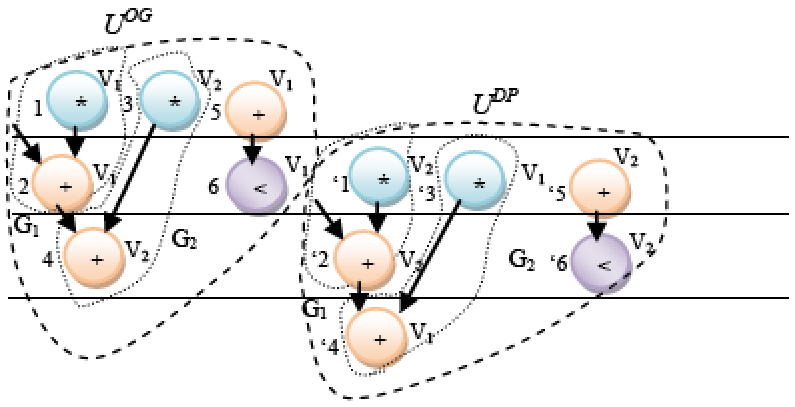
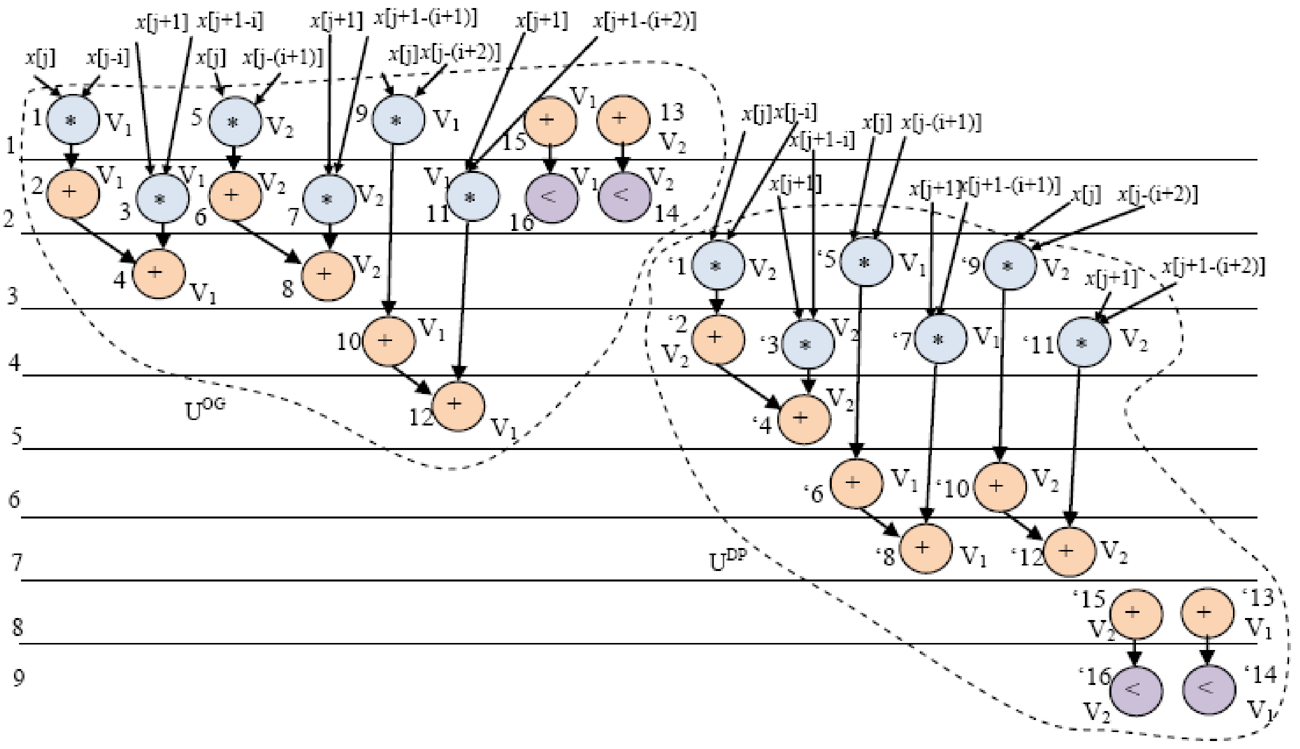


Figure 7(a)

```
For (i=0; i<I_1; i++) // I_1 = Outer loop
{
            For (j=0; j< I_2; j++) I_2 = Inner loop
            {
                        A_i+=B_j*B_{j-i};
            }
}
```

Figure 7(b)



x[j+1] x[j+1-i]
x[j+1] x[j+1-(i+1)]
x[j+1] x[j+1-(i+2)]
x[j] x[j-i]
x[j] x[j-(i+1)]
x[j] x[j-(i+2)]
1 2 3 4 5 6 7 8 9
V1
V2
U^OG
U^DP
x[j+1-i]
x[j+1-(i+1)]

Figure 8

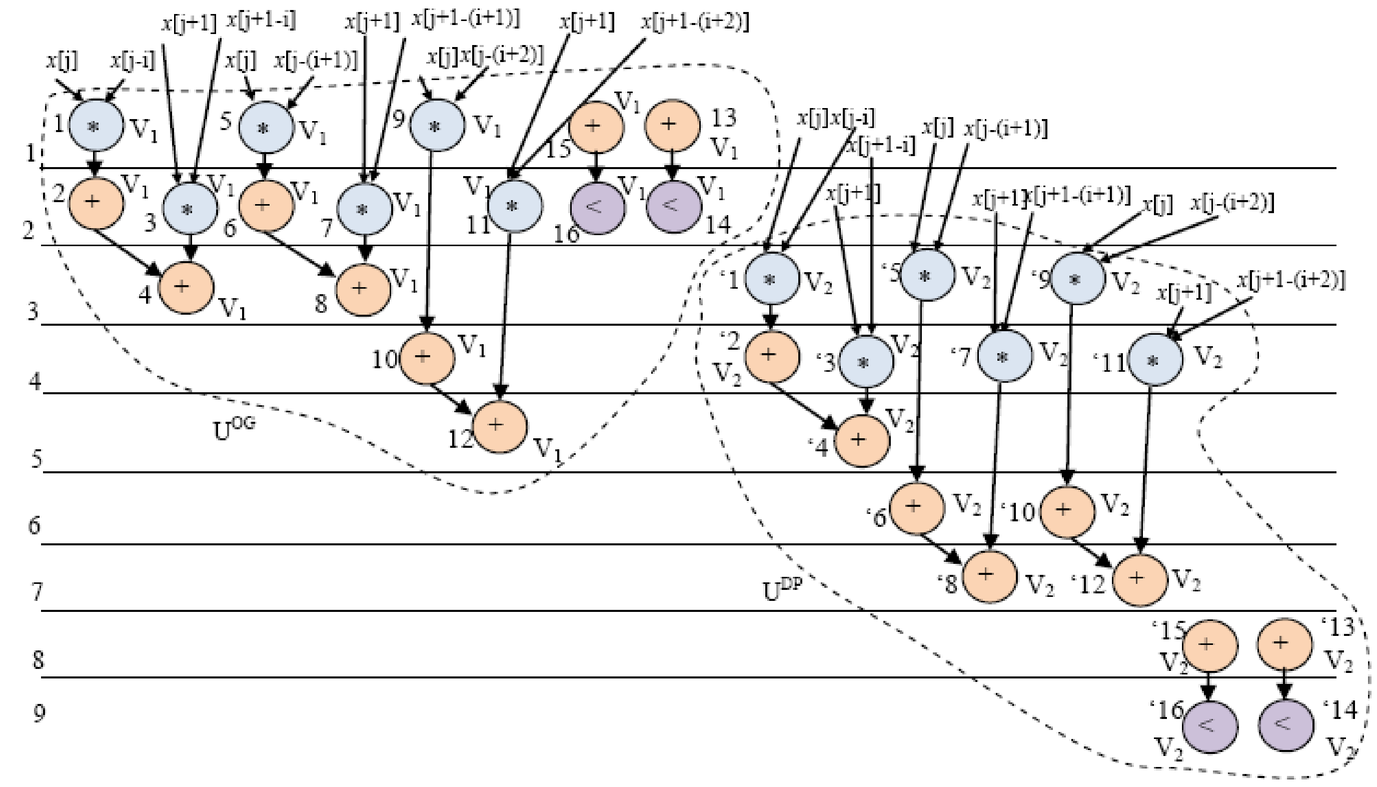

x[j]
x[j-i]
x[j+1]
x[j+1-i]
x[j]
x[j-(i+1)]
x[j+1]
x[j+1-(i+1)]
x[j]
x[j-(i+2)]
x[j+1]
x[j+1-(i+2)]
1
2
3
4
5
6
7
8
9
V1
V2
U^OG
U^DP

Figure 9

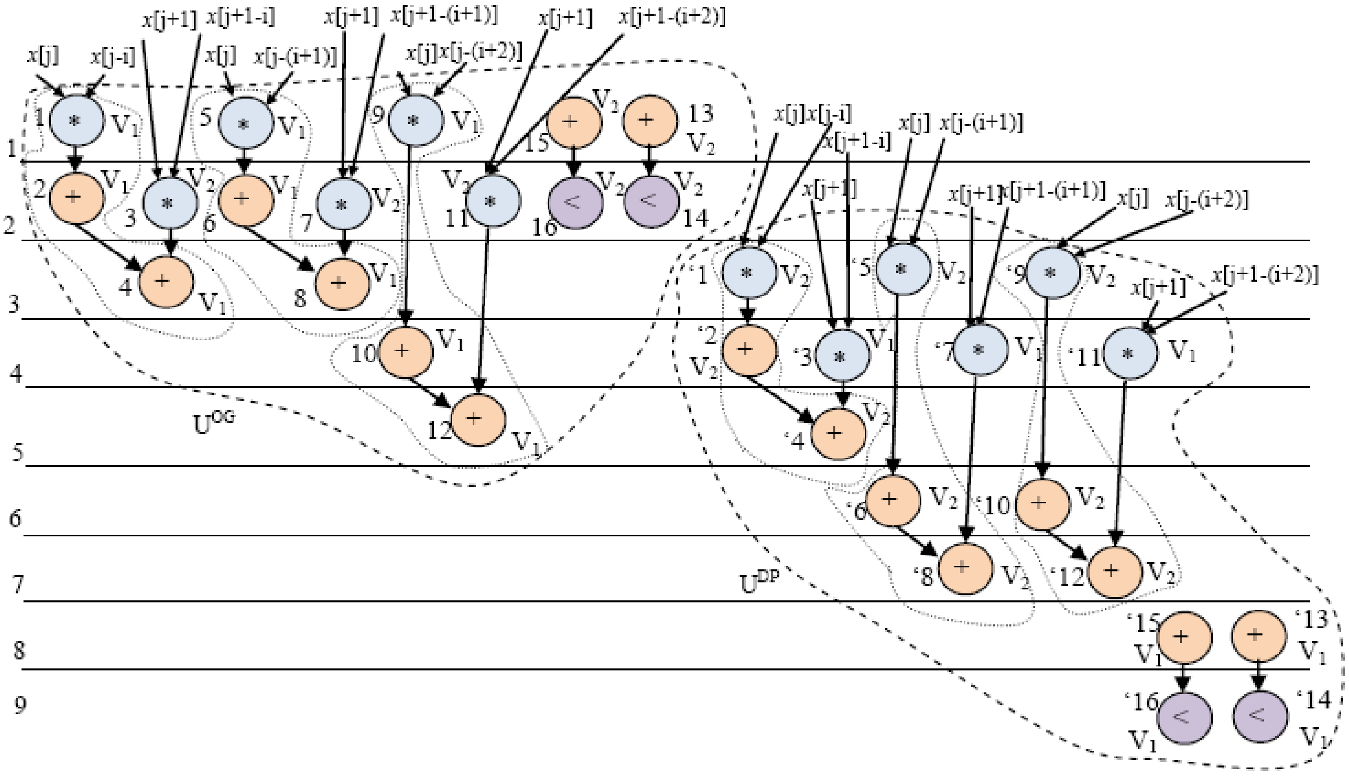
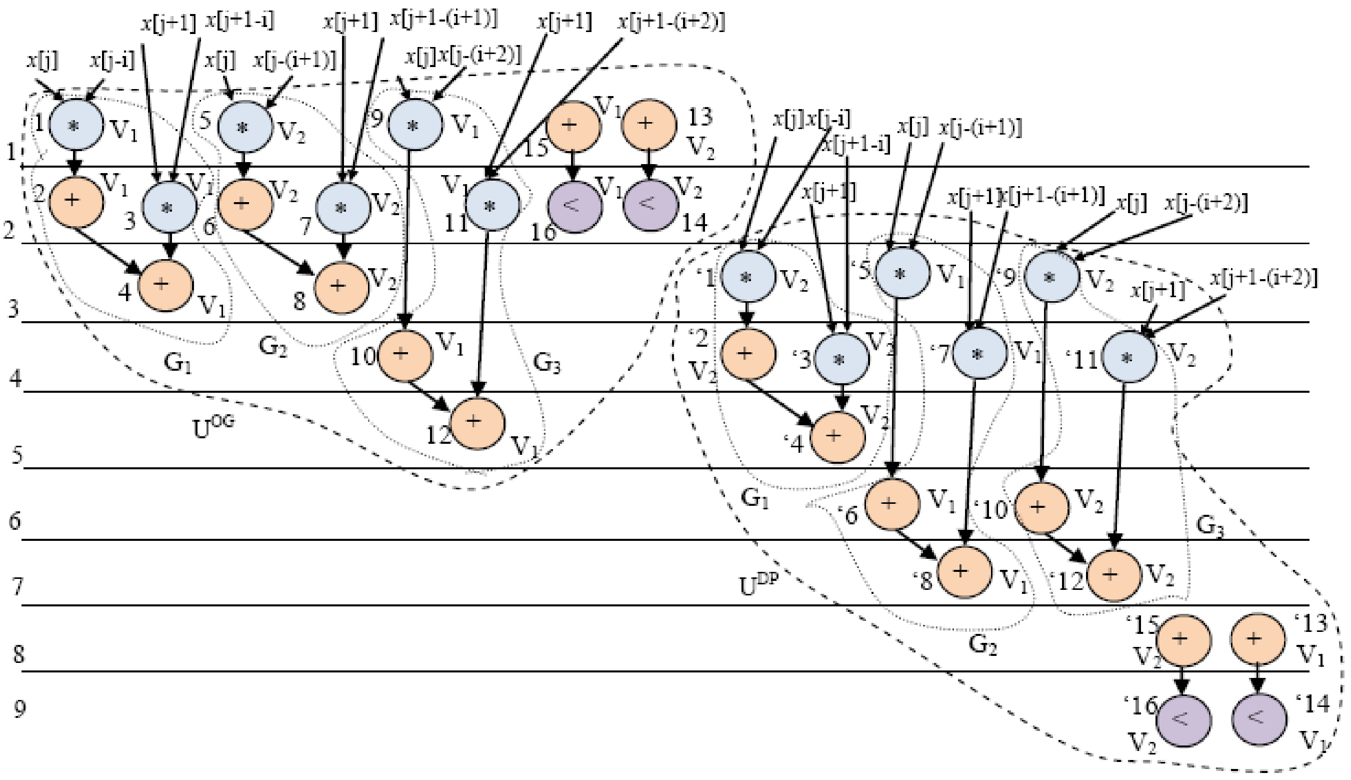

x[j]
x[j-i]
x[j+1]
x[j+1-i]
x[j-(i+1)]
x[j+1-(i+1)]
x[j-(i+2)]
x[j+1-(i+2)]
1
2
3
4
5
6
7
8
9
V1
V2
U^OG
U^DP
'1
'2
'3
'4
'5
'6
'7
'8
'9
'10
'11
'12
'13
'14
'15
'16

Figure 10


x[j]
x[j-i]
x[j+1]
x[j+1-i]
x[j-(i+1)]
x[j+1-(i+1)]
x[j+1-(i+2)]
x[j-(i+2)]
V1
V2
G1
G2
G3
U^OG
U^DP
1
2
3
4
5
6
7
8
9