SAND2020-3456R

Graph Theory and IC Component Design Analysis

James Obert*, Sean Turner, Jason Hamlet
Sandia National Labs
Albuquerque, NM, USA
{jobert, sdturne, jrhamle}@sandia.gov

Abstract— **Graph analysis in large integrated circuit (IC) designs is an essential tool for verifying design logic and timing via dynamic timing analysis (DTA). IC designs resemble graphs with each logic gate as a vertex and the conductive connections between gates as edges. Using DTA digital statistical correlations, graph condensation, and graph partitioning, it is possible to identify high-entropy component centers and paths within an IC design. Identification of high-entropy component centers (HECC) enables focused DTA, effectively lowering the computational complexity of DTA on large integrated circuit graphs. In this paper, a devised methodology termed IC layout subgraph component center identification (CCI) is used to identify described. CCI lowers DTA computationally complexity by condensing IC graphs into reduced subgraphs in which dominant logic functions are verified.**

Keywords— Graph Analysis, Dynamic Timing Analysis; IC Design Verification; IC Design Analysis.

James Obert, Sean Turner and Jason Hamlet are actively inolved in VLSI design analysis research at Sandia National Labs. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

NOTICE: This documenrt was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government, nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, make any warranty, express or implied, or assume any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represent that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof, or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof, or any of their contractors.

I. INTRODUCTION

A. Overview

A chip's timing constraints are tested at a set clock rate using dynamic timing analysis (DTA) [3]. In DTA a set of signals is input to the chip and functionality of the chip's design is verified and predicted output signals are observed. DTA's goal is to verify that an ASIC design can operate without errors at a specified clock rate and is generally accomplished using the simulated manufacturing design synthesis files [4,5]. DTA is used to test circuit logic and is especially useful in the analysis of asynchronous designs or when designs have clocks crossing into multiple domains.

The set of input and output signals associated with a DTA simulation form the basis of an IC layout graph. Logic gates within a layout equate to the vertices in the graph while the conductive connections between the logic gates represent the graph edges. The edges between each vertex are weighted according to the maximum statistical cross-correlation value observed in the signals traveling between the vertices. Cross-correlation is defined in Equation 1 below. The cross-correlation between feature vectors $V1_a$ and $V2_a$ is calculated. Consider $V1_i$ and $V2_i$ which are represented as two time-series, where a = 1,2,3...N and N equals the total number of samples in the series. The cross-correlation C at delay d is defined as [22].

$$C_d = \frac{\sum_{a=1}^{N}(V1_a - mean(R))(V2_{(a-d)} - mean(V2))}{\sqrt{\sum_{a=1}^{N}(V1_a - mean(V1))^2}\sqrt{\sum_{a=1}^{N}(V2_{(a-d)} - mean(V2))^2}} \quad (1)$$

Using Equation 1, the maximum cross-correlation vector $Kq = max(C_d)$ is found for all vertex combinations and an adjacency matrix is formed with the IC gates as the vertices and K_q as the edge weight values on the edges connecting the vertices. When analyzing IC graphs, those vertices that are strongly connected with edge weights Kq greater than a predetermined correlation value are of highest statistical interest. Figure 1 below shows a sample graph. If we are interested, for example, in graphs with all edge weights $Kq >$ 0.9, then the subgraph consisting of the vertices $\{s_1, s_2, s_6, s_4\}$ would be a graph of statistical interest when conducting DTA.

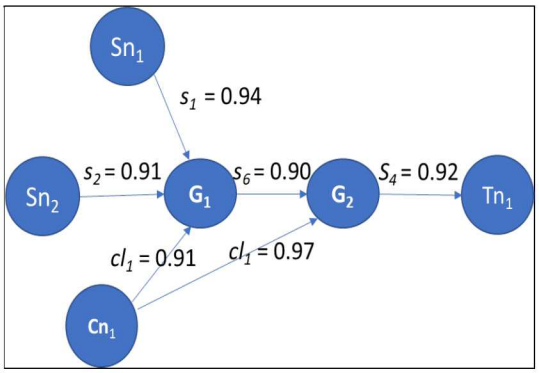


Figure 1: IC statistically significant subgraph.

The subgraph in Figure 1 represents a statistically significant subgraph within the much larger IC graph, and is a directed graph with time dependent edge weights of $\{s_1, s_2, s_6, s_4, cl_1\}$. The output signal s_4 depends on signals $\{s_1, s_2, s_6, cl_1\}$ and is effectively a directed path from output from vertex Tn_1 to source and clocking vertices $\{Sn_1, Sn_2, Cn_1\}$ through logic gates G_1 and G_1. During a DTA simulation, the statistically significant path logic and clocking signals $\{s_1, s_2, s_6, s_4, cl_1\}$ are verified against design timing and logic specifications. As Table 1 shows the adjacency matrix for those edges where $Kq > 0.9$, and where vertex self-loops are

	Sn_1	Sn_2	Cn_1	G_1	G_2	Tn_1
Sn_1	0	0	0	$Prob(s_1)$	0	0
Sn_2	0	0	0	$Prob(s_2)$	0	0
Cn_1	0	0	0	$Prob(cl_1)$	$Prob(cl_1)$	0
G_1	0	0	0	0	$Prob(s_6)$	0
G_2	0	0	0	0	0	$Prob(s_4)$
Tn_1	0	0	0	0	0	0

Table 1: Signal distribution adjacency matrix.

excluded. This process of IC layout subgraph component centers identification (CCI) affords a means for characterizing statistically significant logic components within an IC design. CCI is especially useful in large System-on-Chip (SoC) designs where the number of logic gates exceeds 1 million, and where serially analyzing each logic path or center would be resource intensive. The use of CCI in large layouts provides a means for identifying high-entropy component centers such that focused DTA of associated functions can be achieved using far fewer computing resources.

In this paper various graph analysis methods for implementing CCI are described. Section II describes CCI enabling algorithms including graph vertex importance, condensation, cycle counting, partitioning and associated time and class computing complexities. Section III discusses experimental results, and Section IV provides conclusions and discusses future work.

II. Large IC Design Graph Analysis

A. Condensed and Strongly Connected Graphs

The graph in Figure 1 is a simple graph with only 6 vertices; however, when the number of vertices is large, then graph condensing is an essential tool in identifying logic centers within a layout.

There are two main types of graph compression schemes. The first one discussed is based on *vertex importance* and the second one is based on *vertex similarity*.

Importance-based compression uses vertices weight functions based on degree and shortest paths in a graph. The importance-based weight functions come in a couple of varieties [40]. Global weight functions use a vertex's rank to calculate a weight and compare that weight to the weight of all others in a graph. Such weight functions tend to favor strongly connected vertices in dense graphs. Local weight functions consider a calculated vertex weight using vertex rank as well but compares a vertex's weight to only a local neighborhood of vertices. Compression is achieved in both types of importance-based methods by dropping lower weighted vertices and representing the graph using only the higher weight vertices. Additionally, there are importance-based compression functions which use shortest paths between vertices as the compression guide. Compression is achieved in shortest path schemes by representing the graph using only those vertices and edges having the shortest path between them.

Similarity-based compression compress merging vertices having some minimum number of common neighbors. As importance-based compression has proven to be an effective and accurate method for locating component centers, we have restricted the analysis in this paper to its use. The following sections describe several of the principal centrality methods used in performing importance-based graph compression.

B. Vertex Centrality

Vertex centrality methods are quite diverse depending on the nature of the graph data. The centrality of a vertex in a graph is its relative importance in the graph and can be used as the basis for importance-based graph compression. The more common methods include: neighborhood-based centrality, path-based centrality and iterative refinement centrality [41]. Algorithm for computing centrality are broken up into two categories: degree-based and geodesic. Degree-based algorithms use the degree of a vertex and the degree of its neighbors while geodesic algorithms use computation of shortest paths. For this reason, geodesic algorithms typically are limited by the efficiency of finding shortest paths.

C. Neighborhood-based Centrality

Neighborhood-based centrality roots from the concept that a node's influence is highly correlated to its capacity to impact the behaviors of its surrounding neighbors. The simplest rank-based algorithm counts the number of immediate neighbor connections for a *degree centrality*. centrality. The *ClusterRank* [50] algorithm is an improvement over simple immediate neighbor degree centrality as ClusterRank considers the number of immediate neighbors and a clustering coefficient [49] of a node. The clustering coefficient considers the interactions among the connected nodes. Nodes with a similar number of neighbors are distinguished by the clustering coefficient. Those nodes with smaller clustering coefficients have greater influence or importance than those that have larger clustering coefficients. When considering a node's neighborhood-based centrality, the location of a node also determines its importance.

Degree centrality counts the number of connections for each adjacent node while the ClusterRank centrality algorithm considers the number of nearest neighbors and the interactions between the nodes. In a directed graph with starting vertex v_i to destination vertex v_j , the ClusterRank score for a vertex v_i is defined in Equation 2 below [50],

$$CR_i = f(c_i)\sum_{j\in\tau_i}(k_j^o + 1) \tag{2}$$

where

$f(c_i)$: is a function of the clustering coefficient c_i of the node v_i in the directed graph G.
k_j^o: is the out-degree of v_i.
τ_i^o : is the set of nearest out-neighbors of v_i.

and

$$c_i = \frac{\left|\{(j \to k)|j,k \in \tau_i^o\}\right|}{k_j^o(k_j^o-1)} \tag{3}$$

Typical forms of $f(c_i)$ are ∂^{c_i}, where ∂ is optimally chosen base. Using ClusterRank, the potential information spreading centrality and influence of a node is determined by identifying nodes that connect to multiple communities outside of the local neighborhood. In addition, ClusterRank is useful in locating nodes in neighborhoods exhibiting structural holes. Those nodes near structural holes have the greatest chance of spreading unique data due because that node is less-likely to be deeply immersed in local neighborhood messaging. Nodes operating on the local neighborhood edge, act as bridges to other communities and linking non-redundant information from those communities.

Those nodes that are highly central within the graph, its importance is increased [45]. A process called *k-core decomposition* [46,47] finds the *residual degree* of the graph nodes using an iterative decomposition process. Node coreness measures how centrally located it is relative to the other nodes. In an undirected simple graph G, the coreness of each node designated c_i is the simple degree of the node. In k-core decomposition, iteratively each node is removed from the graph in ascending order. Firstly, all nodes with degree equal 0 are removed. Subsequently, all nodes with degree equal 1 are removed which changes the degree of all higher order nodes that were connected to those nodes. Those remaining higher order nodes now have a residual degree of $k > 1$, and all such nodes belong to the 1-shell. Next, all residual nodes with $k \leq 2$ are removed resulting in a residual set of nodes with degree $k > 2$ that belong to the 2-shell, and so forth. It should be noted that coreness is not a useful metric in tree-like or scale-free [48] graphs where the coreness of nodes are very similar due to the hierarchical or homogenously connected nature of the graphs respectively. Additionally, the coreness as stated thus far, considers only the residual node degree (k_i^r) and connectivity to nodes in the same k-shell or higher order k-shell nodes, and disregards the connections to exhausted node degree (k_i^e) that have been removed. Equation 4 below considers both the residual and exhausted node degree when calculating coreness [51] resulting in a more accurate centrality assignment,

$$k_i^{mix} = k_i^r + \alpha k_i^e \tag{4}$$

where α is a tunable parameter.

Unlike the highly iterative k-core decomposition method which depends on entire graph processing in determining coreness. The Hirsh index [52] or *H*-index is a local centrality method only requiring each node to know a limited amount of information such as the degree of its local neighbors. The operator H is defined as a finite number of real variable set $S = \{x_1, x_2, \ldots, x_m\}$ which yields a $max(h)$ integer value U where among the set S where there are at least h elements greater than or equal to h. The *H*-index of a node in a graph is expressed in Equation 5 as:

$$h_i = H(k_{j_1}, k_{j_2}, \ldots, k_{jk_i}) \tag{5}$$

where $k_{j_1}, k_{j_2}, \ldots, k_{jk_i}$ is the sequence of degree values of the neighbors of v_i. As an example, if a node h_i has local neighboring nodes with degrees (highest to lowest degree) $H(20, 17, 15, 6, 5, 3)$, then the H-index for $h_i = 5$ and is low complexity means for measuring node importance.

D. Path-based Centrality

Path-based centrality algorithms which are termed *geodesic*, calculate the shortest paths between all vertices in a graph. Those nodes exhibiting shortest distance d_{ij} paths between nodes v_i and v_j in a graph are considered more centric and of higher importance. Two of the least computationally complex path-based centrality algorithms are closeness centrality and betweenness centrality. Each of these algorithms are described below.

E. Closeness Centrality

The closeness centrality of node v_i is the inverse of the harmonic mean geodesic distances from v_i to all the other

nodes within the graph and is expressed in Equation 6 below with n equaling the number of nodes in the graph [53].

$$cc_i = \frac{1}{(n-1)} \sum_{j \neq i} \frac{1}{d_{ij}} \quad (6)$$

Larger cc_i values indicate the relative closeness of the node to the other nodes, and the average information transfer efficiency of a graph is expressed in Equation 7.

$$eff = \frac{1}{(n-1)} \sum_{i=1} \sum_{j=1, j \neq i} \frac{1}{d_{ij}} \quad (7)$$

F. Betweeness Centrality

In most cases there is more than one shortest path between source node v_s and destination node v_t that traverses node v_i. The composite shortest-path flow of information through v_i is computed by counting the number of shortest paths flowing through v_i. The normalized betweenness centrality (BC) of node v_i is given by Equation 8,

$$BC_i = \frac{2}{(n-1)(n-2)} \sum_{i \neq s, i \neq t, s \neq t} \frac{u_{st}^i}{u_{st}} \quad (8)$$

Where u_{st} is the number of shortest paths between v_s and v_t and u_{st}^i equals the number of shortest paths which traverse v_i between v_s and v_t.

Communicability BC [54] (CBC) is especially useful in the identification of component centers. CBC uses scaling to more heavily weight shorter paths versus longer paths between a source node v_s and a destination node v_t. For the subgraph G_{st} with adjacency matrix A,

$$G_{st} = (e^A)_{st} \quad (9)$$

and

$$CBC_i = \frac{2}{(n-1)(n-2)} \sum_{i \neq s, i \neq t, s \neq t} \frac{G_{st}^i}{G_{st}} \quad (10)$$

G. Iterative Refinement Centralities

In previously discussed centrality methods, the topology, connectivity and the paths between nodes were the key factors in determining a node's influence. Iterative refinement centralities concern the mutual enhancement effect [55] or influence of a nodes upon one another. The more useful and computationally efficient iterative refinement centrality methods for identifying component centers includes eigenvector centrality [56] and PageRank [57]. Each of these methods will be described below.

H. Eigenvector Centrality

In eigenvector centrality the centrality of a node is proportional to the sum of the connecting node centralities. In Equation 11 below the importance of node v_i is x_i.

$$x_i = c \sum_{j=1}^{n} a_{ij} x_j \quad (11)$$

Equation 12 is the matrix form of Equation 11 with $c = 1/\lambda$, where λ is the is the largest eigenvalue of A. Eigenvector centrality is generally computed by power iteration method in which the relative score of each node is shared with connected neighbor nodes as the scores update on each iteration. A steady state is eventually reached where the values of the nodes become fixed.

$$X = cAX \quad (12)$$

For directed graphs, Equation 13 includes the terms ∂ and $\vec{e}$ which are the relative importance parameter endogenous versus exogenous factors and the exogenous sources status respectively.

$$X = \partial AX + \vec{e} \quad (13)$$

I. PageRank

PageRank was originally developed to rank the importance of web pages according to the quantity and quality of the pages linked to it and has since been utilized in multiple domains. In PageRank, each node in a graph is given a page rank value and using an iterative process each node distributes its page rank value to its neighbor nodes. The page rank value of a node v_i is equal to PR_i at the iterative step m, and expressed mathematically as,

$$PR_i(m) = \sum_{j=1}^{n} a_{ij} \frac{PR_j(m-1)}{k_j^o} \quad (14)$$

where n is the number of nodes in the graph, and k_j^o equals the out degree of node v_j. Once a steady state is reached within the graph, the iterative process will stop.

Nodes present in a graph that have a zero out degree will prevent a steady state convergence to occur since its PR value cannot be distributed. Equation 15 below contains a random jumping factor of $€$ that represents the probability of a visit to node v_j and a probability of (1- $€$) of leaving the same node.

$$PR_i(m) = € \sum_{j=1}^{n} a_{ij} \frac{PR_j(m-1)}{k_j^o} + \frac{(1-€)}{n} \quad (15)$$

J. Algorithm Complexity

In Sections II.B-I a discussion of centrality methods used as the basis for importance-based graph compression. Primarily the computational complexity of the centrality method used in importance-based graph compression is a limiting factor when large IC graphs are analyzed. To gauge which centrality algorithms, give accurate centrality results while minimizing

the computation time and resources, neighborhood-based, path-based and iterative-refinement centrality methods are compared in the experimental results Section III. As a reference, the computational time complexity of those chosen centrality methods analyzed is listed in Table 2 below and includes: degree, PageRank, eigenvector, closeness and betweenness centrality with |V| and |E| being the sizes of the graph vertex and edge sets

Centrality Algorithm	Centrality Type	Time Complexity
Degree	neighborhood-based	$\mathcal{O}(\|V\|)$
PageRank	iterative-refinement	$\mathcal{O}(\|V\|+\|E\|)$
Eigenvector	iterative-refinement	$\mathcal{O}(\|V\|+\|E\|)$
Closeness	Path-based	$\mathcal{O}(\|V\|*(\|V\|+\|E\|))$
Betweeness	Path-based	$\mathcal{O}(\|V\|^3)$

Table 2: Centrality method computational time complexities.

From Table 2, it is evident that neighborhood-based and iterative-refinement centrality algorithms expressed in big O notation run in linear time while path-based algorithms run in quadratic and cubic time. It would stand to reason that in large graphs degree or iterative-refinement centrality algorithms should be used whenever possible to save computing time and resources. Depending on the graph structure, there are instances where path-based centrality algorithms will yield a more accurate node importance measurement as compared to neighborhood-based and iterative-refinement algorithms. Correlation of path-based, iterative-refinement and neighborhood-based graph vectors |V| is recommended during prototyping. In those instances where a high positive correlation exists between a path-based and either an iterative refinement or neighborhood-based algorithm for a graph, it is recommended that lower complexity iterative refinement or neighborhood-based algorithm be used for graphs of similar structure.

K. Component Center Anomalies

As illustrated in Figure 1, using Equation 1, high-entropy subgraphs are extracted based on the max cross-correlation values $max(C_d)$ observed between vertex signals transiting edges $\{s_1, s_2, s_6, s_4, cl_1\}$. Analyzing the relative entropy of such signals over time is used to identify anomalous behaviors within the subgraph. In Equation 15 below, individual signal distribution K-L Divergence values are calculated for baseline (no anomalies) and test (possible anomalies) event windows. In Equation 15, K-L Divergence measures the relative entropy change between s_q and s'_q distributions. Where S_q is a baseline signal probability distribution within a baseline event window and $S_q{}'$ is a target or test probability distribution within the test event window.

$$D_{KLS}(S_q{}'||S_q) = \sum_{s_q, s'_q} Prob(s'_q) \ln \frac{Prob(s'_q)}{Prob(s_q)} \qquad (15)$$

If D_{KLS} exceeds a heuristically derived threshold ($D_{KLS} > T$), the feature s'_q is deemed anomalous. In the results Section III.B, an anomaly detection example of component center signal anomalies is illustrated.

L. Identification of High-entropy Component Centers

Figure 2 below, outlines the processes involved in the identification of high-entropy component centers correlation (HELC**)** followed by the identification of anomalies within component centers**.**

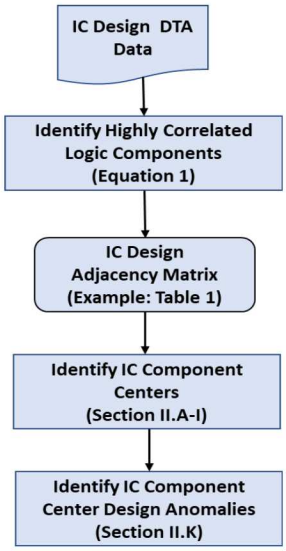


Figure 2: Identification of IC design anomalies.

Step 1 of the HELC process begins with the ingestion of DTA session data from a VLSI design. The DTA data itself is a set of time-series binary pulses transmitted between gates within the VLSI design. These pulses are sampled at 20 times above the maximum signal frequency exhibited within the DTA data. Using Equation 1, the sampled data is formed into an adjacency matrix like that shown in Table 1. The IC component centers are then identified using the methods discussed in Section II.A-I. The final step includes identifying the component center anomalies which themselves may indicate the presence of unintended design deviations. Section III which follows discusses the results of analysis on an experimental DTA dataset.

III. EXPERIMENTAL RESULTS

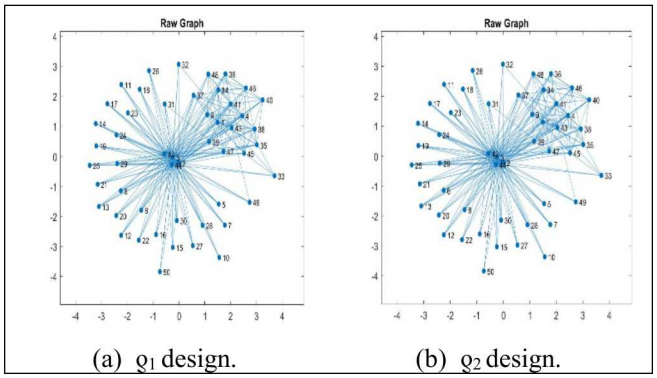
A. Experimental Data

To illustrate the utility of the methodology described in Section II, two IC designs were generated. A baseline (ϱ_1) IC design (no design flaws) and a modified version of the baseline IC design (ϱ_2) which contains design deviations were generated. Two DTA session were run on both ϱ_1 and ϱ_2 designs for 1000 clock cycles such that a 270x1000 sample

window of data was extracted and subsequently converted via HELC to a 50x50 adjacency matrix with 270 edges.

B. Identification of Component Centers

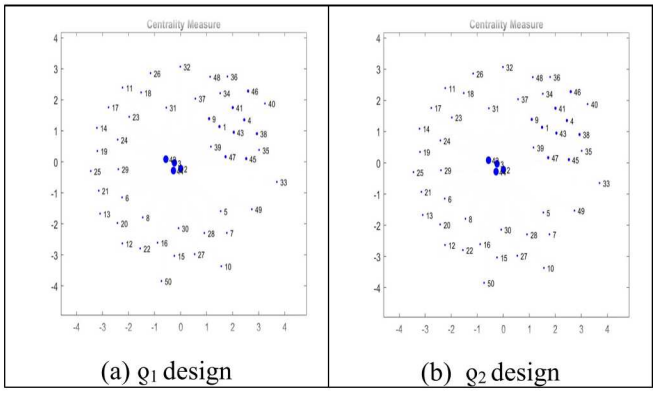
Graphs representing ϱ_1 and ϱ_2 designs are shown in Figure 3 below.



(a) ϱ_1 design. (b) ϱ_2 design.

Figure 3: Design ϱ_1 and ϱ_2 graphs.

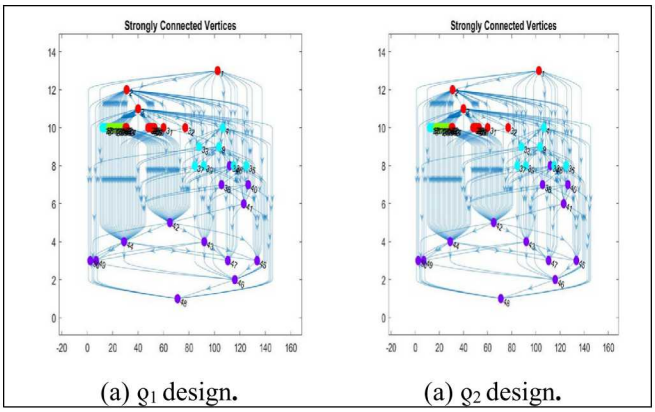
As explained in Section II, vertex centrality is a useful metric for gauging the importance of a vertex within a graph. After comparing the path-based, degree-based, and iterative refinement centralities of designs ϱ_1 and ϱ_2, it was determined using the correlation method in Section II.J that the eigenvector centrality method had the lowest computational complexity while positively correlating with path-based centrality algorithms. Figure 4 below shows the results of finding eigenvector centrality for designs ϱ_1 and ϱ_2. In Figure 4 the highlighted node in the center of the plot have the highest centrality and importance.



(a) ϱ_1 design (b) ϱ_2 design

Figure 4: Eigenvector centrality of ϱ_2 design.

Figure 5 based on eigenvector centrality shows groups of strongly connected node groups. Node groups or *graph components* in Figure 5 are similarly colored neighbor nodes and represent the distinct component centers in the ϱ_1 and ϱ_2 designs. Graph compression is achieved by considering each graph component or component center as a single node in the graph.



(a) ϱ_1 design. (a) ϱ_2 design.

Figure 5: Design ϱ_1 and ϱ_2 strongly connected graphs.

C. Component Center Deviations

Looking at Figures 3-5, from a purely visual standpoint, there is no obvious connectivity differences between design ϱ_1 and ϱ_2 nodes. This is because the primary difference between the two designs lies in the logic signal transition distributions associated with the graph edges in each design. To quantitatively analyze the component centers identified in Figure 5 above, it is necessary to find the differences between the logic transition distributions associated with equivalent edges within equivalent component centers of each design using Equation 15.

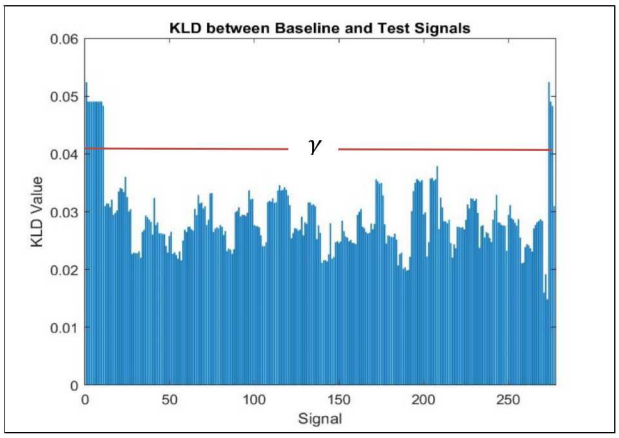


Figure 6: KLD values for ϱ_1 and ϱ_2 component center edge distributions.

Component center deviation detection is quantified using a threshold level γ which is set according to statistical population analysis. The threshold γ is set by finding the mean (m) KLD level for the signals in Figure 6. The detection sensitivity is set using a sensitivity factor ℓ in equation 16 below. A high sensitivity level corresponds to a lower γ value as calculated by Equation 16:

$$\gamma = m + \ell(std) \tag{16}$$

where ℓ is the sensitivity factor and *std* is one standard deviation away from m. A low sensitivity factor translates to a larger γ while a high sensitivity factor produces a smaller γ. A larger γ (low sensitivity) with a threshold line appearing higher vertically in Figure 6, results in lower false positive and higher false negative detection rates. A smaller γ (high sensitivity) results in lower detection false negative and higher false positive rates.

IV. CONCLUSIONS AND FUTURE WORK

In this paper is was shown that graph analysis in IC designs is possible using efficient graph analysis techniques with DTA datasets. Using cross-correlation with DTA data, statistically significant IC logic gate connections are identified from which graph vertices and edges are derived. Using the least computationally complex graph centrality algorithm, graph components consisting of strongly connected node neighborhoods are identified. Such strongly connected node neighborhoods represent IC component centers in which relative entropy measurements are used to efficiently verify component center design integrity.

Although the DTA dataset analyzed in this paper was not on the scale of a VLSI million gate plus design, it is fully expected that the methodology and algorithms discussed can efficiently be scaled to such designs using conventional parallel processing and more advanced computational platforms. Future research will include algorithms research, design and performance analysis of VLSI design verification techniques. When considering complex graphs, the use of classical parallel computing platforms such as GPU clusters typically provide computational time complexity speedups of between 10-30X over typical CPU systems. Greater performance gains are theoretically possible using hybrid quantum-classical multilevel combinatorial optimization frameworks [59], and as applied to VLSI design verification will be an area of future research.

REFERENCES

[1] A. Alnasser, H. Sun, “A Fuzzy Logic Trust Model for Secure Routing in Smart Grid Networks”, IEEE Access, September 2017.

[2] M. Usman, V. Muthukkumarasamy, and X.-W. Wu, ``Mobile agent-based cross-layer anomaly detection in smart home sensor networks using fuzzy logic," *IEEE Trans. Consum. Electron.*, vol. 61, no. 2, pp. 197_205, Feb. 2015.

[3] S. Renubala and K. Dhanalakshmi, ``Trust based secure routing protocol using fuzzy logic in wireless sensor networks," in *Proc. Int. Conf. Comput. Intel. Comput. Res. (ICCIC)*, 2014, pp. 1_5.

[4] W. Meng, H. Zia, H. Song, “A Dynamic Trust Model Based on Recommendation Credibility in Grid Domain”, July 2010.

[5] N. Marchang and R. Datta, ``Light-weight trust-based routing protocol
for mobile ad hoc networks," *IET Inf. Security*, vol. 6, no. 2, pp. 77_83, Jun. 2012.

[6] M. Xiang,W. Liu, and Q. Bai, ``Trust-based geographical routing for smart grid communication networks," in Proc. 3rd IEEE Int. Conf. Smart Grid *Commun.*, Nov. 2012, pp. 704_709.

[7] R. Moghaddass, M. J. Wang, “A Hierarchical Framework for Smart Grid Anomaly Detection Using Large-Scale Smart Meter Data”, IEEE Transactions on Smart Grid, April 2017.

[8] S. Mohagheghi, J. Stoupis, Z. Wang, Member, “Communications Protocols and Networks for Power Systems – Current Status and Future Trends”, IEEE Explore, April 2009.

[9] Network Working Group of the IETF, January 2006, RFC 4251, The Secure Shell (SSH) Protocol Architecture.

[10] Stebila, D.; Green J. (December 2009). ″RFC5656 - Elliptic Curve Algorithm Integration in the Secure Shell Transport Layer"., 12 November 2012.

[11] Seggelmann, R.; Tuxen, M.; Rathgeb, E.P. (18–20 July 2012). "SSH over SCTP — Optimizing a multi-channel protocol by adapting it to SCTP". Communication Systems, Networks & Digital Signal Processing (CSNDSP), 2012 8th International Symposium on: 1–6.

[12] M. Guri, M. Monitz, Y. Mirski and Y. Elovici, "BitWhisper: Covert Signaling Channel between Air-Gapped Computers Using Thermal Manipulations," *2015 IEEE 28th Computer Security Foundations Symposium*, Verona, 2015, pp. 276-289.

[13] Kirat, Dhilung; Vigna, Giovanni; Kruegel, Christopher (2014). Barecloud: bare-metal analysis-based evasive malware detection. ACM. pp. 287–301. ISBN 978-1-931971-15-7.

[14] Bia, A. (2013). Data Management and Security: Applications in Medicine, Sciences and Engineering. WIT Press. ISBN 9781845647087.

[15] Johansen, Håvard; Johansen, Dag; Renesse, Robbert van (2007-05-14). Venter, Hein; Eloff, Mariki; Labuschagne, Les; Eloff, Jan; Solms, Rossouw von, eds. New Approaches for Security, Privacy and Trust in Complex Environments. IFIP International Federation for Information Processing. Springer US. pp. 373–384.

[16] Bank, David (August 17, 2005). *"Spear Phishing Tests Educate People About Online Scams"*. The Wall Street Journal.

[17] Basil Cupa, Trojan Horse Resurrected: On the Legality of the Use of Government Spyware (Govware), LISS 2013, pp. 419–428.

[18] Global Supply Chain Security, James Giermanski, Scarecrow Press, 2012 - 218 Pages.

[19] Cowley, John (2007). Communications and Networking: An Introduction. Springer. ISBN 9781846286452.

[20] Hillebrand, Friedhelm, ed. (December 2001). GSM and UMTS, The Creation of Global Mobile Communications. John Wiley & Sons. ISBN 978-0-470-84322-2.

[21] Chai Keong Toh Ad Hoc Mobile Wireless Networks, Prentice Hall Publishers, 2002. ISBN 978-0-13-007817-9.

[22] Kozierok, Charles M. (1 January 2005). "The TCP/IP Guide: A Comprehensive, Illustrated Internet Protocols Reference". No Starch Press. ISBN 9781593270476.

[23] Jolliffe I.T. *Principal Component Analysis*, Series: Springer Series in Statistics, 2nd ed., Springer, NY, 2002, XXIX, 487 p. 28 illus.

[24] Daniel Jurafsky and James H. Martin (2008). *Speech and Language Processing*, 2nd edition. Pearson Prentice Hall.

[25] Christopher D. Manning and Hinrich Schütze (1999). *Foundations of Statistical Natural Language Processing*. The MIT Press.

[26] *Chapelle, Olivier; Schölkopf, Bernhard; Zien, Alexander (2006). Semi-supervised learning. Cambridge, Mass.: MIT Press.*

[27] Haykin, Simon, "Neural Network. A comprehensive foundation." Neural Networks 2.2004 (2004).

[28] Barlow, Horace B. "Unsupervised learning." Neural computation 1.3 (1989): 295-311.

[29] S. Kullback, R. A. Leibler, “On Information and Sufficiency”, George Washington University, 1956.

[30] Breiman, L., J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Boca Raton, FL: CRC Press, 1984.

[31] Caudill, M., and C. Butler, *Understanding Neural Networks: Computer Explorations*, *Vols. 1 and 2*, Cambridge, MA: The MIT Press, 1992.

[32] Battiti, R., "First and second order methods for learning: Between steepest descent and Newton's method," *Neural Computation*, Vol. 4, No. 2, 1992, pp. 141–166.

[33] C. Cummings, Z. Wang, H. Leather, "End-to-end Deep Learning of Optimization Heuristics", 2017 26th International Conference on Parallel Architectures and Compilation Techniques, September 2017.

[34] J. Patterson, A. Gibson, "Deep Learning a Practitioner's Approach", O'reilly, January 2017.

[35] M. Abeydeera, S. Subramanian, "SAM: Optimizing Multithreaded Cores for Speculative Parallelism", 2017 26th International Conference on Parallel Architectures and Compilation Techniques, September 2017.

[36] Conrad D. James, James B. Aimone, "A historical survey of algorithms and hardware architectures for neural-inspired and neuromorphic computing applications", Biologically Inspired Cognitive Architectures, July 2016.

[37] A. Wichert, "Principles of Quantum Artificial Intelligence" World Scientific Publishing Co. ISBN 978-981-4566-74-2, 2014.

[38] D. Chatterjee; Arijit Roy, "A transmon-based quantum half-adder scheme". Progress of Theoretical and Experimental Physics. **2015** (9): 093A02(16pages), 2015.

[39] J. Obert; A. Chavez, J. Johnson, " Behavioral Based Trust Metrics and the Smart Grid", IEEE Trustcom 2018, July 2018.

[40] T. Suel and J. Yuan. Compressing the graph structure of the web. In Proceedings of the IEEE Data Compression Conference, pages 213–222, 2001.

[41] LinyuanLüb,a,∗,DuanbingChena,c,Xiao-LongRend,Qian-MingZhangc, Yi-ChengZhanga,e,TaoZhouc, "Vital nodes identification in complex networks", Physics Reports, 2016.

[42] McAfee Labs Threat Report 2018, McAfee Labs, 2018.

[43] Rolling Review Kickoff "Network Behavior Analysis Systems", 2008.

[44] S. Karnouskos: *Stuxnet Worm Impact on Industrial Cyber-Physical System Security*. In:*37th Annual Conference of the IEEE Industrial Electronics Society (IECON 2011), Melbourne, Australia*, 7-10 Nov 2011. Retrieved 20 Apr 2014.

[45] M. Kitsak, L.K. Gallos, S. Havlin, F. Liljeros, L. Muchnik, H.E. Stanley, H.A. Makse, Identification of influential spreaders in complex networks, Nat. Phys. 6 (11) (2010) 888–893.

[46] S.N. Dorogovtsev, A.V. Goltsev, J.F.F. Mendes, K-core organization of complex networks, Phys. Rev. Lett. 96 (4) (2006) 40601.

[47] S. Carmi, S. Havlin, S. Kirkpatrick, Y. Shavitt, E. Shir, A model of internet topology using k-shell decomposition, Proc. Natl. Acad. Sci. USA 104 (27) (2007) 11150–11154.

[48] A.-L. Barabási, R. Albert, Emergence of scaling in random networks, Science 286 (5439) (1999) 509–512.

[49] D.J. Watts, S.H. Strogatz, Collective dynamics of 'small-world' networks, Nature 393 (6684) (1998) 440–442.

[50] D.-B. Chen, H. Gao, L. Lü, T. Zhou, Identifying influential nodes in large-scale directed networks: the role of clustering, PLoS ONE 8 (2013) e77455.

[51] Y. Liu, M. Tang, T. Zhou, Y. Do, Core-like groups result in invalidation of identifying super-spreader by k-shell decomposition, Sci. Rep. 5 (2015) 9602.

[52] J.E. Hirsch, An index to quantify an individual's scientific research output, Proc. Natl. Acad. Sci. USA 102 (46) (2005) 16569–16572.

[53] G. Sabidussi, The centrality index of a graph, Psychometrika 31 (4) (1966) 581–603.

[54] E. Estrada, D.J. Higham, N. Hatano, Communicability betweenness in complex networks, Physica A 388 (5) (2009) 764–774.

[55] G.M. Wittenbaum, A.P. Hubbell, C. Zuckerman, Mutual enhancement: toward an understanding of the collective preference for shared information, J. Pers. Soc. Psychol. 77 (5) (1999) 967.

[56] P. Bonacich, Factoring and weighting approaches to status scores and clique identification, J. Math. Sociol. 2 (1) (1972) 113–120.

[57] S. Brin, L. Page, The anatomy of a large-scale hypertextual web search engine, Comput. Netw. ISDN Syst. 30 (1) (1998) 107–117.

[58] H. Hotelling, Simplified calculation of principal components, Psychometrika 1 (1) (1936) 27–35.

[59] H. Ushijima-Mwesigwa, R. Shaydulin, C. Negre, S. Mniszewski, Y. Alexeev, I. Safro, "Multilevel Combinatorial Optimization Across Quantum Architectures", ACM,