STRIPE: Signal Selection for Runtime Power Estimation

James J. Davis, Joshua M. Levine, Edward A. Stott, Eddie Hung,
Peter Y. K. Cheung and George A. Constantinides
Department of Electrical and Electronic Engineering
Imperial College London, London, SW7 2AZ, United Kingdom
E-mail: {james.davis06, josh.levine05, ed.stott, e.hung, p.cheung, g.constantinides}@imperial.ac.uk

***Abstract*—Knowledge of power consumption at a subsystem level can facilitate adaptive energy-saving techniques such as power gating, runtime task mapping and dynamic voltage and/or frequency scaling. While we have the ability to attribute power to an arbitrary hardware system's modules in real time, the selection of the particular signals to monitor for the purpose of power estimation within any given module has yet to be treated as a primary concern. In this paper, we show how the automatic analysis of circuit structure and behaviour inferred through vectored simulation can be used to produce high-quality rankings of signals' importance, with the resulting selections able to achieve lower power estimation error than those of prior work coupled with decreases in area, power and modelling complexity. In particular, by monitoring just eight signals per module ($\sim$0.3% of the total) across the 15 we examined, we demonstrate how to achieve runtime module-level estimation errors 1.5–6.9$\times$ lower than when reliant on the signal selections made in accordance with a more straightforward, previously published metric.**

I. Introduction

The power behaviour of bus-based, modular hardware systems, including those implemented on FPGAs, at subsystem granularities is of ever-increasing concern as user expectations for simultaneous performance and energy efficiency improvements rise. Information about such behaviour can be used to inform runtime decision-making, allowing, for example, tasks to be power-efficiently mapped to the hardware upon which they execute. In our previous work [1], we showed how module-level power breakdowns could facilitate power savings of up to 8%. Otherwise-identical modules within a system may, due to their design, variation at commissioning or degradation thereafter, behave differently at runtime; always assuming worst-case conditions leads to suboptimal performance and efficiency. Since the facilitation of separate power islands for module-level power measurement is usually impractical, a proxy—in particular, switching activity—must be used to estimate modules' power contributions via a model.

Given fixed overhead budgets, we are faced with the problem of selecting which signals to monitor in order to maximise the quality of a system power breakdown. Since, as shown by our results, monitoring overheads are proportional to the number of signals selected and, in the absence of model overfitting, the quality of the power estimate will improve monotonically with each additional signal monitored, we can cast this challenge within an optimisation setting: select the N signals likely to provide the best-quality power estimate, for any N, for each of the modules a system is composed of.

Figure 1 contrasts results obtained for this paper with those obtained in our prior work [1], the state-of-the-art power estimation framework from which we use for instrumentation

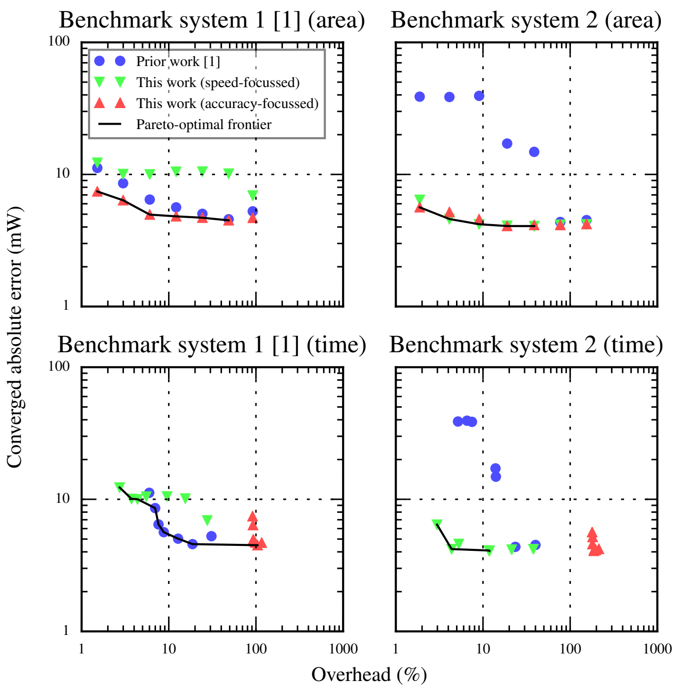

Fig. 1. Scatter plots of area and compilation time overheads vs achieved power estimation error for two benchmark systems using signal selection methods from our prior work [1] and those herein. The proposed selection methods are described in Sections III-A and III-B, the latter with $T = 1000$, while the benchmark systems are those in Sections V-A and V-B. The experiments performed are described in Section VI. Overhead results are normalised to those for the equivalent system lacking runtime power estimation capability.

and modelling. The plots' frontiers highlight that the two novel signal selection techniques we propose can achieve greater accuracy for lower overheads than when relying upon the more simplistic equivalent currently found in the literature. Both show generality while the speed (Section III-A)- and accuracy (III-B)-focussed methods demonstrate their respective superiorities over that used in our previous work.

The automatic signal selection for runtime power estimation (or *STRIPE*) of arbitrary hardware systems is a subject that has yet to be comprehensively studied. We present its first exploration in this paper, making the following novel contributions:

- We propose two new signal selection methodologies based on the automated analysis of modules' structural and statistical properties at compilation time, the former based on a fast graph centrality-computing algorithm

(PageRank) and the latter on a linear independence-revealing technique (QR decomposition).

- We describe the tools written to realise those methods, which have been incorporated within a CAD flow capable of instrumenting arbitrary hardware systems.
- We benchmark them on 15 modules assembled across two multi-module systems, comparing the achieved power estimation accuracies and incurred overheads and discussing their applicability and generality.

II. Background & Related Work

Compile-time estimation of power consumption is commonplace. Tools to predict hardware system power initially did so via cycle-accurate *vectored* circuit simulation, later using faster, probabilistic *vectorless* techniques [2]; the latter have also been used as input to an FPGA tool flow, facilitating power-aware compilation [3]. While neither vectored nor vectorless simulations necessitate the supply of representative input stimuli, their accuracies can generally be improved through their provision. Compile-time power estimation for modular systems has also been explored [4]. Macro modelling has shown promise in automatically identifying power modes [5], however this approach, as with all offline techniques, is unable to compensate for changes in exogenous conditions online.

While useful for ascertaining whether designs will meet their power requirements or to compare different implementations without performing placement and routing, runtime power estimation requires online monitoring of circuit behaviour. Zick et al. proposed the use of distributed sensors independent from application circuitry to monitor phenomena including power [6] while Najem et al. augmented designs with activity counters instead, using offline modelling to translate those counts to power estimates [7]. These works are unsuitable for module-level power estimation, however; system components rarely map separately enough to infer their behaviour from indirect measurements, such as the former's [6], and the latter [7] produces only system-wide estimates. Furthermore, reliance on offline models prevents adaptation to untrained-for effects. We sought to address these issues with *KAPow* [1] by using both direct measurement of switching activity and online modelling to estimate module-level power; Section II-A details its principles since we rely on them to evaluate STRIPE.

While the automatic signal selection for arbitrary hardware has not yet been explored within a power-monitoring context, it is an established method in other VLSI areas. For FPGAs, the most prominent is that of debugging; Hung et al. proposed several selection algorithms considering circuit structure, demonstrating feasibility for large designs using a graph centrality-based technique [8]. In the world of microprocessors, the attachment of appropriate events (control signals)—only a few of which can typically be captured simultaneously [9]—to counters is a related, and widely studied, problem. Recent work introduced *PowMon*, a CPU power estimation tool that strives to reduce the multicollinearity of selected events and was reported to achieve errors of 3–4% [9].

A. KAPow: Module-level Power Estimation

KAPow [1], which we use as the framework for our investigation, allows arbitrary FPGA-implemented hardware systems,

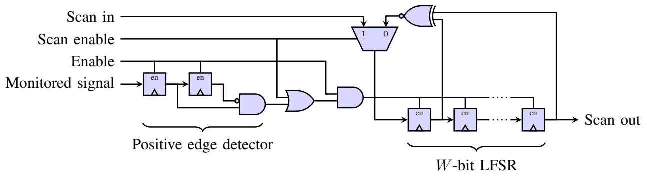

Fig. 2. KAPow activity counter [1]. If Enable $= 1$ and Scan enable $= 0$, the LFSR advances by one state per positive edge of the signal-under-monitoring. When, instead, Enable $= 1$ and Scan enable $= 1$, the shift registers of neighbouring counters are cascaded, allowing read-back via a scan chain.

logically split into M RTL modules and assembled with Altera's system integration tool, QSys, to be automatically and transparently instrumented in order to facilitate realtime activity monitoring. System partitioning is under user control; the granularity at which power estimates are desired dictates the module boundaries. Each module's most frequently toggling signals according to compile-time activity predictions, produced as detailed in Section II-A1, are instrumented as explained in Section II-A2. An adaptive online model, discussed in Section II-A3, runs in software in parallel with the application to combine these measurements with directly measured system-level power consumption in order to estimate the module-level contributions to the total power.

1) Signal Selection: KAPow's default signal selection relies on vectorless activity estimation, distinct from its vectored counterpart since switching rates are determined for all signals probabilistically by propagating fixed rates at primary inputs through to all others via evaluation of their Boolean relationships, making it fast. Neither user-supplied input stimuli nor testbenches are required. Altera's power analysis tool, PowerPlay, is used to produce estimates of signal-wise toggle rates for each module as switching activity files; these are then parsed, ordered and the top N—a user-specified parameter—highest-ranked signals per module are selected.

2) Instrumentation: Selected signals are augmented with fast, lightweight activity counters, shown in Figure 2, by manipulating Verilog Quartus Mapping (VQM) files—fully flattened technology-mapped netlists extracted post-placement and -routing—generated from each module's RTL. A W-bit linear-feedback shift register (LFSR) is used as the basis for each counter, where W is a further parameter specified by the user. Each module's counters are connected to form a single-bit scan chain, the head of which is grounded to reset the LFSRs during read-back; a controller embedded within each module facilitates the latter. The modified but functionally equivalent VQMs are then substituted for the original RTL.

3) Online Modelling: A linear relationship between signal activities $\boldsymbol{a}$ and measured system power y, $y = \boldsymbol{a}^{\mathrm{T}}\boldsymbol{x}$, is assumed; coefficient vector $\boldsymbol{x}$ contains scaling factors that determine the relative power contributions of each signal. KAPow uses a weighted linear model, fed by $\boldsymbol{a}$ and containing an estimate of $\boldsymbol{x}$, $\hat{\boldsymbol{x}}$, to compute an estimated system power $\hat{y}$. The identification setup used is shown diagrammatically in Figure 3. $\hat{\boldsymbol{x}}$ is determined and tuned online to minimise e via recursive least-squares (RLS) filtering; there is no dedicated training phase and the model is able to adapt to changes in workload and environmental conditions automatically.

Module-level power $\hat{y}_m$ can be estimated for each module

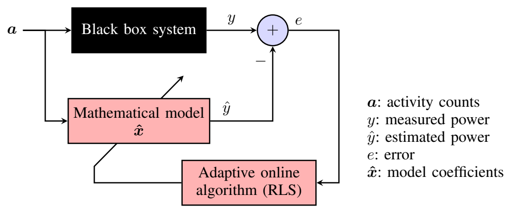

Fig. 3. System identification arrangement used by KAPow [1].

m as shown in (1), (2) and (3). Scalars a_s and $\hat{x}_\text{s}$ represent the 'activity' and associated coefficient of the system static power; a_s is set to a non-zero constant, allowing $\hat{x}_\text{s}$ to be tuned. The model is able to produce meaningful estimates for $\hat{\boldsymbol{x}}$ after $MN+1$, the model's order, updates of its elements.

$$\boldsymbol{a} = \begin{bmatrix} a_\text{s} \; \boldsymbol{a}_0^\text{T} \; \boldsymbol{a}_1^\text{T} \; \ldots \; \boldsymbol{a}_{M-1}^\text{T} \end{bmatrix}^\text{T} \tag{1}$$

$$\hat{\boldsymbol{x}} = \begin{bmatrix} \hat{x}_\text{s} \; \hat{\boldsymbol{x}}_0^\text{T} \; \hat{\boldsymbol{x}}_1^\text{T} \; \ldots \; \hat{\boldsymbol{x}}_{M-1}^\text{T} \end{bmatrix}^\text{T} \tag{2}$$

$$\hat{y}_m = \boldsymbol{a}_m^\text{T} \hat{\boldsymbol{x}}_m \tag{3}$$

III. Signal Selection Methodologies

When determining which signals to monitor, three primary factors should be considered:

1) Signals with high switching activity are more likely to be indicative of power consumption than those without.
2) Those driving high-capacitance (long and/or high-fanout) wires are also likely to be useful indicators.
3) Signal activities will be correlated, particularly within pipelines; monitoring (time-delayed) copies of existing signals will provide limited additional information.

Since KAPow's default signal selection, previously detailed in Section II-A1 and which we use for baseline comparison, dealt only with the first of these, in this section we propose two alternative, novel selection methods that address the remaining factors. Each produces lists of a module's signals whose order is determined by the particular methodology; a resultant selection is therefore the N highest-ranked entries.

A. Centrality

Centrality provides numerical measures of the influences of vertices (in our case, signals) V through edges (signal-to-signal relationships inferred through gates, flip-flops, etc.) E within a directed graph (module) G. While many metrics exist, in this paper we focus on *eigenvector* centrality [10] [11] since it considers the connectedness of a graph as a whole by exploiting the observation that vertices with central neighbours are themselves likely to be central, thereby providing more information than those that only consider vertices in isolation such as degree centrality (fanin or fanout). Consideration of circuit structure tackles the second of the above factors.

Eigenvector centrality c_eigen is defined for each vertex v as a positive multiple of the sum of adjacent vertices' centralities, as shown in (4). Adjacency is captured by each a_{vj}: this is set to 1 if an edge exists between v and target vertex j, otherwise it is 0. If all $c_\text{eigen}[v]$ are arranged in vector form as $\boldsymbol{c}_\text{eigen} = \begin{bmatrix} c_\text{eigen}[0] \; c_\text{eigen}[1] \; \cdots \; c_\text{eigen}[|V|-1] \end{bmatrix}^\text{T}$, their calculation requires only the eigendecomposition of matrix $\boldsymbol{A}^{|V|\times|V|}$, formed from all a_{vj}, resulting in the largest-magnitude eigenvalue. The problem to be solved is shown in (5). We are interested in 'right' eigenvector centrality, which considers vertices' out-edges (fanouts) rather than in-edges.

$$c_\text{eigen}[v] = \frac{1}{\lambda} \sum_{j \neq v \in V} a_{vj} c_\text{eigen}[j] \quad \forall v \in V \tag{4}$$

$$\lambda \boldsymbol{c}_\text{eigen} = \boldsymbol{A} \boldsymbol{c}_\text{eigen} \tag{5}$$

Unfortunately, 'vanilla' eigenvector centrality suffers from convergence problems with directed graphs that are not strongly connected; in particular, parents of strongly connected vertices are assigned zero centrality unless they are themselves children of central vertices. Katz centrality [12] [13] overcomes this by transforming (5) into (6); an additional constant β is added to assign all vertices a minimum centrality regardless of their connectedness. Here, $\boldsymbol{\beta} = \beta \mathbf{1}^{|V|}$.

$$\boldsymbol{c}_\text{Katz} = \alpha \boldsymbol{A} \boldsymbol{c}_\text{Katz} + \boldsymbol{\beta} \tag{6}$$

PageRank [14], the original Google 'importance'-ranking algorithm, is essentially a further refinement; its definition is given in (7). The addition here is $\boldsymbol{B}^{-1}$: a diagonal matrix in which the j^th element is $1/\sum_{i \neq j \in V} a_{ij}$, i.e. the number of in-edges to vertex j. Most attractively, $\boldsymbol{B}^{-1}\boldsymbol{A}$ has a large eigengap (difference between its highest- and second highest-magnitude eigenvalues) by virtue of the scaling effect of $\boldsymbol{B}^{-1}$, thus $\boldsymbol{c}_\text{PageRank}$ can be approximated to a high degree of accuracy by the power iteration method for large graphs in relatively few steps; computation time scales approximately with $\log(|V|)$. PageRank is therefore the algorithm we adopt for the computation of centrality in this paper. Note that we treat graphs as unweighted since we do not have direct access to capacitance data and that normalisation is not a concern since we are only interested in vertices' relative importance.

$$\boldsymbol{c}_\text{PageRank} = \alpha \boldsymbol{B}^{-1} \boldsymbol{A} \boldsymbol{c}_\text{PageRank} + \boldsymbol{\beta} \tag{7}$$

B. QR Decomposition

Once a particular signal has been selected, limited additional information can be gained by monitoring other nets with correlated switching activity since a portion of that data will already be captured by the existing signal's monitor. At first glance, it may appear that standard statistical methods such as principal component analysis (PCA) might offer an optimal solution to the problem of decorrelation; indeed, PCA could be applied such that sets of monitors capturing the same data with zero correlation were produced. The resulting principal components would, however, infer the creation of monitors each consisting of linear combinations of all nets, which would clearly be infeasible to implement. Instead, our aim must be to select a *maximally linearly independent* subset of N signals to monitor for a given module, thereby addressing the third factor in the aforementioned list.

Consider a matrix $\boldsymbol{X}$ in which each column corresponds to a particular signal, of which there are S, and each row is a time step (clock cycle), of which we have observed T. Let x_{ts} be 1 if the state of signal s changed (toggled) between time steps $t-1$ and t, otherwise let it be 0. The *subset selection* problem [15] consists of determining a permutation (column-reordering) matrix $\boldsymbol{P}$ such that $\boldsymbol{X}\boldsymbol{P} = \begin{bmatrix} \boldsymbol{X}_1^{T\times N} \; \boldsymbol{X}_2^{T\times(S-N)} \end{bmatrix}$, with $\boldsymbol{X}_1$ consisting of N columns of signals to monitor while

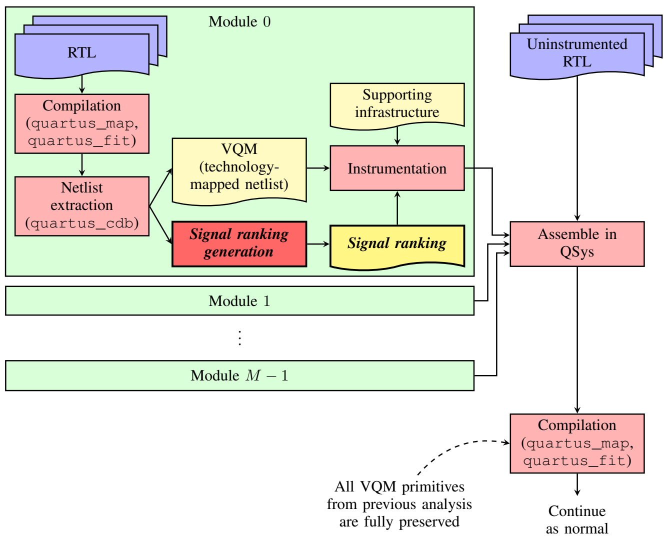
Module 0
RTL
Compilation (quartus_map, quartus_fit)
Netlist extraction (quartus_cdb)
VQM (technology-mapped netlist)
Signal ranking generation
Supporting infrastructure
Instrumentation
Signal ranking
Module 1
Module $M-1$
Uninstrumented RTL
Assemble in QSys
Compilation (quartus_map, quartus_fit)
All VQM primitives from previous analysis are fully preserved
Continue as normal

Fig. 4. Automated tool flow for signal selection and instrumentation of arbitrary multi-module systems [1]. Steps performed for signal ranking generation, shown emphasised, change dependent on the chosen methodology.

$\boldsymbol{X}_2$ contains $S-N$ columns not to monitor. Note that $\boldsymbol{X}$ can be obtained by simulating a module for $T+1$ clock cycles.

A high-quality choice of $\boldsymbol{P}$ determines two key factors. Firstly, that columns in $\boldsymbol{X}_2$ are 'well explained' by columns in $\boldsymbol{X}_1$, implying that there is little reason to monitor the former. Mathematically, this means ensuring that $\min_{\boldsymbol{Z}} \|\boldsymbol{X}_1\boldsymbol{Z} - \boldsymbol{X}_2\|$ is as small as possible, with zero indicating that signals in $\boldsymbol{X}_2$ provide no useful information. Secondly, that columns of $\boldsymbol{X}_1$ are as linearly independent as possible, meaning that the convergence of the RLS algorithm used, described in Section II-A3, is fast and does not suffer from numerical ill-conditioning at runtime. This corresponds to the smallest singular value of $\boldsymbol{X}_1$ being as large as possible.

Fortunately, this mathematical framework has been studied, with Golub and Van Loan reporting a fast heuristic for the problem based on the QR decomposition of a given $\boldsymbol{X}$ [16]. By introducing column pivoting via $\boldsymbol{P}$ to the QR decomposition equation as $\boldsymbol{XP} = \boldsymbol{QR}$, $\boldsymbol{X}$ can be decomposed into an orthogonal $\boldsymbol{Q}$ and upper-triangular $\boldsymbol{R}$ with 1s placed in $\boldsymbol{P}$ such that $|r_{11}| \geq |r_{22}| \geq \cdots \geq |r_{TT}|$. Hence, $\boldsymbol{P}$ reveals the column (signal) ordering of $\boldsymbol{X}$ resulting in a decomposition with such non-increasing diagonal elements of $\boldsymbol{R}$, thereby allowing $\boldsymbol{X}_1$ to be formed for any N. It is this *rank-revealing* QR decomposition that we rely on in this paper.

IV. Tooling

Figure 4 is a high-level representation of the tool flow we used to analyse and instrument systems assembled in QSys. This is based on that in our prior work [1], with signal-ranking tooling inserted as shown emphasised. Once each of the M modules of interest is compiled into a VQM and signal ranking files are generated per the steps outlined in the subsections that follow as dictated by the chosen methodology, new, instrumented netlists are produced as detailed in Section IV-C. The instrumented VQMs are then substituted for the original RTL sources and full-system compilation commences.

A. Graph Extraction

To work around not having direct access to Quartus' proprietary data, we wrote a script that transforms a given VQM into a directed graph: in our case, a list of all signal $\rightarrow$ signal edges. To do this, the script first establishes the {input, inout} $\rightarrow$ {output, inout} port relationships within all primitives—low-level modules such as LUTs and flip-flops—by parsing the vendor's simulation files used to describe their behaviour. The VQM is then parsed, and, for each instance found, edges are generated for every {input, inout} $\rightarrow$ {output, inout} connection pair within it. For example, if a signal x feeds a dffeas (flip-flop) instance's d (input) port and its output q drives a second signal y, an x $\rightarrow$ y edge is created because we know from the earlier primitive parsing that a d $\rightarrow$ q relationship exists within the dffeas primitive. We used NetworkX's [17] implementation of PageRank to transform our edge lists into vertex (signal) rankings, with the most central ranked the highest.

B. Simulation

VQM netlists are not suitable for simulation since their primitive instances are rarely fully connected; instantiations are written out in 'shorthand,' with unconnected ports simply excluded. Verilog Output (VO) files—structurally identical to VQMs when generated for the same module save for the inclusion of input/output buffering in the former—do not suffer from this affliction, but they are not able to be synthesised. Consequently, we are forced to work with both netlist types. Unfortunately, the two do not share a signal-naming convention, so those found to be of importance in one netlist type cannot be used directly in the other. To overcome these restrictions, we simulate using VO netlists and perform signal name conversion by parsing and matching against signals found in the equivalent VQMs thereafter, allowing us to produce consistently formatted lists.

Our simulation script takes a module's VQM and VO netlists as input and uses Modelsim to compile and simulate the VO, writing the internal nodes' logic levels to a value change dump (VCD) file. All flip-flop instances are found and set to random initial starting states before simulation commences, as are input ports (excluding clock and reset signals). In each of the $T+1$ simulation steps that follow—where T is user-defined—input port bits are inverted with a likelihood determined by a toggling probability factor and the clock is advanced. We left the probability fixed at 12.5% in our evaluation to match that used by industrial tools. Note that although our simulations are vectored, in line with our prior work [1] we do without specific input stimuli or testbenches for generality and ease of use. Once a simulation completes, the resultant VCD is converted—including VO-to-VQM signal name conversion—to a 'toggle' file format: a normally zero matrix $\boldsymbol{X}$ containing ones for the signals (columns) whose values changed in given cycles (rows) along with a list of columns' signal names. Toggle files form the input to the decomposition described in Section III-B, implemented using SciPy [18], whence signal rankings are generated.

C. Netlist Augmentation

Each of the aforementioned tools, along with one written to implement KAPow's ranking method, produces identically formed text files of signal names ordered line-by-line from most to least significant. A script consumes a given VQM and ranking file, augmenting the netlist with the instrumentation described in Section II-A2; N activity counters, each W bits wide, are selected and connected to the N most-significant *legal* signals found in the ranking file. A preprocessing step is used to establish legal signals: the VQM is parsed and a list of nets connected to known-good instance outputs, e.g. `dffeas q` ports, is generated. This ensures that counters are not fed by problematic signals, such as those constituting carry chains or LUT-to-register pass-throughs ('feeders'); instrumenting these would cause unacceptable structural changes to the circuitry.

V. Benchmarks

We used two multi-module systems to evaluate the costs and effectiveness of our proposed signal selection techniques. Modules were described in RTL, with each wrapped in a harness that facilitated communication with a host OS and fed the module with values taken from an embedded RAM; signal selection and instrumentation were performed per module as outlined in Section IV. Each module's stimuli could be changed at runtime by overwriting the contents of the respective wrapper's RAM from the host software.

A. Filter Bank

Our first system was that originally used to benchmark KAPow [1], thereby enabling direct comparison: a bank of seven FIR filters, each with 5×5 fixed-point Q4.8 convolution kernels operating on 240×160 8-bit grayscale images, where each filter formed an independently controllable module. The Cyclone V targetted contained 112 DSP blocks but, since the seven filters demanded a total of 175 multipliers, a differing portion—from none to all (25)—of each filter's multipliers were mapped to LUTs rather than DSPs. Across the modules, 17802 signals were available for monitoring, from 2384–2715 per filter. The pre-instrumentation system used 39% of the ALMs, over 74% of the available logic array blocks (LABs), 43% of its registers, 95% of block RAMs and 80% of the DSPs available on the device. Modules were able to be clocked independently by a runtime-adjustable PLL from 10–200MHz.

Filter workload could be changed at runtime by selecting a different combination of input data, coefficient values and clock frequency. The nine input data sets consisted of two checkerboard (repeated minimal and maximal values) and two gradient (linearly increasing values) patterns with different periods, two groups of uniform random values and red, green and blue frames of Lena. The ten sets of coefficients contained an all-zero, identity, two different 3×3 edge detection kernels, 3×3 and 5×5 box blurs, two Gaussian blurs (also 3×3 and 5×5), a 3×3 sharpen kernel and 5×5 unsharpen.

B. Arithmetic Toolbox

Eight modules composed the second system, each based around an arithmetic core taken from Altera's IP library: exponential, logarithm, power and division single-precision floating-point blocks, (co)sine- and arctangent-calculating 32-bit CORDIC modules and two FIR filters, one with 16-bit and the other 32-bit coefficients, operating on 8-bit data. 22215 signals were spread across the system, unevenly spread with between 214 for the smallest (16-bit FIR) to 8080 in the largest (arctangent CORDIC) modules. The uninstrumented 'toolbox' system occupied 31% of the available ALMs across 60% of its LABs, 29% of registers, 98% of block RAMs and 71% of the total DSPs. As for the filter bank system, modules were able to be clocked independently, in this case from 10 to 100MHz.

Input stimuli for the arithmetic toolbox were generated as required from six different random distributions: two uniform (narrow and wide), triangular, exponential and two Gaussian (monomial and binomial), formatted as appropriate for the data type of the module being exercised. γ-bit fixed-point uniform data was limited to either $\left[0, 2^{\gamma/2} - 1\right]$ (narrow) or $[0, 2^{\gamma} - 1]$ (wide), while floating-point data was bounded within $[-1, 1]$ (narrow) or $\left[-2^{64}, 2^{64}\right]$ (wide). Fixed-point triangular data was selected between $[0, 2^{\gamma} - 1]$, with floating point in $\left[-2^{64}, 2^{64}\right]$, both with symmetry around the midpoints, while exponentially distributed values had mean $\mu = 2^{\gamma-1}$ or 1000 for fixed and floating point, respectively. Gaussian distributions had $\mu = 2^{\gamma-1}$ and standard deviation $\sigma = 2^{\gamma-3}$ in the fixed-point case and $\mu = 0$, $\sigma = 2^{16}$ for floating point. Binomial Gaussian values were selected from either $\mu = 2^{\gamma-2}$ or $2^{\gamma-1} + 2^{\gamma-2}$, both with $\sigma = 2^{\gamma-4}$ (fixed point), and $\mu = 2^{-16}$ or 2^{16}, with $\sigma = 2^{16}$ (floating point), with a 50% chance of choosing either μ.

VI. Experiments & Results

Altera's Quartus II 64-bit toolchain, version 15.0.0, was used to compile the benchmarks for a Cyclone V SX (5CSXFC6D6F31C6), a 42k-adaptive logic module (ALM) FPGA coupled to a dual-core ARM Cortex-A9 CPU, upon which we ran Ubuntu 14.04. The Cyclone V SX System-on-Chip Development Board used features a Linear Technology LTC2978 power regulator for controlling FPGA core voltage, from which we obtained system power measurements. Online modelling, detailed in Section II-A3, was run in software on the hard CPU cores; we used RLS parameters $\boldsymbol{P}[0] = 1000\boldsymbol{I}$, $\hat{\boldsymbol{x}}[0] = \boldsymbol{0}$ and $\lambda = 0.999$ throughout for consistency with our prior work [1], and counter width W was always 9. We kept core voltage fixed at 1.1V, CPU frequency at 925MHz and ran the CPU-FPGA bus at 50MHz during all experiments.

A. Power Estimation Accuracy

Of particular concern is the ability for the power estimation model to produce accurate module-level breakdowns with signals chosen using each of the proposed selection methodologies. To gauge this, we conducted experiments in iterations, each of which consisted of the application of a different workload—a random combination of clock frequency and input stimuli (and coefficients, for the filter bank)—to each module, after which activity and system power measurements were captured to update the model. On every tenth iteration, modules were successively clock gated, with power measurements repeated to establish their true power consumptions; these were then used for comparison against the model output.

The LTC2978 power regulator used was found to give measurement errors in the $\pm$5mW range; estimates below this magnitude are therefore considered to be below the noise floor, and are thereby indistinguishable. We quote absolute rather than relative errors consistently due to the wide variations in module power consumptions and the aforementioned behaviour of the power regulator we use for system- and module-level measurements. System power was typically around 1W.

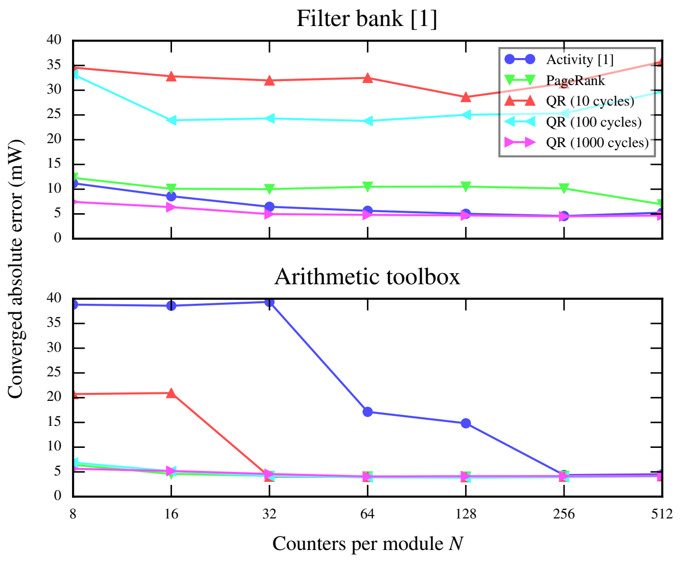

Fig. 5. Average absolute per-module power estimation error of instrumented systems by signal selection method and N. Results shown are averaged across the final 100 power breakdowns obtained for each experiment.

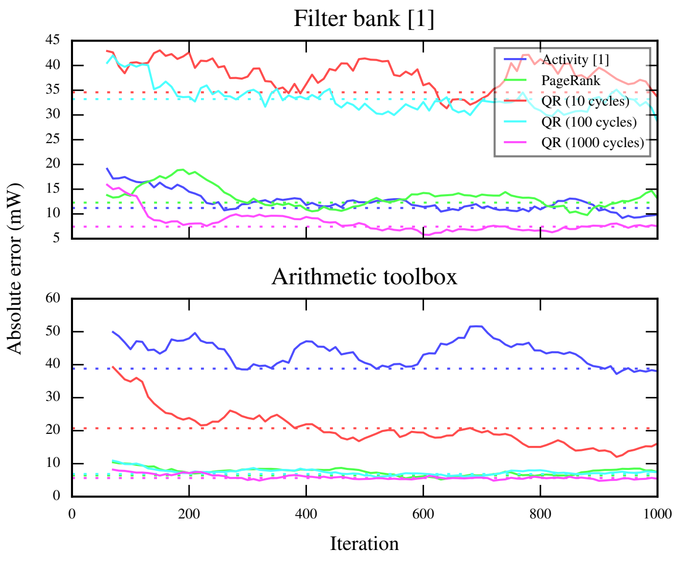

Fig. 6. Changes in average per-module power estimation error over time for instrumented systems by signal selection method with $N = 8$ ($\sim$0.3% of all signals). Results shown are for the first 100 power breakdowns obtained for each experiment, commencing after $MN + 1$ iterations have elapsed. 10-point moving-average windowing was applied to highlight the models' trends. Dashed lines represent the respective converged errors shown in Figure 5.

Figure 5 shows the average absolute power estimation error for each system's modules across the final 10% of iterations performed for each signal selection method. Formally, this is $^1/_{TM}\sum_{t=0}^{T-1}\sum_{m=0}^{M-1}|\hat{y}_m[t] - y_m[t]|$, where $\hat{y}_m[t]$ and $y_m[t]$ are the modelled and measured module powers, respectively, at time step t, M is the number of modules and $T = 100$ in this case. We can see that, while it always performs well ($\sim$5mW converged absolute error) for $N \geq 256$ ($\sim$10% or more of all signals), KAPow's activity-based selection does not produce consistently accurate estimates for our second benchmark system; errors close to 40mW were seen with $N \leq 32$ ($\sim$1% or fewer). PageRank, in contrast, demonstrates generality; low converged errors of 12 and 6.4mW were observed with $N = 8$ ($\sim$0.3%) for the filter bank and arithmetic toolbox, respectively. A clear dependency exists between simulation time and the quality of resultant signal selections for the QR decomposition-based technique. While capable of achieving 5mW error with $N \geq 32$ for the arithmetic system, estimates for 10-cycle QR were poor for the filter bank—with error actually increasing with N for $N > 128$—as were those with simulations completed for 100 cycles. 1000-cycle QR-based selection, however, yielded outstanding results: 7.4 and 5.6mW converged errors for the systems with $N = 8$—improvements of 1.5× and 6.9× over KAPow's default selections with the same N—with both dropping below the noise floor for $N \geq 32$, whereupon coverage exceeded $\sim$1% of signals.

The FIR modules were deeply pipelined and each contained several high-fanout nets, suggesting that selection techniques based on connectivity (PageRank) and temporal correlation (QR) should both perform well, which our results validate. The arithmetic modules were less deeply pipelined but more hierarchical in structure; the excellent estimation errors achieved for this benchmark therefore support eigencentrality's effectiveness in encapsulating structural information in ways that 'flatter' metrics, such as fanout, cannot.

Also of interest are the rates of model convergence and relative noise of errors for each signal selection type. In Figure 6, the average module-level absolute error is plotted over time for systems with eight counters per module, the lowest number tested, for the first 10% of experimental iterations. The converged error results from Figure 5 are included as dashed lines. It can be noted that rates of convergence are related to converged error magnitudes; methods that perform poorly in terms of converged error also tend to take longer to reach those long-term errors than their better-performing alternatives. A similar relationship exists between error magnitude and noise, with lower-converged error methods such as 1000-cycle QR exhibiting far less deviation after convergence than their higher-error counterparts.

B. Compile-time Overheads

The number of counters per module N determines the area and compilation time overheads incurred through instrumentation. Since the signal selection's quality dictates the N necessary to achieve a given estimation accuracy, we can use such overheads to quantify the gains that can be realised by using more sophisticated ranking techniques. Figure 7 shows approximately linear relationships between N and the additional ALMs, registers and compilation time needed to assemble a system over its uninstrumented equivalent, i.e. that with $N = 0$; the change in LABs is less straightforward since it is determined by a packing heuristic. For $N = 8$, or $\sim$0.3% of the average module's signals, mean overheads in ALMs, registers and compilation time of 1.1%, 2.0% and 0.42% were found, respectively, across the two benchmark systems.

The compilation times obtained all include that which was required to compile modules into VQM netlists, generate the QSys system and compile the resultant RTL, but exclude that consumed by signal selection since this is dependent upon the particular method chosen. For instrumented systems they also

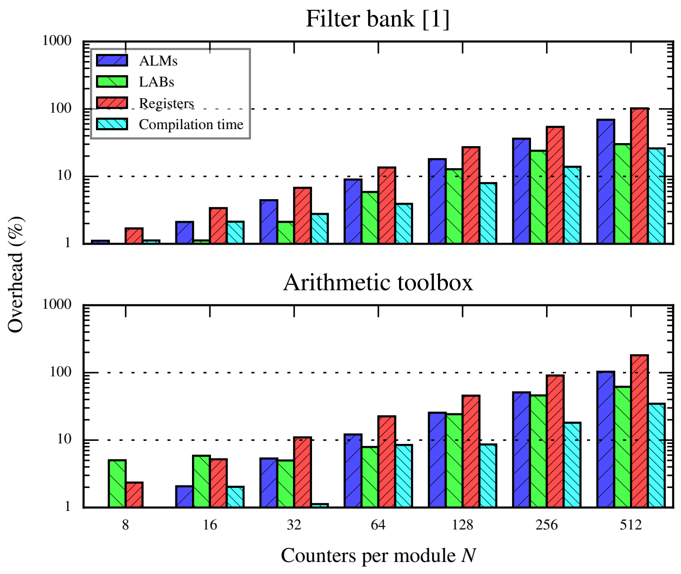

Fig. 7. Resource usage and compilation time overheads of instrumented systems by N, independent of signal selection method. Results are normalised to those for the equivalent uninstrumented system.

include the time needed to instrument the VQMs as outlined in Section IV-C. Note that while netlist generation is not strictly necessary for uninstrumented systems, it was done for fairness of comparison; this would not need to be performed as a separate step in a fully integrated tool flow at all.

Figure 8 demonstrates the wide range of times needed to complete the tasks required for generating lists of signals ranked by differing metrics. The VQM parsing needed to extract graphs was fast, taking 3.9s to complete per module when averaged across the 15 present in our two benchmark systems, while running PageRank on those graphs consumed just 2.7s per module. Simulations used to generate the toggle files needed for QR decomposition, on the other hand, took significant time to complete, with 10-, 100- and 1000-cycle simulations and subsequent signal name and file format conversions taking average per-module times of 22s, 48s and 360s, respectively. The subsequent QR decompositions resulted in small additions to those times, bringing the totals to 23s, 49s and 370s. In contrast, the average duration observed for KAPow's activity-based ranking was 15s.

Since the overheads shown in Figure 8 were also normalised to the respective uninstrumented system's compilation time, they can be summed with the results presented in Figure 7 to give a total overhead for a given N and selection type. For $N = 8$, for example, PageRank- and QR (1000-cycle)-based selections resulted in total compilation time overheads of 2.9% (60s) and 91% (1900s) for the filter bank and 3.0% (61s) and 180% (3700s) for the arithmetic toolbox, respectively.

C. Runtime Overheads

Our final concerns lie with the power overhead of, and modelling complexity when using, activity-counting instrumentation, both of which are determined by N but not by the signal selection method chosen. The results presented in Figure 9 were obtained for each system by applying 100 randomly generated workloads, created as described in Section VI-A,

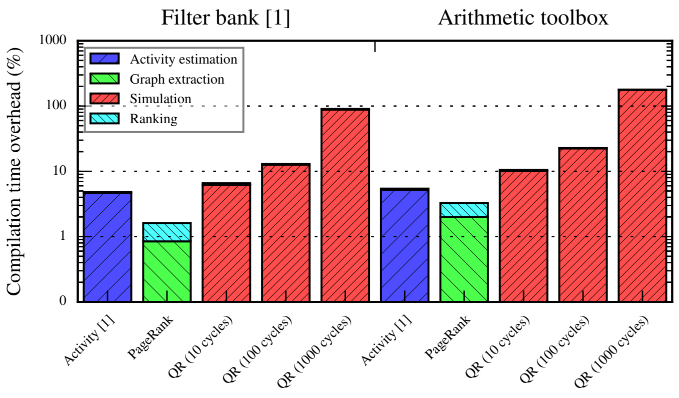

Fig. 8. Signal ranking-induced compilation time overhead of instrumented systems showing breakdowns of ranking tasks, independent of N. Results are normalised to compilation times for the equivalent uninstrumented system.

to designs with each N in turn. For each workload, 10 pairs of activity and power measurements were taken and used to update the model at a rate of 0.2Hz; a limitation imposed by the speed of modelling with $N = 512$, although not necessarily an unrealistic update rate. We ran all modules at their maximum-rated frequencies throughout the experiment.

It can be seen that, while there is a straightforward exponential relationship between N and the model update time, the effect of N on power is less obvious, particularly for the filter bank. Here, power was actually found to marginally decrease for lower values of N, being around 0.5–1.4% below that for $N = 0$ with $N \leq 64$ (less than around 2% of the available signals). The arithmetic toolbox exhibited the worst-case power overheads, with 3.5% (43mW averaged across all tested workloads) found for $N = 8$ ($\sim$0.3% of signals) while for $N = 512$ ($\sim$20%) it was 13% (160mW). The slight noisiness, including the aforementioned dips below zero, can be attributed to non-preservation of placement and routing information between the separate module-wise and whole-system compilations shown in Figure 4. Clearly, however, there is value in reducing N, where possible, from both power and modelling complexity perspectives, the latter facilitating either more frequent model updates or lower CPU usage.

Both PageRank- and 1000-cycle QR-based selections have shown generality and high quality in our analysis; the key distinguishing factor between them is compilation time. Since the cycle-accurate simulations necessary to perform QR decompositions take significant time to complete, we can conclude that PageRank is likely to be 'good enough' for the majority of situations but, where very high estimation accuracy is required, QR-based ranking is a viable alternative. In situations where such high accuracy is unnecessary, our results still demonstrate value in using improved signal selection. For the arithmetic system, for example, achieving sub-20mW accuracy requires $N = 64$ when using KAPow's default method. PageRank, on the other hand, can produce twice-as-accurate rankings in less time with just $N = 8$, resulting in area overhead and modelling complexity reductions of over an order of magnitude each.

VII. Conclusion

In this paper, we introduced two new ranking techniques facilitating intelligent signal selection for runtime power es-

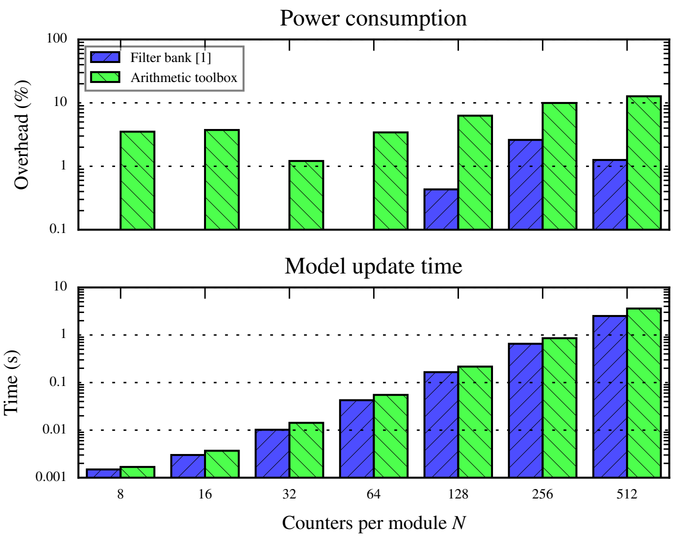

Fig. 9. Power overhead and model update times of instrumented systems by N, independent of signal selection method. Power results are normalised to those for the equivalent uninstrumented system.

timation (STRIPE): structural analysis based on PageRank, a fast centrality metric, and statistical analysis using rank-revealing QR decomposition, taking signal values gleaned from simulation as input. We presented the tooling required to automate their application, using KAPow, our existing power instrumentation and modelling framework, to assess their relative effectiveness primarily in terms of estimation accuracy. Other metrics explored as part of this work included fanout, vectorless estimation-derived power estimates and alternative centrality scores including closeness and betweenness, however all performed worse than those proposed and were therefore not discussed for the sake of brevity.

We showed that the use of PageRank on graphs created from device vendor netlists could generate selections resulting in converged power estimation accuracy of $\sim$9mW in an average of just 6.6s per module for $N = 8$, the lowest tested and representing coverage of $\sim$0.3% of the average module's signals. With significantly more time spent up-front—370s per module, on average—on analysis including 1000-cycle simulation and QR decomposition, $\sim$7mW was achievable for the same N, marginally above our noise floor. These results compare favourably to those for KAPow's original activity-based signal selection, for which the equivalent error was $\sim$25mW across the two multi-module systems we tested. For the arithmetic benchmark system, the use of our proposed QR decomposition-based signal selection in place of its switching activity-based equivalent achieved approximately the same converged estimation error while allowing for area, power and modelling complexity overhead reductions of 98%, 65% and 100% (to 2 s.f.), respectively. We envisage using PageRank as KAPow's default ranking method from now on, since it is faster and generally produces higher-quality results than the method it replaces, while also leaving QR-based selection as an option when very high-accuracy estimates are desired.

In the future, we will explore using combinations of structural and statistical analyses with the aim of pushing the boundaries of the required number of monitors, reducing N as much as possible while still achieving high estimation accuracy. We will also look at possible enhancements to our simulations that may allow the time needed for their completion to be reduced without sacrificing the quality of resultant QR decompositions. Runtime signal selection, likely fine-tuning of initial, wider selections made at compilation time, will also be explored, potentially making use of dynamic partial reconfiguration. Finally, we would like to investigate whether it is possible to use our analysis backwards: estimating the minimum N needed per module of an arbitrary system to achieve power estimation errors with a given accuracy.

Acknowledgements

This work was supported by the EPSRC-funded PRiME Project (grant numbers EP/I020357/1 and EP/K034448/1).
http://www.prime-project.org

The authors also acknowledge the support of Imagination Technologies and the Royal Academy of Engineering.

References

[1] E. Hung, J. J. Davis, J. M. Levine, E. A. Stott, P. Y. K. Cheung, and G. A. Constantinides, "KAPow: A System Identification Approach to Online Per-module Power Estimation in FPGA Designs," in *IEEE International Symposium on Field-programmable Custom Computing Machines (FCCM)*, 2016, pp. 56–63.

[2] F. N. Najm, "A Survey of Power Estimation Techniques in VLSI Circuits," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 2, no. 4, pp. 446–455, 1994.

[3] J. H. Anderson and F. N. Najm, "Power Estimation Techniques for FPGAs," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 10, pp. 1015–1027, 2004.

[4] A. Lakshminarayana, S. Ahuja, and S. Shukla, "High-level Power Estimation Models for FPGAs," in *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2011, pp. 7–12.

[5] N. R. Potlapally, A. Raghunathan, G. Lakshminarayana, M. S. Hsiao, and S. T. Chakradhar, "Accurate Power Macro-modeling Techniques for Complex RTL Circuits," in *International Conference on VLSI Design*, 2001, pp. 235–241.

[6] K. M. Zick and J. P. Hayes, "On-line Sensing for Healthier FPGA Systems," in *ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA)*, 2010, pp. 239–248.

[7] M. Najem, P. Benoit, F. Bruguier, G. Sassatelli, and L. Torres, "Method for Dynamic Power Monitoring on FPGAs," in *International Conference on Field-programmable Logic and Applications (FPL)*, 2014, pp. 1–6.

[8] E. Hung and S. J. E. Wilton, "Scalable Signal Selection for Post-silicon Debug," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 6, pp. 1103–1115, 2013.

[9] M. J. Walker, S. Diestelhorst, A. Hansson, A. Das, S. Yang, B. M. Al-Hashimi, and G. V. Merrett, "Accurate and Stable Run-time Power Modeling for Mobile and Embedded CPUs," *IEEE Transactions on CAD of Integrated Circuits and Systems*, vol. 36, no. 1, pp. 106–119, 2017.

[10] W. W. Leontief, *The Structure of the American Economy, 1919–1929; an Empirical Application of Equilibrium Analysis.* Harvard University Press, 1941.

[11] J. R. Seeley, "The Net of Reciprocal Influence: A Problem in Treating Sociometric Data," *The Canadian Journal of Psychology*, vol. 3, no. 4, pp. 234–240, 1949.

[12] L. Katz, "A New Status Index Derived from Sociometric Analysis," *Psychometrika*, vol. 18, no. 1, pp. 39–43, 1953.

[13] C. H. Hubbell, "An Input-Output Approach to Clique Identification," *Sociometry*, vol. 28, no. 4, pp. 377–399, 1965.

[14] S. Brin and L. Page, "The Anatomy of a Large-scale Hypertextual Web Search Engine," *Computer Networks and ISDN Systems*, vol. 30, no. 1, pp. 107–117, 1998.

[15] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning: with Applications in R.* Springer, 2013.

[16] G. Golub and C. Van Loan, *Matrix Computations.* Johns Hopkins University Press, 1996.

[17] NetworkX, "High-productivity Software for Complex Networks," https://networkx.github.io/, 2016.

[18] SciPy, "SciPy," https://www.scipy.org/, 2017.