Speeding up System Identification Algorithms on a Parallel RISC-V MCU for Fast Near-Sensor Vibration Diagnostic

Amirhossein Moallemi*, Riccardo Gaspari, Federica ZonziniLuca De Marchi*, Davide Brunelli[1]**, and Luca Benini[3]**

Dept. of Electrical, Electronic and Information Engineering, University of Bologna, Italy
[1] *Dept. of Industrial Engineering, University of Trento, Italy*
[2] *Dept. of Information Technology and Electrical Engineering, ETH Zürich, Switzerland*
* *Member, IEEE*
** *Senior Member, IEEE*
*** *Fellow, IEEE*

Abstract—Structural Health Monitoring (SHM) systems currently utilize a combination of low-cost, low-energy sensors and processing units to monitor the conditions of target facilities. However, utilizing a dense deployment of sensors generates a significant volume of data that must be transmitted to the cloud, requiring high bandwidth and consuming substantial power, particularly when using wireless protocols. To optimize the energy budget of the monitoring system, it is crucial to reduce the size of the raw data near the sensors at the edge. However, existing compression techniques at the edge suffer from a trade-off between compression and accuracy and long latency resulting in high energy consumption. This work addresses these limitations by introducing a parallelized version of an unconventional data reduction method suited for vibration analysis based on System Identification models. Our approach leverages the unique capabilities of GAP9, a multi-core RISC-V MCU based on the parallel ultra-low power (PULP) architecture. Compared to the sequential implementation, we achieve a maximum execution time reduction of $\approx 60\times$ and power consumption of just $48.3\,\mathrm{mW}$ while preserving the spectral accuracy of the models.

Index Terms—Embedded Systems, RISCV architecture, System Identification, Vibration analysis

I. INTRODUCTION

The safety of large-scale civil infrastructures, aerospace vehicles, and industrial machines has raised the government's attention in the last decades to tackle the environmental impacts affecting such complex systems [1], [2]. Structural Health Monitoring (SHM) explores techniques to continuously monitor the conditions of technical facilities, which are susceptible to various damaging phenomena [3]. Vibration-based monitoring, in particular, is the most effective technique for inspecting structures in the dynamic regime, i.e., structures that are completely characterized by frequency-related quantities [1], [4].

Current SHM systems are composed of sensor nodes distributed over the target structure, each of which embeds several low-cost sensors and a computational unit that is able to sense and outsource data to the cloud, where information is stored and further processed to assess the level of structural integrity [5]. This sensor-to-cloud continuum generates a big volume of information, in the range of hundreds of MB per day for a single structure, leading to expensive and non-scalable solutions that barely adapt to large-scale scenarios [6]. Moreover, this continuous flow of long-time series requires large bandwidths and grid-based powering since the transmission phase is very expensive in energy consumption and can rapidly exhaust the capacity provided by battery-operated supplies.

To address these challenges, a new generation of smart SHM systems based on edge processing has emerged. In this paradigm, raw data is processed near the sensors, reducing network traffic between the edge sensor and the cloud. Implementing data reduction techniques at the sensor level is crucial to minimize the amount of data transmitted, resulting in shorter transmission times and negligible impact on sensor energy budgets [7]. Different reduction techniques for vibration data were successfully implemented onboard, spanning from Principal Component Analysis (PCA) to Compressed Sensing (CS). For example, in [8] and [7], authors' embedded lightweight versions of PCA in microprocessors, reaching a compression level of $1.4\times$ and $15\times$, respectively. In [9], an adapted version of CS has been presented to handle vibration data showing good performances for reduction levels up to $6\times$. However, all these strategies present some drawbacks, such as the fact that they need training data for the optimal definition of compression parameters, along with the limited compression level they can allow for to ensure a sufficient quality in the retrieved structural properties [9].

Alternatively, System Identification (SysId) has been proposed in [10] as an unconventional but promising data compression method for vibration data processing, allowing up to $50\times$ dimension reduction, which is at least an order of magnitude higher than the ones mentioned above. SysId is a signal-processing technique that builds a mathematical model of a dynamic system based on a linear time-invariant filter, whose taps, also called model parameters (Θ), can reproduce the observed system dynamics. By knowing model parameters, the power spectrum can be computed, from which all the frequency-related properties of the structure (e.g., natural frequencies) can be extracted. The advantage of SysId is that just a dozen of model parameters are sufficient to accurately replicate the observed system response, even for the most complicated geometries [11]. Thus, by transmitting Θ instead of raw data, which usually amounts to thousands of samples, one can reach very large compressing factors.

However, implementing SysId algorithms involves computationally and memory-intensive operations. Few attempts have been made in the literature to implement SysId in near-sensor scenarios. One of the very first implementations can be found in [12], in which authors succeeded in porting SysId filters on the Imote sensor platform; nonetheless, the limitations on storage constraints of this board forced the adoption of input-output correlation-based schemes, which are not compatible with the execution of output-only solutions as the ones of real interest for on-condition maintenance where the exciting stimulus is always non-measurable. These issues were overcome in a more recent work [10] by resorting to advanced algebraic

Corresponding author: A. Moallemi (e-mail: amirhossein.moallem2@unibo.it).

procedures taken from the big data processing framework, which allowed to fit output-only models on a prototype sensor equipped with an STM32L5 based on an ARM Cortex-M33 microcontroller unit (MCU). Nevertheless, the implementation offered in [10] suffers from one crucial limitation, which is the long execution time (greater than 120 s in the most cumbersome scenario) due to the sequential nature of the computational workflow forced by the single-core architecture of the computing processor. Eventually, this evidences the lack of on-sensor SysId implementations compatible with continuous and real-time diagnostic functionalities.

In this work, we offer a significant step forward in the deployment of SysId as an effective data compression technique for extreme edge sensors by tackling the challenges related to long latency and high power consumption. We reach this goal by:

- Moving to parallel implementation of output-only SysId algorithm on a multi-core RISC-V MCU, GAP9, achieving a maximum worst-case execution time of 1.65 s @110 MHz clock system, with an energy consumption of 80.1 mJ.
- Compared with single-core MCU implementations, we achieved a speed-up close to 60× while maintaining the same spectral accuracy.

The rest of this paper is organized as follows. In Sec. II, the main contribution is presented, in which we introduce the parallelization steps toward the SysId algorithm. Further, the multi-core MCU, GAP9, the platform used to embed the parallel version of SysId, is introduced. Then, Sec. III examines the precision of SysId performed on the GAP9 platform. Further, it assesses the model in terms of execution time, energy consumption, and memory footprint. Finally, Sec. IV concludes this work.

II. SOFTWARE & HARDWARE FRAMEWORK

A. Output-only SysId models for vibration analysis

Two output-only SysId models have been implemented on the target platform since they are among the most effective for processing ambient-excited vibration data: the Autoregressive (AR) and Autoregressive with Moving Average (ARMA) [13]. Given an N-long discrete time-series sampled at regular intervals kT_s (k being the generic time index), the mathematical formulation of the AR and ARMA models are described by Eq. (1) and (2), respectively:

$$s[k] + \sum_{i=0}^{q} \theta_i s[k-i] = e[k] \tag{1}$$

$$s[k] + \sum_{i=1}^{q} \theta_i s[k-i] = e[k] + \sum_{s=0}^{p} \gamma_s e[k-s] \tag{2}$$

In these expressions, $N_p = p + q + 1$ is defined as the model order, while $e[k]$ is assumed equal to a zero–mean white noise Gaussian term with prescribed variance, serving as a proxy of the unknown input force. The length N is conveniently selected proportionally to N_p according to $N = N_p N_{s/1p}$, $N_{s/1p}$ being the number of time samples necessary to identify one single model parameter accurately. Therefore, SysId aims at computing the N_p model parameters, a task that can be fulfilled by means of ordinary least-squares (OLS) applied to the linear regression form of Eq. (1) and (2), which generically reads as

$$Y = \Psi\Theta \tag{3}$$

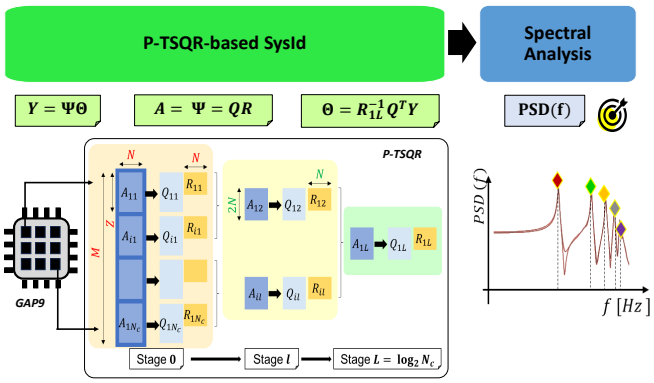

Fig. 1: Workflow of the proposed P-TSQR-based SysId approach proposed in this work.

with $Y \in \mathbb{R}^{N\times 1}$ and $\Psi \in \mathbb{R}^{N\times N_p}$ being the measured vibration response and the regression matrix, respectively[1].

B. Parallel tall-skinny QR for SysId

Factorizing the regression matrix via QR decomposition, i.e., $\Psi = QR$, is required to reduce the phenomena of numerical instability and rounding effects of the OLS algorithm employed in conventional SysId solutions. This yields the solution of Eq. (3) to be computed as

$$\Theta = R^{-1} Q^T Y \tag{4}$$

However, the memory requirements for standard QR decomposition make it inapplicable in most of the extreme-edge computing domains. To overcome this issue, the *parallel tall-skinny QR decomposition* (P-TSQR), a different QR decomposition algorithm suited for big data processing frameworks [14], is proposed in this work. P-TSQR has to be preferred over other QR decomposition methods since its parallel implementation scheme (doable for a multi-core processing framework) allows to significantly speed up the computation time over sequential (S-TSQR) variants, such as those exploited in [10] to accomplish the same task.

Given a generic input matrix $A \in \mathbb{R}^{M\times N}$ ($M \gg N$) to be decomposed and supposing that N_c chunks (or cores) are available for parallelization, P-TSQR follows a binary tree implementation requiring $L = \log_2 N_c$ iterations. The latter is depicted in Fig. 1 and can be described as:

- *stage 1*: A is divided into N_c partitions of dimension $Z = M/N_c$. For each i-th chunk $A_{i1} \in \mathbb{R}^{Z\times N}$, the QR decomposition is computed independently, leading to $A_{i1} = Q_{i1} R_{i1}$, with $R_{i1} \in \mathbb{R}^{N\times N}$ and $Q_{i1} \in \mathbb{R}^{Z\times N}$ the factorizing matrices.
- *next stages*: at generic stage $l \in [2, \ldots, L]$, the R_{i1} matrices are vertically concatenated two-by-two into N_c/l matrices $A_{il} = [R_{i(l-1)} \quad R_{(i+1)(l-1)}] \in \mathbb{R}^{2N\times N}$ which are then decomposed by a new step of QR factorization.

The whole computation ends for $l = L$ resulting in a single $R_{L1} \in \mathbb{R}^{N\times N}$ matrix, that is indeed the R matrix that would have been obtained by directly decomposing the whole initial matrix A, and a certain number of Q_{il} matrices, that can reconstruct the original Q matrix.

[1]For the definition of Ψ, see [10]

C. Hardware Platform

The GAP9 MCU [15] was exploited since it is an ultra-low-power multi-core microprocessor targeted for IoT applications at the extreme edge. GAP9 features 9 compute RISC-V cores and a single-core fabric controller based on the PULP (Parallel Ultra Low Power) instruction-set architecture (ISA) extensions [16]. The nominal frequency processor can be increased up to 370 MHz while keeping the power consumption in the nominal operating mode below 50 mW. Conversely, other multi-core RISCV-based MCUs (such as GAP8) or dual-core ARM Cortex devices provide less cores with the same or smaller clock frequency. Indeed, although multi-core devices such as the $STM32H745ZIT6$ runs at the same frequency as the GAP9 (i.e., 400 MHz), less current consumption and more number of cores make GAP9 a suitable MCU for the parallelizable applications. GAP9 speeds up the execution of Digital Signal Processing (DSP) algorithms and has a dedicated floating point unit (FPU) with 8,16, and 32 precision, whereas other MCUs are either not able to support FPU, e.g., GAP8 or provide a single precision FPU unit. GAP9 has a hierarchical memory architecture with 512 kB of fast L1, as well as L2-SRAM with 1.5 MB and an L2 non-volatile memory of 2 MB. Compared to its former version, i.e., GAP8 with 80 kB (6.4× less) of L1 and only 512 kB (≈ 3× less) of L2-SRAM, GAP9 provides more capacity for memory-hungry algorithms. Conversely, the STM32L5 exploited in [10] for the same task is limited to only 256 kB (≈ 6× less) of RAM and 512 kB (≈ 4× less) of flash.

III. EXPERIMENTAL VALIDATION

A. Experimental Setup

Vibration data collected from a four-storey frame structure under white noise base excitation (sampling rate equal to 50 Hz) were used for testing. All the possible combinations of N_p and $N_{s/1p}$ values were explored by varying the former quantity between 9 and 57 (step size equal to 8), whereas the latter was swept between {25, 30, 35}. The search space for N_p has been selected to model a wide range of target structures, considering that the number of parameters typically settles below a couple of dozens, even for the most complicated systems [11]. Besides, the choice for $N_{s/1p}$, which is responsible for the length of time series to be processed, has been selected to meet the storage capabilities of the computing unit even in the most cumbersome configurations. We show in Sec. III-B that the selected upper boundaries are large enough for robust modelling of the PSD of the input signal, which is, however, large enough for robust modelling.

B. Structural validation

The Itakura Saito Spectral Divergence (ISD) has been exploited to evaluate the level of spectral superimposition between the Power Spectral Densities (PSD) obtained via built-in MATLAB utilities ($PSD_{MAT}(f)$) and the one estimated from the GAP9 coefficients ($PSD_{GAP9}(f)$):

$$ISD = \frac{1}{N}\sum_{f=1}^{N}\left[\frac{PSD_{GAP9}(f)}{PSD_{MAT}(f)} - \log\left(\frac{PSD_{GAP9}(f)}{PSD_{MAT}(f)}\right) - 1\right] \quad (5)$$

PSD curves can be easily computed by moving Eq. (1) and (2) in the frequency domain, depending on the selected AR/ARMA model. ISD ranges between 0 and 1, with $ISD = 0$ meaning a perfect match between the two curves.

Results are reported in Table 1 in the ISD column block. As can be observed, all the values are stably below $1.05 \cdot 10^{-2}$ even for the worst-performing configuration, proving the spectral accuracy of the deployed SysId models. Additionally, it is worth observing that these results compare favorably with respect to the benchmark solution in [10], in which a maximum ISD of $0.93 \cdot 10^{-2}$ was scored when working with the same time series.

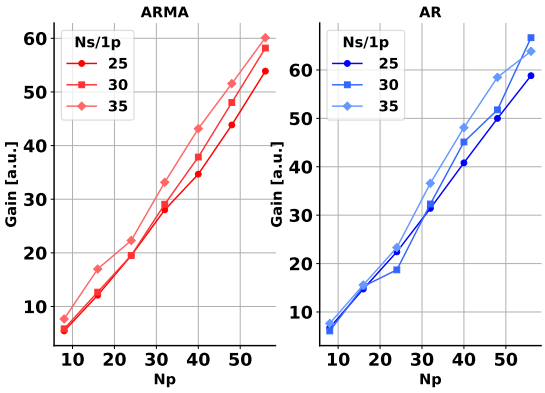

Fig. 2: Time gain achieved by moving from sequential to parallel implementation.

C. Hardware Profiling

1) Execution Time: Minimizing the algorithmic latency is fundamental to deploying near-sensor streaming data analysis. Accordingly, the execution time necessary to process one batch of SysId model parameters has been quantified by measuring the number of cycles N_{cycles} given the operating frequency F_{ck} of the processor, i.e., Exe. Time = N_{cycles}/F_{ck}. In order to perform a fair comparison with the setup in [10], GAP9 was clocked at F_{ck} = 110 MHz. Results are summarized in Table 1.

As expected, the highest computational time is associated with the maximum N_p = 57 and $N_{s/1p}$ = 35 parameters, i.e., those yielding to the longest vector to be processed (corresponding to 1995 time samples). More in detail, the largest AR and ARMA model requires 0.83 ms and 1.65 ms, respectively, which are almost 47× and 24× smaller than the time taken to acquire the input signal, the latter amounting to 1995/50Hz ≈ 40 s. Consequently, the devised pipeline proves to be compatible with the real-time execution of SysId at the sensor level.

More importantly, these outcomes significantly outperform the sequential SysId implementation in [10], characterized by worst-case execution times of 52.80 s and 99.12 s for the same parameters. Evidence is proved in Fig. 2, which showcases the time gain when moving from the sequential to the parallel implementation, expressed as the ratio between the execution time required when running in the STM32L5 board with respect to the one scored by the GAP9 platform. Independently from the considered $N_{s/1p}$, the gain achieved by the parallel solution follows a linear trend, moving from a minimum of 5.44× (N_p = 9, $N_{s/1p}$ = 25) to a maximum of 60.11× (N_p = 57, $N_{s/1p}$ = 35) in case the ARMA model is considered. The same speed-up ranges from 6.75× to 63.85× for the AR counterpart. Three main reasons can motivate this important improvement: i) hardware accelerators for matrix multiplication embedded in GAP9, ii) switch of paradigm from the sequential implementation of QR decomposition to the parallel version of it, i.e., moving from one core to 8 cores deployment, and iii) in the case of sequential implementation, a large matrix should be segmented to multiple chunks to execute a matrix multiplication, whereas the large L2 memory in GAP9 (compared

Table 1: Performance indicators for varying N_p and $N_{s/1p}$ when parallelizing the AR and ARMA SysId models on the GAP9 platform.

		ISD [a.u.]			**E [mJ]**			**Memory [KBytes]**			**Execution time [ms]**		
		$N_{s/1p}$			$N_{s/1p}$			$N_{s/1p}$			$N_{s/1p}$		
		25	**30**	**35**	**25**	**30**	**35**	**25**	**30**	**35**	**25**	**30**	**35**
ARMA	**9**	2.18e-05	3.21e-05	8.07e-06	0.74	0.82	0.75	18.44	22.46	26.88	15.80	17.50	16.00
	17	5.33e-03	9.04e-04	2.43e-03	2.75	3.21	3.18	70.80	86.58	103.93	58.60	68.10	67.70
	25	1.36e-03	2.39e-05	9.30e-03	6.94	8.29	8.77	157.19	192.46	231.25	147.00	176.00	186.00
	33	5.29e-04	4.76e-03	1.13e-02	13.80	16.30	17.20	277.61	340.11	408.86	290.00	345.00	360.00
	41	1.05e-02	8.79e-04	9.25e-03	25.90	28.40	30.40	432.07	529.53	636.75	545.00	600.00	635.00
	49	2.93e-03	5.64e-03	6.42e-03	41.20	46.00	51.50	595.71	733.06	884.46	863.00	965.00	1070.00
	57	9.8e-04	4.55e-03	3.03e-03	62.90	69.50	80.10	843.07	1033.65	1243.38	1300.00	1440.00	1650.00
AR	**9**	2.68e-04	2.41e-04	3.70e-05	0.33	0.43	0.40	15.18	17.93	20.88	6.80	9.10	8.40
	17	8.74e-04	9.20e-04	8.02e-04	1.26	1.50	1.70	57.74	68.44	79.93	26.80	31.80	36.10
	25	8.23e-04	7.88e-04	7.12e-04	3.41	4.52	4.71	127.80	151.65	177.25	72.10	95.90	100.00
	33	5.51e-04	5.42e-04	4.71e-04	7.03	8.28	8.76	225.36	267.55	312.86	148.00	175.00	183.00
	41	13.40e-04	11.70e-04	1.16e-04	12.70	13.60	15.30	350.43	416.15	486.75	264.00	286.00	320.00
	49	9.13e-04	1.63e-04	3.72e-04	20.80	24.30	25.60	502.99	597.44	698.93	434.00	508.00	535.00
	57	1.35e-04	3.87e-04	3.87e-04	32.60	34.30	39.40	683.05	811.43	949.38	681.00	721.00	827.00

with a maximum of 256 KBytes of RAM for the STM32L522 board) allows for the large matrix multiplications to be executed without chunking.

2) Memory footprint: The precision of SysId routines increases when working with longer time series; however, when dealing with memory-constrained devices, limitations have to be respected. Hence, the memory footprint of the models has been evaluated to find the maximum SysId configuration (N_p,$N_{s/1p}$) compatible with the available GAP9 storage capabilities. The memory column in Table 1 specifies the space occupied in the L2 memory for each combination, showing that the ARMA model utilizes, in general, nearly $1.3\times$ more memory than the corresponding AR model. This result is coherent with the inherently more complex nature of the ARMA routines [10]. Further, Table 1 reports that $Np = 57$ and $N_{s/1p} = 35$ put a tight constraint for embedding the ARMA model in GAP9 when precision is set to `float32`, as this configuration requires 1.2 MBytes. Nevertheless, this hardware constraint is compatible with the majority of civil and industrial facilities [11].

3) Energy Consumption: Low energy consumption plays a crucial role in a sustainable battery-based system in long-term monitoring. Table 1 reports energy consumption for one run of the SysId deploying AR and ARMA model. Noticeably, the energy demanded by ARMA is averagely double the one consumed by AR due to the two-step nature of the adopted ARMA algorithm. This is mainly due to the longer execution time of the ARMA models; however, notice that since both models computationally use merely matrix multiplication, they yield similar power consumption of 48 3mW.

IV. CONCLUSION

This work presented a parallelized implementation of a vibration compression algorithm based on SysId, necessary to decrease the volume of data transmitted to the cloud, using an ultra-low-power multi-core platform, namely GAP9, as the computing platform. We showed that by shifting the paradigm from sequential to parallel implementation, an improvement up to $\approx 60\times$ could be attained in terms of execution time, which is fundamental to avoid the long latency of the sequential version and enables the in-field deployment of the SysId algorithm for streaming vibration data processing. Further, we showed that the most power-hungry deployment of the model consumes 48.3 mW per each run of the algorithm, making it suitable for long-term self-sustainable battery-based monitoring systems.

ACKNOWLEDGMENT

This research was partially funded by PNRR – M4C2 – Inv. 1.3, PE00000013 – "FAIR" project – Spoke 8 "Pervasive AI," funded by the EU under the NextGeneration EU programme. Further, this work was supported by the Italian Ministry for University and Research (MUR) under the program "Dipartimenti di Eccellenza (2023-2027)".

REFERENCES

[1] S. S. Saidin, S. A. Kudus, A. Jamadin, M. A. Anuar, N. M. Amin, A. B. Z. Ya, and K. Sugiura, "Vibration-based approach for structural health monitoring of ultra-high-performance concrete bridge," *Case Studies in Construction Materials*, vol. 18, p. e01752, 2023.

[2] A. Sabato, C. Niezrecki, and G. Fortino, "Wireless mems-based accelerometer sensor boards for structural vibration monitoring: a review," *IEEE Sensors Journal*, vol. 17, no. 2, pp. 226–235, 2016.

[3] P. F. Giordano, S. Quqa, and M. P. Limongelli, "The value of monitoring a structural health monitoring system," *Structural Safety*, vol. 100, p. 102280, 2023.

[4] A. Kamariotis, E. Chatzi, and D. Straub, "A framework for quantifying the value of vibration-based structural health monitoring," *Mechanical Systems and Signal Processing*, vol. 184, p. 109708, 2023.

[5] F. Di Nuzzo, D. Brunelli, T. Polonelli, and L. Benini, "Structural health monitoring system with narrowband iot and mems sensors," *IEEE Sensors Journal*, vol. 21, no. 14, pp. 16 371–16 380, 2021.

[6] H. X. Nguyen, S. Zhu, and M. Liu, "A survey on graph neural networks for microservice-based cloud applications," *Sensors*, vol. 22, no. 23, p. 9492, 2022.

[7] A. Burrello, A. Marchioni, D. Brunelli, S. Benatti, M. Mangia, and L. Benini, "Embedded streaming principal components analysis for network load reduction in structural health monitoring," *IEEE Internet of Things journal*, vol. 8, no. 6, pp. 4433–4447, 2020.

[8] Q. Chen, J. Cao, and Y. Xia, "Physics-enhanced pca for data compression in edge devices," *IEEE Transactions on Green Communications and Networking*, vol. 6, no. 3, pp. 1624–1634, 2022.

[9] F. Zonzini, M. Zauli, M. Mangia, N. Testoni, and L. De Marchi, "Model-assisted compressed sensing for vibration-based structural health monitoring," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 11, pp. 7338–7347, 2021.

[10] F. Zonzini, V. Dertimanis, E. Chatzi, and L. De Marchi, "System identification at the extreme edge for network load reduction in vibration-based monitoring," *IEEE Internet of Things Journal*, vol. 9, no. 20, pp. 20 467–20 478, 2022.

[11] E. N. Chatzi and C. Papadimitriou, *Identification methods for structural health monitoring*. Springer, 2016, vol. 567.

[12] J. Kim and J. P. Lynch, "Autonomous decentralized system identification by markov parameter estimation using distributed smart wireless sensor networks," *Journal of Engineering Mechanics*, vol. 138, no. 5, pp. 478–490, 2012.

[13] E. Reynders, "System identification methods for (operational) modal analysis: review and comparison," *Archives of Computational Methods in Engineering*, vol. 19, pp. 51–124, 2012.

[14] G. Ballard, J. Demmel, L. Grigori, M. Jacquelin, H. D. Nguyen, and E. Solomonik, "Reconstructing householder vectors from tall-skinny qr," in *2014 IEEE 28th International Parallel and Distributed Processing Symposium*. IEEE, 2014, pp. 1159–1170.

[15] G. Technologies. (2014) GreenWaves Technologies gap9 official description. [Online]. Available: https://greenwaves-technologies.com/gap9_processor/

[16] D. Rossi, F. Conti, M. Eggiman, A. Di Mauro, G. Tagliavini, S. Mach, M. Guermandi, A. Pullini, I. Loi, J. Chen *et al.*, "Vega: A ten-core soc for iot endnodes with dnn acceleration and cognitive wake-up from mram-based state-retentive sleep mode," *IEEE Journal of Solid-State Circuits*, vol. 57, no. 1, pp. 127–139, 2021.