Resource-Efficient FPGA-based Machine Learning Control for Active Structural Damping in Shock Environments

Trotter Roberts[1], Joud N. Satme[1], Puja Chowdhury[1], Austin R.J. Downey[1,2], and Jason D. Bakos [3]
University of South Carolina, Columbia, SC 29208
[1]Department of Mechanical Engineering
[2]Department of Civil and Environmental Engineering
[3]Department of Computer Science and Engineering

ABSTRACT

Maintaining structural integrity of electronic components in dynamic environments requires effective vibration control strategies, particularly in aerospace, automotive, and industrial applications where transient impacts and sustained oscillations can degrade performance and reliability. Traditional passive damping techniques provide limited flexibility to varying external forces, necessitating active control approaches for enhanced vibration suppression. This study evaluates the computational feasibility and performance trade-offs of Proportional-Derivative (PD), Linear Quadratic Gaussian (LQG), and Multi-Layer Perceptron (MLP) controllers for active vibration mitigation in a simulated cantilever beam system. A finite element model is used to simulate structural response under impact loading, while real-time implementation is conducted on an Field Deployable Gate Array (FPGA)-based control system. The PD and LQG controllers follow structured control frameworks, with the LQG approach incorporating optimal control and state estimation for improved stabilization. The MLP controller, trained on LQG-derived synthetic data, learns a control strategy without explicit system modeling. Performance metrics, including peak displacement, settling time, RMS displacement, control effort, and damping efficiency, provide insight into each controller's trade-offs. Additionally, FPGA resource utilization is analyzed to assess the computational overhead of deploying structured and learning-based controllers in embedded systems. Results highlight that while structured controllers maintain efficiency and predictable performance, the MLP controller demonstrates the potential for adaptive vibration suppression while using 22.5% fewer FPGA resources than the LQG controller in terms of total slices. This is achieved while maintaining performance comparable to the LQG controller, with a settling time of 1.36 s compared to 1.01 s for LQG, and significantly outperforming the PD controller, which has a settling time of 1.82 s. These findings emphasize that the choice of control strategy depends on application constraints, balancing computational efficiency with flexibility.

Keywords: vibration mitigation, piezoelectric actuators, multi-layer perceptron, real-time control, FPGA, dynamic environments, structural control, survivability

1. INTRODUCTION

Active structural control for vibration suppression is critical in mitigating the effects of dynamic loads in aerospace, defense, automotive, civil, and industrial applications. These sectors frequently operate under conditions where high-energy, short-duration impacts[?] induce structural stress and deformation, potentially compromising system integrity.[1] High-rate dynamic environments introduce non-stationarities, heavy disturbances, and uncertainties, which pose challenges for conventional control strategies.[2] Traditional passive damping techniques, such as tuned mass dampers and viscoelastic materials, are effective under steady-state conditions but lack flexibility to rapidly changing external forces.[3] To address these limitations, active control strategies have been widely adopted, providing the ability to adjust in real-time to suppress unwanted vibrations.[4]

Proportional-derivative (PD) controllers remain a widely used structured control approach, offering computational efficiency and ease of implementation in embedded systems. PD control has been successfully applied to aerospace structures, industrial machinery, and electronic assemblies, where fast stabilization and low computational demand are essential. However, PD controllers rely on fixed gain parameters and may struggle to adapt to nonlinear, high-impact environments, limiting their effectiveness in dynamic applications.[5]

The Linear Quadratic Gaussian (LQG) controller integrates optimal control with state estimation, improving performance in noisy and uncertain environments. By combining a Linear Quadratic Regulator (LQR) with a Kalman filter, LQG optimally balances performance and control effort while compensating for system disturbances.[6] This makes LQG well-suited for vibration suppression applications where system uncertainties, external disturbances, and sensor noise affect real-time control.[7] However, LQG's reliance on accurate system models limits flexibility in highly nonlinear or time-varying conditions.

The integration of machine learning techniques into active control systems introduces a promising alternative for handling complex, nonlinear dynamics.[8] Unlike traditional controllers that require explicit system models, ML-based controllers learn system behavior from data, making them well-suited for applications where governing dynamics are difficult to model accurately. Multi-Layer Perceptron (MLP) networks, in particular, have demonstrated strong function approximation capabilities, enabling them to predict and optimize control responses in real time. This adaptability makes ML-based controllers attractive for vibration suppression, especially in cases where system conditions vary over time.[9]

While ML-based controllers offer adaptability and learning capabilities, their deployment in real-time embedded systems presents computational challenges. Neural network-based control demands higher memory usage, increased computational complexity, and greater FPGA resource consumption compared to traditional feedback controllers.[10] To make ML-based control viable in resource-constrained environments, optimizations are essential in areas such as weight storage, inference latency, and numerical precision. This aligns with the principles of TinyML, where neural networks are tailored for efficient execution on low-power hardware, enabling real-time performance without excessive resource overhead.[11]

This study evaluates the FPGA resource utilization and control performance of PD, LQG, and MLP controllers for active vibration suppression of a simulated cantilever beam subjected to external impact forces. A finite element model (FEM) is used to simulate the beam's dynamic behavior, while all controllers are implemented on a FPGA to assess real-tme computational efficiency. FPGAs are chosen for their ability to provide low-latency, high-throughput digital processing, making them well-suited for real-time control applications that require deterministic execution and rapid response to dynamic loads.[12] Unlike general-purpose processors, FPGAs enable truly parallel processing of control algorithms,[13] optimizing computation-heavy tasks such as matrix operations in LQG[14] and neural network inference in MLP.[15] The primary contributions of this work are twofold. First, we propose a framework for implementing vibration mitigation controllers on FPGAs, demonstrating their feasibility for real-time structural control. Second, we show that the MLP controller achieves better resource efficiency than the LQG controller, challenging the assumption that learning-based methods are inherently more computationally demanding than traditional optimal control approaches. These findings provide insights into the resource requirements and trade-offs between classical and machine learning-based control strategies in FPGA-based implementations.

2. METHODOLOGY

This study models the dynamic behavior of a cantilever beam subjected to an external impact force using FEM[16] and explores active vibration control strategies. The focus is on evaluating the differences in computational requirements and implementation characteristics between a traditional PD controller, an optimal LQG controller, and a machine learning-based MLP controller. Properties for the beam are based on a slender steel design, shown in Table 1.

Table 1: The simulated cantilever beam dimensions and material properties.

width	thickness	length	density	Young's modulus
0.05 m	0.005 m	1.00 m	7850 kg/m^3	210 GPa

2.1 Finite Element Model

The cantilever beam is modeled using the Euler-Bernoulli beam theory, governed by the equation:

$$M\ddot{x}(t) + C\dot{x}(t) + Kx(t) = u(t) \tag{1}$$

where M is the mass matrix, C is the Rayleigh damping matrix, K is the stiffness matrix, $x(t)$ is the displacement vector, and $u(t)$ is the external force vector. The beam is discretized into $n = 50$ nodes, resulting in $2n = 100$ degrees of freedom (DOFs), as each node represents transverse displacement and rotation. The elemental mass and stiffness matrices are assembled into global matrices, and boundary conditions for a cantilever beam (fixed at one end) are imposed. Figure 1 depicts the continuous cantilever beam and the application points for external and control forces and the FEM discretization of the cantilever beam, highlighting node placement.

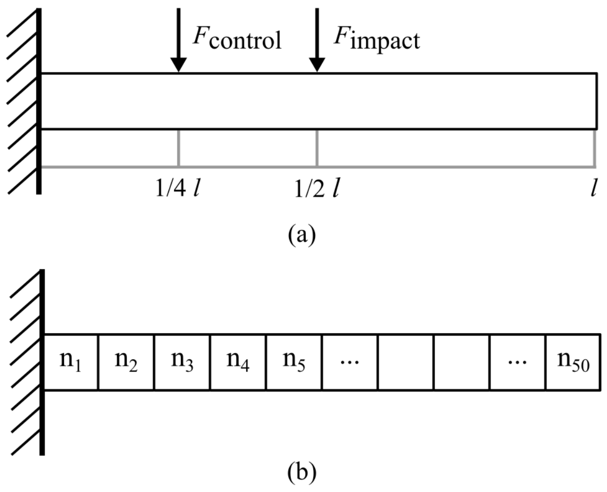

Figure 1: An illustration of: (a) the continuous cantilever beam showing where the impact and control forces are applied, and; (b) the discretized cantilever beam showing how each node is placed.

The Rayleigh damping matrix is defined as a linear combination of the mass and stiffness matrices:[17]

$$C = \alpha M + \beta K \tag{2}$$

where the coefficients $\alpha = 0.01$ and $\beta = 0.001$ are empirically selected to match the realistic damping properties of steel. Time integration of the beam dynamics is performed using the Newmark-beta method,[18] which iteratively updates displacements, velocities, and accelerations. The effective stiffness matrix, incorporating damping and inertia effects, is given by:

$$K_{\text{eff}} = K + \frac{\gamma}{\beta_n \Delta t} C + \frac{1}{\beta_n \Delta t^2} M \tag{3}$$

with Newmark-beta parameters $\beta_n = 0.25$ and $\gamma = 0.5$.

2.2 Control Strategies

Two control strategies are implemented to suppress vibrations caused by an impact force ($F_{\text{impact}} = 1000$ N) applied at the midpoint of the beam ($n/2$) at $t = 1.0$ s. Control forces are applied at a 1/4-length node ($n/4$) to minimize tip displacement.

2.2.1 Proportional-Derivative Control

The PD controller is a widely used feedback control technique due to its simplicity and effectiveness. It applies a control force based on displacement and velocity feedback at the control point. By tuning the proportional gain (k_p) and derivative gain (k_d), the system balances vibration suppression and control effort. In this case, the PD was tuned heuristically.[19] A properly tuned PD controller minimizes overshoot and provides a rapid response, though it lacks flexibility to system variations and external disturbances. The computational steps for the PD controller are outlined in Algorithm 1, and a flowchart representation is provided in Figure 2.

Algorithm 1 Proportional-Derivative (PD) Control

```
Initialize $k_\mathrm{p}$ and $k_\mathrm{d}$
for each time step $t$ do
    Compute error based on displacement and velocity:
        $$e_t = x_\mathrm{desired} - x_{c,t}, \quad \dot{e}_t = \dot{x}_\mathrm{desired} - v_{c,t} \tag{4}$$
    Compute PD control force:
        $$u_\mathrm{PD}(t) = k_\mathrm{p} e_t + k_\mathrm{d} \dot{e}_t \tag{5}$$
    Apply control force at the control node:
        $$u(t) = u_\mathrm{PD}(t) \tag{6}$$
    Compute system dynamics:
        $$M\ddot{x}(t) + C\dot{x}(t) + Kx(t) = u(t) \tag{1}$$
    Update displacement using Newmark-beta integration:
        $$K_\mathrm{eff} x_{t+1} = F_\mathrm{eff} \tag{7}$$
    Update acceleration:
        $$a_{t+1} = \frac{1}{\beta_n \Delta t^2}(x_{t+1} - x_t) - \frac{1}{\beta_n \Delta t} v_t - \left(\frac{0.5}{\beta_n} - 1\right) a_t \tag{8}$$
    Update velocity:
        $$v_{t+1} = v_t + \Delta t\left((1-\gamma)a_t + \gamma a_{t+1}\right) \tag{9}$$
    Store $x_{t+1}$ for analysis
end for
```

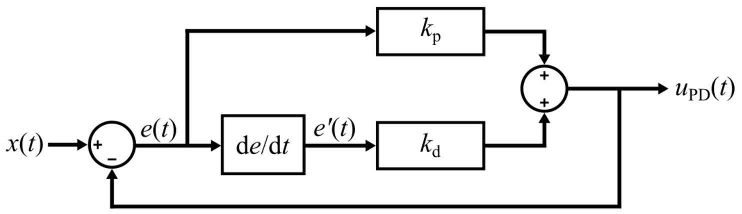

Figure 2: A flowchart of the PD controller used.

2.2.2 Linear Quadratic Gaussian Control

The LQG controller integrates optimal control and state estimation to achieve robust performance in dynamic systems.[6] It consists of a Linear Quadratic Regulator (LQR) for computing an optimal control force and a Kalman filter for estimating the system state. The controller balances performance and control effort while compensating for system noise and disturbances. The LQR determines the control input by solving an optimization problem that minimizes deviations from the desired state and excessive control effort. The Kalman filter continuously updates the state estimate using system measurements and model predictions. By integrating these two components, LQG provides improved stability and disturbance rejection compared to classical controllers. The computational steps for the LQG controller are outlined in Algorithm 2, and a flowchart representation is provided in Figure 3.

Algorithm 2 Linear Quadratic Gaussian (LQG) Control

Initialization:

Define system dynamics:

$$\dot{x}(t) = Ax(t) + Bu(t) + w(t) \tag{10}$$

Define measurement equation:

$$y(t) = Cx(t) + v(t) \tag{11}$$

Define cost function:

$$J = \int_0^\infty (x^\intercal Q x + u^\intercal R u)\, dt \tag{12}$$

Solve the continuous-time Riccati equation for LQR:

$$P = A^\intercal P + PA - PBR^{-1}B^\intercal P + Q = 0 \tag{13}$$

Compute LQR gain:

$$K = R^{-1}B^\intercal P \tag{14}$$

Solve the continuous-time Riccati equation for the Kalman filter:

$$S = AS + SA^\intercal - SC^\intercal V^{-1}CS + W = 0 \tag{15}$$

Compute Kalman gain:

$$L = SC^\intercal V^{-1} \tag{16}$$

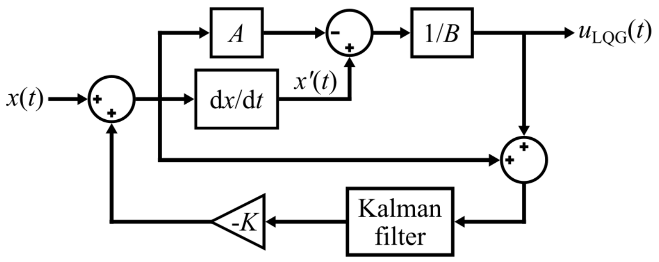

Figure 3: A flowchart of the LQG controller used.

Algorithm 2 (continued)

```
for each time step $t$ do
    Measure system output $y(t)$
    Update state estimate using Kalman filter:
        $\hat{x}(t+1) = A\hat{x}(t) + Bu(t) + L(y(t) - C\hat{x}(t))$        (17)
    Compute optimal LQG control input:
        $u_{\text{LQG}}(t) = -K\hat{x}(t)$        (18)
    Apply control force:
        $u(t) = u_{\text{LQG}}(t)$        (19)
    Compute system dynamics:
        $M\ddot{x}(t) + C\dot{x}(t) + Kx(t) = u(t)$        (1)
    Update displacement using Newmark-beta integration:
        $K_{\text{eff}}x_{t+1} = F_{\text{eff}}$        (7)
    Update acceleration:
        $a_{t+1} = \frac{1}{\beta_n \Delta t^2}(x_{t+1} - x_t) - \frac{1}{\beta_n \Delta t}v_t - \left(\frac{0.5}{\beta_n} - 1\right)a_t$        (8)
    Update velocity:
        $v_{t+1} = v_t + \Delta t\left((1-\gamma)a_t + \gamma a_{t+1}\right)$        (9)
    Store $x_{t+1}$ for analysis
end for
```

2.2.3 Multi-Layer Perceptron Control

The MLP controller leverages artificial neural networks to determine the control force without requiring an explicit system model. Instead, it learns input-output relationships from training data, allowing it to adapt to varying conditions and nonlinearities.[16] The network consists of an input layer representing displacement, velocity, and acceleration; two hidden layers with 64 neurons each, activated by ReLU; and an output layer predicting the control force.

The MLP is trained using real control data collected from the LQG controller during simulation. At each time step, the LQG controller computes a control force based on the system state, and these state-control pairs are stored as training data. The training process minimizes the difference between MLP-predicted forces and the recorded LQG control forces, allowing the neural network to learn an approximate control policy. Once trained, the MLP controller directly predicts control forces based on the current system state, and the system dynamics are updated using the same numerical integration method as the PD and LQG controllers. The full computational process is outlined in Algorithm 3, and the corresponding neural network architecture is shown in Figure 4.

Algorithm 3 Multi-Layer Perceptron (MLP) Control

```
Define neural network architecture:
Input: $[x_c, v_c, a_c]$
Two hidden layers with ReLU activation
Output: $u_{\text{MLP}}$
Train MLP model using PLQG controller as reference:
```

Algorithm 3 (continued)

```
for each training sample $i$ do
    Generate synthetic data $(x, v, a, u_{\mathrm{PD}})$
    Compute MLP output:
        $u_{\mathrm{MLP}} = W_3\sigma(W_2\sigma(W_1[x, \dot{x}, \ddot{x}] + b_1) + b_2) + b_3$    (20)
    Minimize loss function:
        $L = \frac{1}{N}\sum_{i=1}^{N}(u_{\mathrm{MLP},i} - u_{\mathrm{LQG},i})^2$    (21)
end for
for each time step $t$ do
    Extract input state:
        $\mathbf{s}_t = [x_{c,t}, v_{c,t}, a_{c,t}]$    (22)
    Compute MLP control force:
        $u_t = \mathrm{MLP}(\mathbf{s}_t)$    (23)
    Apply control force at the control node:
        $u(t) = u_{\mathrm{MLP}}(t)$    (24)
    Compute system dynamics:
        $M\ddot{x}(t) + C\dot{x}(t) + Kx(t) = u(t)$    (1)
    Update displacement using Newmark-beta integration:
        $K_{\mathrm{eff}}x_{t+1} = F_{\mathrm{eff}}$    (7)
    Update acceleration:
        $a_{t+1} = \frac{1}{\beta_n\Delta t^2}(x_{t+1} - x_t) - \frac{1}{\beta_n\Delta t}v_t - \left(\frac{0.5}{\beta_n} - 1\right)a_t$    (8)
    Update velocity:
        $v_{t+1} = v_t + \Delta t\,((1-\gamma)a_t + \gamma a_{t+1})$    (9)
    Store $x_{t+1}$ for analysis
end for
```

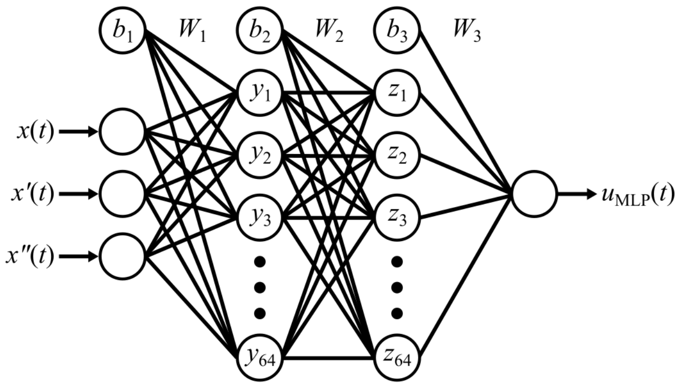

Figure 4: A flowchart of the MLP controller used.

The training progress of the MLP controller is evaluated using a learning curve, which tracks the mean squared error (MSE) loss function over training epochs. Figure 5 presents the loss curve, illustrating how quickly and effectively the network converges. The initial high loss values indicate significant errors in early iterations, which gradually decrease as the model refines its internal parameters. The loss curve is used in the figure to highlight the overall trend by filtering out minor fluctuations due to stochastic gradient descent. A rapid reduction in loss signifies efficient learning, while a plateau suggests convergence to an optimal or near-optimal solution. The final loss value, typically falling between 10 and 20 MSE, indicates the MLP's ability to approximate the LQG control forces accurately, demonstrating that the neural network successfully learns a control policy based on the recorded state-control pairs.

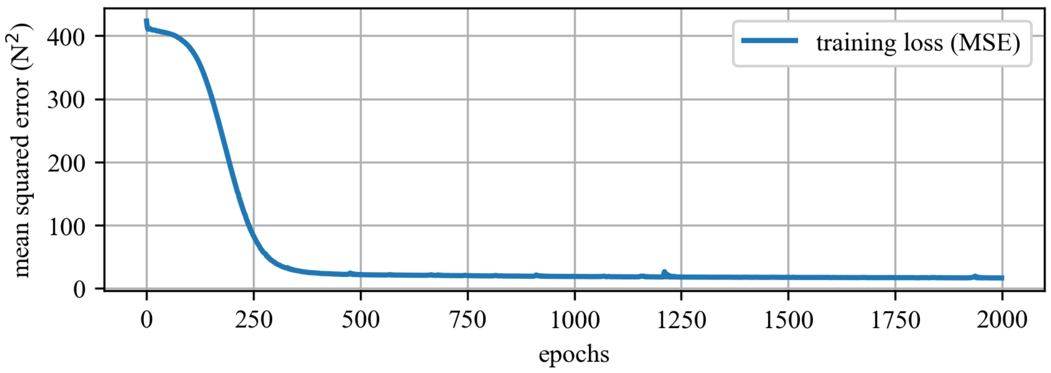

Figure 5: The learning curve for the MLP controller, showing MSE loss over training epochs.

To determine the optimal number of neurons in each hidden layer of the MLP controller, an optimization analysis was performed by varying the number of neurons and evaluating the mean squared error (MSE) for each configuration. Figure 6 illustrates the relationship between MSE and the number of neurons in each hidden layer. The search indicated that increasing the number of neurons initially reduced MSE, but diminishing returns were observed beyond 32 neurons, where additional neurons led to minimal improvement while increasing FPGA resource usage. A best-fit curve was obtained using an exponential function:

$$\text{MSE} = ae^{bN} + c \tag{25}$$

where N is the number of neurons per hidden layer and parameters a, b, and c were determined using nonlinear curve fitting. Based on this analysis, we selected a 32-neuron architecture for FPGA implementation to balance accuracy and efficiency.

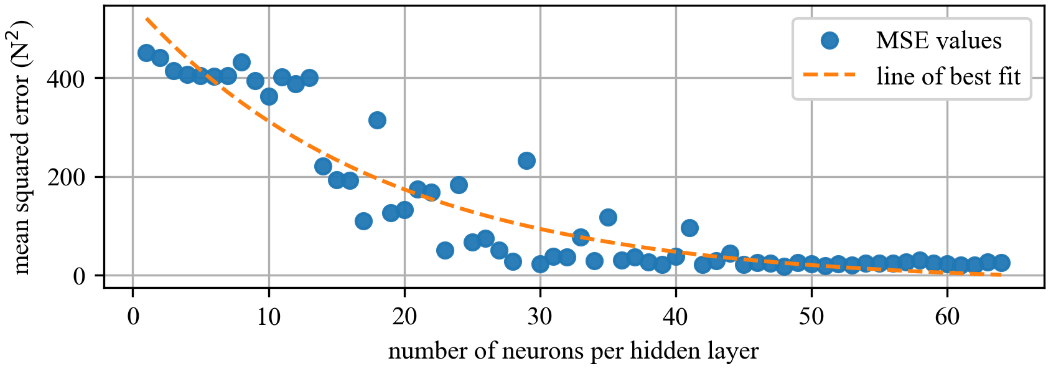

Figure 6: Optimization results showing the relationship between the number of neurons in each hidden layer and MSE.

2.3 FPGA-Based Implementation

The controllers were deployed on a National Instruments cRIO-9035, a reconfigurable embedded control and acquisition system. The cRIO-9035 features an integrated Xilinx Kintex-7 FPGA, which provides real-time processing capabilities for executing control algorithms. The primary goal of this implementation was to analyze hardware resource utilization, rather than the specific performance of the control forces or displacements.[20]

The FPGA-based implementation was structured to process simulated beam displacement data and generate control force outputs, but the focus remained on evaluating the computational requirements of each controller. Figure 7 illustrates how the FPGA was set up. All computations were performed using single-precision floating-point (SGL) arithmetic, ensuring numerical precision while maintaining efficient processing on the FPGA. The MLP network's weights and biases were stored in Block RAM (BRAM) to facilitate fast retrieval during inference, minimizing memory access delays. To manage data transfer efficiently, FIFO buffers were used between processing elements, optimizing pipelining and reducing latency. The code for this paper is made available through a public repository.[21]

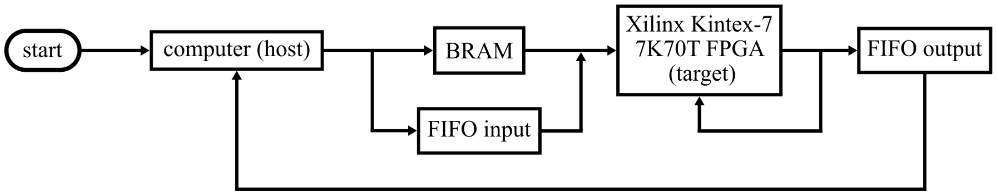

Figure 7: A flowchart of the general FPGA implementation.

3. RESULTS AND ANALYSIS

The tip displacement histories for both uncontrolled and controlled cases are shown in Figure 8, with peak displacement values summarized in Table 2. All controlled cases reduce peak displacement relative to the uncontrolled beam. The LQG controller achieves the largest reduction in peak displacement. Peak displacement is a critical parameter in structural applications, as excessive deflections contribute to material fatigue and potential structural damage. The LQG controller, by solving an optimization problem at each time step, minimizes peak displacement efficiently while keeping control effort low. The MLP controller, trained on LQG-generated data, exhibits a similar response The PD controller, in contrast, follows a direct feedback approach that reduces displacement and performs the worst of the three.

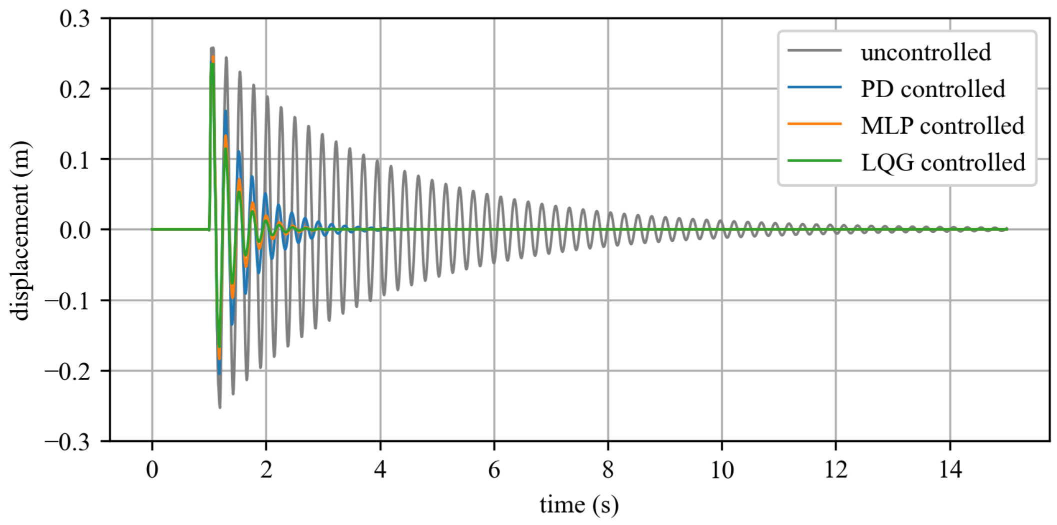

Figure 8: The tip displacement of both simulated uncontrolled and controlled beams.

Table 2: The peak displacement for the simulated controlled beam responses.

uncontrolled	PD controlled	LQG controlled	MLP controlled
0.2575 m	0.2435 m	0.2343 m	0.2451 m

Settling time measures how quickly vibrations decay, making it an important metric for real-time applications. Faster stabilization allows for quicker system recovery after an impact. As shown in Table 3, all controllers significantly reduce settling time compared to the uncontrolled case. The LQG controller achieves the fastest stabilization, followed by the MLP controller, which learns from LQG control strategies and closely approximates its performance. The PD controller, while effective, takes slightly longer to settle compared to the MLP controller.

Table 3: The settling times for both simulated uncontrolled and controlled beam responses.

uncontrolled	PD controlled	LQG controlled	MLP controlled
8.74 s	1.82 s	1.01 s	1.36 s

Control effort quantifies the total force applied by each controller, making it a key factor in power efficiency and actuator longevity. Table 4 shows that the LQG controller applies the least control effort while still effectively suppressing vibrations, demonstrating its efficiency in optimizing force application. The PD controller requires the highest control effort, followed by the MLP controller, while the LQG controller is the most efficient in minimizing control effort. This suggests that structured controllers like LQG are preferable in energy-sensitive applications, whereas learning-based controllers such as MLP may provide a balance between flexibility and efficiency.

Table 4: The control efforts for the simulated controlled beam responses.

PD controlled	LQG controlled	MLP controlled
0.91 N	0.53 N	0.70 N

RMS displacement provides a measure of overall vibration suppression over time. Unlike peak displacement, which captures only the largest deviation, RMS displacement accounts for sustained oscillations, making it a key indicator of long-term stability in dynamic environments. Reducing RMS displacement is particularly important for structures subject to continuous loading, as lower cumulative motion helps prevent material fatigue and unwanted resonant effects. Table 5 summarizes the RMS displacement values for each control approach. The results show that all controlled cases achieve a reduction in RMS displacement compared to the uncontrolled beam.

Table 5: The RMS displacements for both simulated uncontrolled and controlled beam responses.

uncontrolled	PD controlled	LQG controlled	MLP controlled
0.0591 m	0.0266 m	0.0204 m	0.0234 m

Damping efficiency evaluates how effectively each controller suppresses vibrations relative to the control effort applied. Efficient damping is particularly relevant in energy-constrained environments, where reducing unnecessary actuator activity can reduce power consumption. Table 6 presents the damping efficiency values. Results indicate that all active control strategies improve damping efficiency compared to the uncontrolled case.

Table 6: The damping efficiency for the simulated controlled beam responses.

PD controlled	LQG controlled	MLP controlled
3.57 %	7.32 %	5.09 %

Time-weighted damping efficiency evaluates vibration suppression and stabilization speed, crucial for systems needing rapid response. Table 7 shows variations across control methods. The LQG controller achieves the highest efficiency, stabilizing quickly with minimal effort. The MLP controller balances suppression and speed, while the PD controller, though fast, is less efficient in control effort. These results highlight trade-offs between structured and learning-based control, with effectiveness depending on response speed, force allocation, and flexibility.

Table 7: The time-weighted damping efficiency for the simulated controlled beam responses.

PD controlled	LQG controlled	MLP controlled
1.27 %	3.64 %	2.25 %

The selected performance metrics provide insight into the trade-offs between structured and learning-based control strategies for active vibration suppression. The LQG controller demonstrates efficiency in minimizing both control effort and settling time, making it ideal for applications that prioritize rapid stabilization with minimal energy use. The PD controller, while computationally simple, does not explicitly optimize force application but still achieves reasonable vibration suppression. The MLP controller, trained on LQG-generated data, offers flexibility and achieves similar performance but at the cost of increased control effort.

3.1 FPGA Resource Utilization

The FPGA implementation of the PD, LQG, and MLP controllers was analyzed to assess logic utilization, memory consumption, and computational efficiency. To determine the actual utilization of each controller, the resource usage of the FIFO-based data transfer, implemented separately, was subtracted from the total utilization. Rather than serving as a direct performance comparison, these metrics highlight the distinct computational trade-offs associated with each control strategy. Table 8 summarizes the FPGA resource utilization for each controller and Figure 9 adding a direct comparison, both provide insight into how the controllers' computational structures impact hardware deployment. The important metric here is 'total slices,' which represent the fundamental configurable logic blocks (CLBs) used in the FPGA, each containing Look-Up Tables (LUTs), flip-flops, and multiplexers for implementing digital logic. The MLP controller utilizes 22.5% fewer FPGA resources than the LQG controller in terms of total slices, highlighting its improved resource efficiency while maintaining comparable performance.

Table 8: FPGA resource utilization for PD, LQG, and MLP controllers.

resource	PD controller	LQG controller	MLP controller
total slices	5.3% (539/10250)	36.1% (3699/10250)	28.0% (2867/10250)
slice registers	1.0% (841/82000)	17.4% (14259/82000)	7.9% (6439/82000)
slice LUTs	3.6% (1482/41000)	15.1% (6208/41000)	17.5% (7165/41000)
block RAMs	0.0% (0/135)	1.5% (2/135)	2.2% (3/135)
DSP48s	0.8% (2/240)	1.3% (3/240)	2.9% (7/240)

The PD controller exhibits the lowest FPGA resource usage, relying on minimal computational logic for proportional-derivative control. The LQG controller requires more resources due to its combined LQR optimization and Kalman filtering. The MLP controller, despite its neural network complexity, remains more resource-efficient than LQG, particularly in slice utilization. As shown in Table 8, the MLP controller demands the most Block RAM for storing learned weights, while the LQG controller requires additional memory for state covariance. DSP usage follows a similar trend, with MLP utilizing the most due to matrix operations, LQG requiring moderate DSP resources for filtering, and PD consuming the least.

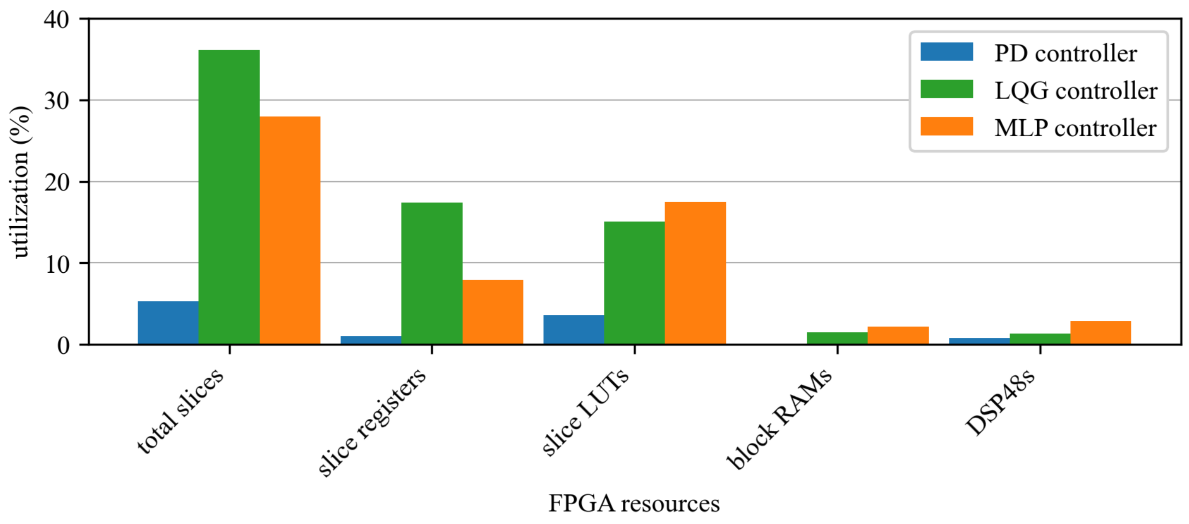

Figure 9: A side-by-side comparison of controller FPGA utilization.

4. CONCLUSION

This study assessed the computational and control trade-offs between PD, LQG, and MLP controllers for FPGA-based vibration suppression. Performance metrics such as peak displacement, settling time, RMS displacement, control effort, and damping efficiency were evaluated alongside FPGA resource utilization to determine real-time feasibility. Each approach demonstrated distinct advantages: PD prioritized simplicity and efficiency, LQG balanced control effort with performance, and MLP achieved greater resource efficiency than LQG. These findings highlight the trade-offs between structured and learning-based control, demonstrating that MLP can offer competitive resource utilization when optimized for FPGA deployment.

While PD and LQG controllers provide predictable, model-based responses suited for efficiency and stability, the MLP controller offers flexibility with potential for further optimization. Techniques such as network pruning, weight quantization, and FPGA-specific implementations could improve its real-time feasibility, making it a strong candidate for adaptive control in resource-constrained environments. Future work will explore hybrid control strategies that integrate structured control with learning-based adaptability while validating the MLP controller on experimental hardware.

5. ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation grant numbers 1937535, 1956071, 2234921, and 2237696, with additional support from the Air Force Office of Scientific Research (AFOSR) through award no. FA9550-21-1-0083. Any opinions, findings conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation, or the United States Air Force.

REFERENCES

1. Wani, Z. R., Tantray, M., Noroozinejad Farsangi, E., Nikitas, N., Noori, M., Samali, B., and Yang, T., "A critical review on control strategies for structural vibration control," *Annual Reviews in Control* **54**, 103–124 (2022).
2. Dodson, J., Downey, A., Laflamme, S., Todd, M. D., Moura, A. G., Wang, Y., Mao, Z., Avitabile, P., and Blasch, E., [*High-Rate Structural Health Monitoring and Prognostics: An Overview*], 213–217, Springer International Publishing (Oct. 2021).
3. Fisco, N. and Adeli, H., "Smart structures: Part I—active and semi-active control," *Scientia Iranica* **18**, 275–284 (June 2011).

4. El-Khoury, O. and Adeli, H., "Recent advances on vibration control of structures under dynamic loading," *Archives of Computational Methods in Engineering* **20**, 353–360 (Oct. 2013).
5. Downey, A. and Micheli, L., "Vibration mechanics: A practical introduction for mechanical, civil, and aerospace engineers," (2024).
6. Athans, M., "The role and use of the stochastic linear-quadratic-gaussian problem in control system design," *IEEE Transactions on Automatic Control* **16**, 529–552 (Dec. 1971).
7. Chowdhury, P., Downey, A. R. J., Bakos, J. D., Laflamme, S., and Hu, C., "Hardware implementation of nonstationary structural dynamics forecasting," in [*Active and Passive Smart Structures and Integrated Systems XVII*], Tol, S., Nouh, M. A., Shahab, S., Yang, J., and Huang, G., eds., 67, SPIE (Apr. 2023).
8. Jiang, Z.-P., [*Learning-Based Control*], no. v.22 in Foundations and Trends® in Systems and Control Ser., Now Publishers, Norwell, MA (2020). Description based on publisher supplied metadata and other sources.
9. Zaparoli Cunha, B., Droz, C., Zine, A.-M., Foulard, S., and Ichchou, M., "A review of machine learning methods applied to structural dynamics and vibroacoustic," *Mechanical Systems and Signal Processing* **200**, 110535 (Oct. 2023).
10. Shawahna, A., Sait, S. M., and El-Maleh, A., "Fpga-based accelerators of deep learning networks for learning and classification: A review," *IEEE Access* **7**, 7823–7859 (2019).
11. Iodice, G. M., [*TinyML Cookbook*], Packt Publishing Limited, Birmingham, 1. ed. (2022).
12. Leva, A. and Piroddi, L., "Fpga-based implementation of high-speed active noise and vibration controllers," *Control Engineering Practice* **19**, 798–808 (Aug. 2011).
13. Kastner, R., Matai, J., and Neuendorffer, S., "Parallel programming for fpgas," (May 2018).
14. Cauwels, M., Zambreno, J., and Jones, P. H., "Hw/sw configurable lqg controller using a sequential discrete kalman filter," in [*2018 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*], 1–8, IEEE (Dec. 2018).
15. Zhang, H., Zuo, J., Zheng, H., Liu, S., Luo, M., and Zhao, M., "Implementing neural networks on non-volatile fpgas with reprogramming," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **43**, 3961–3972 (Nov. 2024).
16. Petyt, M., [*Introduction to Finite Element Vibration Analysis*], Cambridge University Press (Aug. 2010).
17. Alarcon, A., *Modeling of damping in linear dynamics.* Code Aster (June 2016).
18. Gavin, H. P., "Numerical integration in structural dynamics." Course Note (Aug. 2020).
19. Hover, F. S. and Triantafyllou, M. S., [*System Design for Uncertainty*], ch. 11, 11.8, LibreTexts (Jan. 2024).
20. Popa, B., Selișteanu, D., and Popescu, I. M., [*Real-Time Parallel Processing of Vibration Signals Using FPGA Technology*], 236–257, Springer International Publishing (Aug. 2022).
21. Roberts, T., Satme, J., and Downey, A., "Paper 2025 resource-efficient FPGA-based machine learning control for active structural damping in shock environments." GitHub (2025).