Neural Network Model-Predictive Control for CHB Converters With FPGA Implementation

Francesco Simonetti, *Student Member, IEEE*, Alessandro D'Innocenzo, *Member, IEEE*, and Carlo Cecati, *Fellow, IEEE*

***Abstract*—Finite control set model-predictive control appears an interesting and effective control technique for cascaded H-bridge converters but, because of its computational complexity, becomes impractical when the number of levels of the converter increases. This article proposes a neural-network-based approach capable of overcoming the computational burden of conventional predictive control algorithms. The proposed control is, then, applied to a cascaded H-bridge static synchronous compensator using a field-programmable gate array and tested via hardware in the loop. Results and analysis demonstrate that the optimal control of multilevel converters with many levels can be obtained with low computational effort.**

***Index Terms*—Model-predictive control (MPC), multilevel converters (MLCs), neural networks (NNs), power converters, static synchronous compensator (STATCOM).**

I. Introduction

CASCADED H-bridge multilevel converters (CHB-MLCs) are characterized by well-known advantages, such as modularity, low dv/dt, operations at low switching frequency, and, in medium-voltage systems, the possibility of avoiding step-up or step-down transformers [1], [2], [3]. During the past few years, the model-predictive control (MPC) strategy has been attracting increasing attention from researchers and practitioners also in the field of power converters [4], [5]. The most popular declination of MPC is the so-called finite control set model-predictive control (FCS-MPC), where the control problem is formulated considering the switching combinations of the converter as a finite and countable set of inputs [6], [7]. The possibility of directly controlling the switching signals avoids the use of a modulator, provides a fast dynamic response, and limits switching frequency, thus ensuring low losses for the converter.

Manuscript received 19 January 2022; revised 13 September 2022 and 29 November 2022; accepted 26 December 2022. Date of publication 3 January 2023; date of current version 24 July 2023. This work was supported in part by the Italian Ministry of University and Research through PRIN2017 project under Grant 2017MS9F49 and in part by DigiPower srl, L'Aquila, Italy, through "HORIZON 2020" PON I&C2014-2020 program under Grant F/050220/X32CUP B18I15000100008. Paper no. TII-22-0298. *(Corresponding author: Francesco Simonetti.)*

The authors are with the Department of Engineering and Information Science and Mathematics, University of L'Aquila, 67100 L'Aquila, Italy (e-mail: francesco.simonetti1@graduate.univaq.it; alessandro.dinnocenzo@univaq.it; carlo.cecati@univaq.it).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TII.2023.3233973.

Digital Object Identifier 10.1109/TII.2023.3233973

FCS-MPC is also popular for its simplicity since it just searches for the best switching pattern driving the converter [8]. However, in MLCs, computational complexity grows exponentially with the number of levels of the converter. Therefore, calculating the optimal input for the next sampling step becomes challenging [9]. Significant efforts can be found in the literature to improve the efficiency of FCS-MPC algorithm to make it implementable in a few tenth microseconds. One possible approach is reducing the search space within a subset of all the possible inputs. However, with this approach, only a suboptimal solution can be found with the existing real-time constraints [10], [11], [12]. Nasiri et al. [13] and Ni and Narimani [14] proposed control strategies that explicitly exploit system's dynamics. However, they do not compute the control solving an optimization problem. In [15], the optimization problem is solved by employing a sphere decoding algorithm. However, it requires the determination of an initial radius and involves several iterations.

An interesting case is the control of the CHB-MLC static synchronous compensator (CHB-STATCOM), which is one of the most advanced flexible alternative current transmission systems and is capable of regulating and stabilizing the grid voltage through the control of the reactive power. STATCOMs are commonly used in the stabilization of medium-voltage ac lines; hence, they take advance from multilevel topologies with a large number of levels, which makes their control with the FCS-MPC strategy challenging. Because of the high computational complexity and nonlinearity, a common solution divides the overall optimization problem into subproblems that are solved separately [16], [17], [18]. For instance, Zhang et al. [16] proposed first solving the current control problem and then ensuring voltage balancing. The resulting computational complexity is polynomial. Zhang et al. [19] proposed a branch and bound strategy to further reduce the computational complexity of the current controller, obtaining an algorithm with linear complexity. However, the number of computations is still high.

Owing to the large number of signals necessary to drive their power switches, MLCs usually employ field-programmable gate arrays (FPGA)s because they make available a large amount of input/output pins necessary for driving the signals of the CHB converter. Moreover, they are used for acquiring inputs, for the modulation algorithm, and for achieving the highest computational speed [20], [21], [22]. Zhang et al. [20] implemented the polynomial-level FCS algorithm in [19] by employing a DSP for the main calculations of the control law, accompanied by an

FPGA for reading the analog input signals, for sending the gate signals to the switches, and for hardware acceleration of most expensive parts of algorithm computation. Owing to the large number of computations needed by an FCS-MPC approach, it is a good solution to implement the overall control law on the FPGA [23], [24], [25], [26].

In recent years, machine learning (ML) techniques have been widely used in many engineering fields. One popular application in the automatic control field is to approximate a control law that is too computationally expensive to be implemented in real time. With offline computations, the ML procedure computes a nonlinear function that embeds the optimization solver. This function can be used in online implementation as a replacement for the original controller. In [27], a shallow neural network (NN) is trained to learn the MPC law for a two-level inverter for different circuit parameters and loads. In [28], the same strategy is applied to reduce the computational cost of a two-level inverter when considering horizons equal to one, two, and three. In [29], horizons 1 and 2 are evaluated on a three-level neutral point clamped topology. In [30], an NN is used for a flying capacitor multilevel topology. A comparison of timings and space complexity of the algorithm is carried out, and the number of output neurons linearly depends on the number of levels. In [31], a comparison between classification and regression NNs for modular MLCs is presented, where regression turns out to be more effective than classification. Timings are compared for predictive horizons of lengths one and two. In [32], different ML techniques are compared for approximating MPC on a CHB inverter. In a previous work [33], a preliminary methodology was proposed for deriving an NN capable of learning the optimal FCS-MPC. Owing to the low computational burden and the hardware acceleration, it is not necessary to consider the step ahead delay, and the control input is actuated in the same sampling interval.

The manuscript presents: 1) a deep study of the performance of the NN-MPC compared to the FCS-MPC on both the steady-state and transient conditions; 2) the generalization of the algorithm for different numbers of levels and prediction horizons via simulative results and the analysis of the performance when weighting coefficients and number of neurons change; 3) the description at the register transfer level (RTL) of the experimental implementation of the NN-MPC on the FPGA platform; 4) the evaluation of clock cycles needed for the algorithms execution and the actual time spent in computations on a Cyclone V FPGA on DE-10 Nano board; and 5) the analysis of the impact of the computation delay on the controller performance via hardware-in-the-loop simulation by employing the FPGA and the Simulink model.

The rest of this article is organized as follows. In Section II, the FCS-MPC strategy for a CHB inverter is discussed, and the NN-MPC approach is derived. In Section III, the CHB-STATCOM mathematical model is described. In Section IV, the FCS formulation for a CHB-STATCOM is presented. In Section V, the simulation results are discussed. In Section VI, the RTL implementation of the two algorithms is described, the number of computations is derived, and the HIL comparison is presented. Finally, Section VII concludes this article.

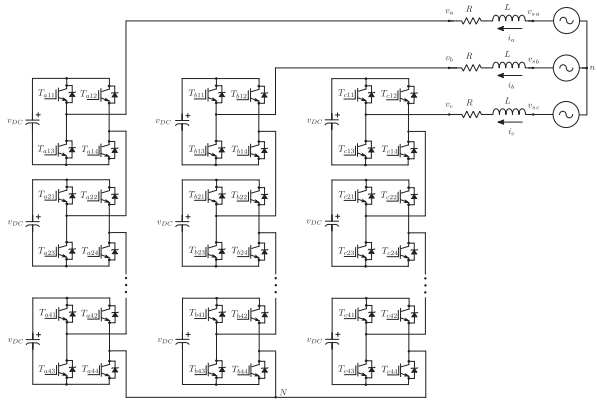

Fig. 1. Cascaded H-bridge inverter.

II. MPC of a Cascaded H-Bridge Inverter

A. Finite Control Set Model-Predictive Control

Consider a general CHB inverter composed of N H-bridges per phase as in Fig. 1. The currents are governed by Kirchhoff's laws

$$L\frac{di_{a,b,c}}{dt} = v_{s(a,b,c)} - Ri_{a,b,c} - v_{a,b,c} \tag{1}$$

where $i_{a,b,c}$ are the three-phase currents, $v_{a,b,c}$ are the voltages at the inverter output, $v_{s(a,b,c)}$ are the voltages at the load side, and R and L are the resistance and the inductance of the load, respectively. Using the Clarke transformation, the $\alpha\beta$ reference frame currents are given by

$$L\frac{di_{\alpha,\beta}}{dt} = v_{s(\alpha,\beta)} - Ri_{\alpha,\beta} - v_{\alpha,\beta}. \tag{2}$$

Previous equations, discretized via Euler approximation, become

$$i_{\alpha,\beta}(k+1) = i_{\alpha,\beta}(k) + \frac{T}{L}\left(-Ri_{\alpha\beta}(k) + v_{s(\alpha,\beta)}(k) - v_{\alpha,\beta}(k)\right) \tag{3}$$

where T is the sampling interval and k is the generic sampling instant.

In the FCS-MPC, the state in the next sampling instant is predicted using the dynamical model of the system. The control law calculations consist of the minimization of an optimization function, which generally includes two weighing terms: one for the deviation of the state from a reference value and the other for the effort of the controller. The FCS current control aims to solve the following minimization problem:

$$\min_{S_{\alpha,\beta}(k)} \sum_{p=1}^{h} \Big(\left\| i^{\text{ref}}_{\alpha,\beta}(k+p) - i_{\alpha,\beta}(k+p) \right\|_{Q_i} + \left\| S_{\alpha,\beta}(k+p-2) - S_{\alpha,\beta}(k+p-1) \right\|_{Q_s} \Big)$$
$$\text{s.t.} \quad (1)$$
$$S_{\alpha,\beta}(k) \in V_{\alpha,\beta} \tag{4}$$

where the first term is the weighted norm of the error between the reference current and the predicted current at time $k+p$, while the second term weights the variation of the applied switching vector to limit the switching loss of the converter. The weighting matrices can be expressed as $Q_i = w_i \times I$ and $Q_s = w_s \times I$, where I is the 2×2 identity matrix and w_i and w_s are scalar numbers. The prediction horizon h determines the number of future predictions to be computed, and the optimal input $S_{\alpha,\beta}(k)$ is searched within the set of all the possible switching combinations $V_{\alpha,\beta}$.

With the FCS approach, the solutions are reached by computing the cost for each possible vector and, then, its minimum. The prediction is computed making an explicit use of the model in (1) imposing the CHB output $v_{s(\alpha,\beta)}(k) = V_{\text{DC}}\, S_{\alpha,\beta}(k)$, where V_{DC} is the nominal dc-link voltage. The number of possible switching vectors is $(12 \times N^2 + 6 \times N + 1)^h$, and the exhaustive search algorithm is quadratic with respect to the number of levels and exponential with respect to the prediction horizon.

B. Neural Network Model-Predictive Control

In MLCs, the complexity of the controller becomes challenging when the number of levels becomes relevant, as occurs in medium-voltage systems. To overcome this difficulty, a novel approach is proposed to solve the optimal current control problem via an NN approximation, which consists of two distinct steps: in the first step, the CHB is simulated and the optimal vectors $S_{\alpha\beta}(k)$ are computed solving (4) for all the collected configurations of $i^{\text{ref}}_{\alpha,\beta}(k), i_{\alpha,\beta}(k), v_{s(\alpha,\beta)}(k), S_{\alpha,\beta}(k-1)$. Further data are collected by solving the control problem for random configurations. In the second step, the overall dataset is used to train an NN to obtain optimal current control inputs. A shallow NN is employed, using the Bayesian regularization backpropagation to minimize the sum squared error. The NN has eight input and two output variables; therefore, the input and output neurons are eight and two, respectively.

It is proposed to use η hidden neurons with a hyperbolic tangent activation function and a simple linear one for the output layer. Note that the computational complexity of the NN does not depend on the number of levels and prediction horizons and is only given by the adopted architecture. For the considered simple NN, $8 \times \eta + \eta \times 2$ sums and multiplications and η activation function computations are needed. This approach brings a closed form of the approximated solution of the optimal problem, which can be quickly calculated in real time. Since it is not possible to obtain a feasible result by simply rounding the $S_{\alpha\beta}$ vector, the result is transformed into a reference frame, in which the α-axis is delayed by $\pi/3$, obtaining the $S^{\pi/3}_{\alpha\beta}$ vector

$$S^{\pi/3}_{\alpha\beta} = \sqrt{3}\begin{bmatrix} \cos\left(-\frac{\pi}{3}\right) & \sin\left(-\frac{\pi}{3}\right) \\ 0 & 1 \end{bmatrix} S_{\alpha\beta} = \begin{bmatrix} \frac{3}{2} & -\frac{\sqrt{3}}{2} \\ 0 & \sqrt{3} \end{bmatrix} S_{\alpha\beta} \quad .$$

Then, the feasible points are arranged on a grid with orthogonal axes. The transformation also provides a scale factor in order to make them integer values. In this way, the feasible switching vector is obtained by rounding $S^{\pi/3}_{\alpha\beta}$. Hence, the inverse transformation into the abc frame leads to

$$S_{abc} = \begin{bmatrix} 1 & 0 \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} \\ -\frac{1}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} S_{\alpha\beta} = \begin{bmatrix} 1 & 0 \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} \\ -\frac{1}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} \begin{bmatrix} \frac{2}{3} & \frac{1}{3} \\ 0 & \frac{1}{\sqrt{3}} \end{bmatrix} S^{\pi/3}_{\alpha\beta} = \begin{bmatrix} \frac{2}{3} & \frac{1}{3} \\ -\frac{1}{3} & \frac{1}{3} \\ -\frac{1}{3} & -\frac{2}{3} \end{bmatrix} S^{\pi/3}_{\alpha\beta}.$$

The NN, including the reference transformation, is shown in Fig. 2.

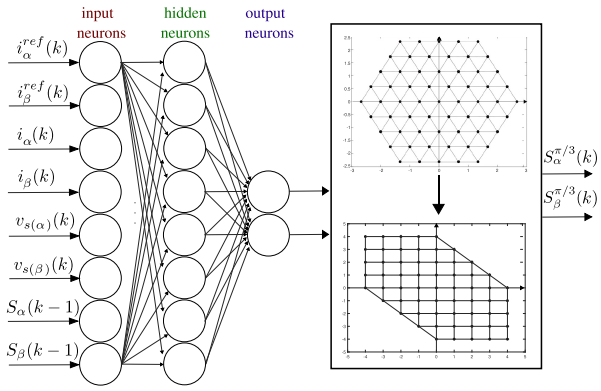

Fig. 2. NN current control with $\eta = 8$ hidden neurons.

III. CHB-STATCOM Model

The proposed approach is tested using a CHB-STATCOM composed of an inverter with N H-bridges per phase, each one including a dc-link capacitor C. The converter is connected with the grid through a series inductor L, as shown in Fig. 3. Each H-bridge can output positive, negative, or zero voltage, whose rated value is V_{DC}.

According to this, the system can be modeled considering the inputs of the system as discrete variables $s_{(a,b,c)i} \in \{-1, 0, 1\}$, where $i = 1, \ldots, N$ indicates that the H-bridge i of a, b, c phases is supplying negative, zero, or positive voltage. The total output voltage of the STATCOM is the sum of the individual cells' output voltages

$$v_{a,b,c} = \sum_{i=1}^{N} s_{(a,b,c)i}\, v_{\text{DC}(a,b,c)i} \simeq \sum_{i=1}^{N} s_{(a,b,c)i} V_{\text{DC}} \qquad (5)$$

where $v_{\text{DC}(a,b,c)i}$ is the dc-link voltage of the ith cell of the corresponding phase. Since the inverter is sized to have a small ripple voltage on the dc side, usually $v_{\text{DC}(a,b,c)i}$ is assumed to be equal to V_{DC}. The dynamic model of the CHB-STATCOM is

$$L\frac{di_{a,b,c}}{dt} = v_{s(a,b,c)} - Ri_{a,b,c} - v_{a,b,c}$$

$$C\frac{dv_{\text{DC}(a,b,c)i}}{dt} = s_{(a,b,c)i} i_{a,b,c}, \quad i = 1, \ldots, N \qquad (6)$$

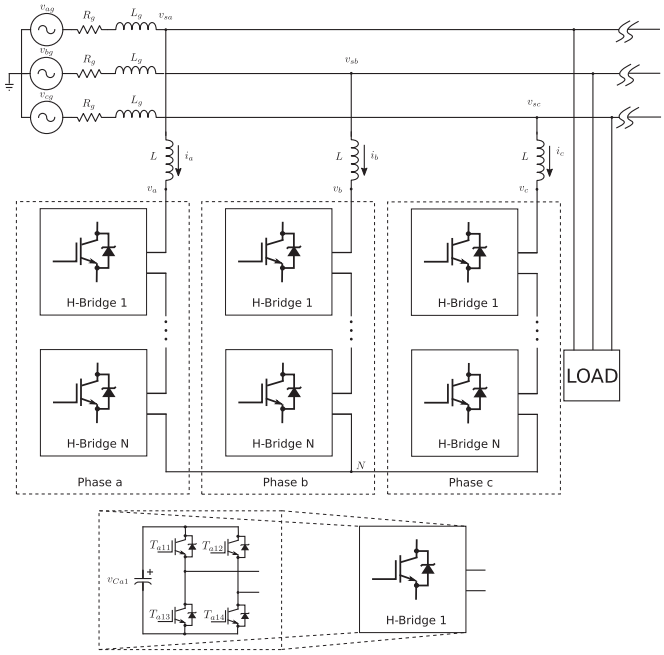

Fig. 3. Schematic diagram of a CHB-STATCOM.

where $i_{a,b,c}$ are the three-phase currents fed by the inverter, $v_{s(a,b,c)}$ are the three-phase grid voltages, and R is the equivalent resistance of the STATCOM. Using Clarke's transformation and discretizing, we have

$$i_{\alpha,\beta}(k+1) = i_{\alpha,\beta}(k) + \frac{T}{L}\left(v_{s(\alpha,\beta)}(k) - Ri_{\alpha,\beta}(k) - v_{\alpha,\beta}(k)\right) \tag{7}$$

$$v_{C(a,b,c)i}(k+1) = v_{C(a,b,c)i}(k) + \frac{T}{C}\left(s_{(a,b,c)i}(k) i_{a,b,c}(k)\right). \tag{8}$$

IV. MPC for CHB-STATCOM

As discussed in [33], the FCS-MPC problem can be divided into three distinct subproblems: 1) current tracking, which is presented in Section II; 2) cluster voltage balancing; and 3) individual voltage balancing. Once the switching vector $S_{\alpha\beta}$ is computed, it is transformed into abc coordinates. The zero-sequence voltage is computed to balance the three clusters. Given the predicted average voltages of the three clusters $\overline{v}_{a,b,c}(k+1)$, the cluster voltage balancing problem is

$$\begin{aligned} \min_{S_{a,b,c}(k)} \quad & \left\|\overline{v}^{ref}_{a,b,c}(k+1) - \overline{v}_{a,b,c}(k+1)\right\|_{Q_{cb}} \\ & + \|S_a(k) + S_b(k) + S_c(k)\|_{R_{cb}} \\ \text{s.t.} \quad & \overline{v}_{a,b,c}(k+1) = \overline{v}_{a,b,c}(k) + \frac{T}{C_{eq}}\left(S_{a,b,c}(k)\, i_{a,b,c}(k)\right) \\ & S_{a,b,c}(k) \in V_{a,b,c} \end{aligned} \tag{9}$$

where C_{eq} is the equivalent phase capacitance, equal in all the phases. The first term is a weighted norm of the deviation of the three cluster voltages from the reference value, and the second term aims to reduce the common-mode voltages. Then, the control algorithm searches among the redundant vectors in the abc coordinates, which are $4N+1$.

The third subproblem aims to find which H-bridges must supply the voltage in the proper polarity to obtain the overall desired voltage. Specifically, the individual voltage balancing problem solves the following optimization problem for each phase:

$$\begin{aligned} \min_{s_{(a,b,c)i}(k)} \quad & \left\|v^{\text{ref}}_{C(a,b,c)i}(k+1) - v_{C(a,b,c)i}(k+1)\right\|_{Q_v} \\ & + \left\|s_{(a,b,c)i}(k) - s_{(a,b,c)i}(k-1)\right\|_R \\ \text{s.t.} \quad & (8) \\ & \sum_{i=1}^{N} s_{(a,b,c)i}(k) = S^*_{(a,b,c)}(k) \end{aligned}$$

$$s_{(a,b,c)l} \times s_{(a,b,c)m} \geq 0 \quad \forall\, l, \quad m = 1,\ldots,N : l \neq m \tag{10}$$

where $S^*_{(a,b,c)}(k)$ is the switching vector computed by (9). Zhang et al. [16] also proposed an efficient algorithm for solving the capacitor voltage balancing problem. The solution can only be $s_{(a,b,c)i}(k) \in \{0,1\}$ if $S^*_{(a,b,c)}(k) \geq 0$; otherwise, $s_{(a,b,c)i}(k) \in \{-1,0\}$: the optimization problem becomes a $\{0,\ 1\}$ programming problem. Then, rather than computing the cost for every possible $\{0,\ 1\}$ combination of the switching states, which are 2^N+1, the cost increment in choosing 1 rather than 0 is computed for every phase cell. Once the N costs are determined, they are sorted in ascending order, and the first $|S^*_{(a,b,c)}(k)|$ elements of the array identify the best possible $s_{(a,b,c)i}(k)$ that must be selected to ensure the desired phase voltage. With this procedure, the computational cost is no longer exponential and is limited to the cost of the sorting algorithm, which must sort an array of N elements.

V. Simulation Results

A. Training Procedure

The power system in [34] was used as a test bench for the STATCOM. The original test bench was modified to deal with a medium-voltage grid. In particular, the voltage source was set to 10 kV, while the nominal power was set to 6 MVA. The CHB-STATCOM replaced the original two-level STATCOM. Three different case studies were considered, i.e., three STATCOM configurations with $N = 5$, 10, and 20 H-bridges, respectively. STATCOMs were sized to account for grid voltage variations of $\pm 20\%$ of the nominal voltage, providing up to ± 600 kvar. The inductor was 44 mH, while the capacitors for the three cases were 2600 V–250 μF, 1300 V–500 μF, and 650 V–1000 μF [3]. The test bench simulates changes in the voltage source propagating through grid elements until they reach the node where the STATCOM is connected. Several step changes in the source voltage were simulated in the interval of [0.8, 1.2] per unit (p.u.) voltage. In this way, the STATCOM, controlled by FCS-MPC,

reacted by providing reactive power in all the ranges of the rated STATCOM power (600 kvar). For each configuration of N, η, (w_i, w_s), and h, the data were collected from a simulation of 8 s with 80 step changes. Since the controller sampling time was 40 μs, the overall dataset was composed of 200 000 samples. Starting from this dataset, a further dataset was created by perturbing the inputs of the optimization problem, i.e., $i^{\text{ref}}_{\alpha,\beta}(k), i_{\alpha,\beta}(k), v_{s(\alpha,\beta)}(k)$, and $S_{\alpha,\beta}$(k-1), with random noise and computing the optimum values $S_{\alpha\beta}(k)$ for these new points. The original dataset and the perturbed one were merged into one dataset: this procedure gives robustness to the learning procedure and improves the NN performance. Once the dataset was collected, several NNs were trained in MATLAB, changing the activation functions of hidden and output neurons, training algorithms, and performance indexes. The training dataset was 70% of the overall set, while the validation and the test datasets were 15%. For each configuration, the three sets (training, validation, and test) were collected randomly grabbing the points from the overall set. In this way, every learning procedure had a different division of the datasets and computed a slightly different NN; therefore, it was possible to select the best one. This procedure was done to reduce the effect of dataset division on the NN performance. In this article, ten NNs were trained for each parameter combination, and the best one was selected. The best NN had hyperbolic a tangent function ("tansig") in the hidden neurons and a linear activation function ("purelin") in the output neurons, while the Bayesian regularization ("trainbr") was used for the learning procedure, considering as performance index the sum squared errors ("sse"). The selected NN was the one with minimum sum squared error. A shallow NN was employed since it was the simplest architecture and was accurate enough for our scope. Each NN training, considering 1000 epochs, takes on average about 4 min to be computed on the authors' PC (on 11th Gen Intel Core i7, eight cores, 2.8 GHz, 8-GB RAM, Linux Ubuntu 20.04, MATLAB 2021b).

B. Performance Evaluation

The performance of the different approaches was compared considering the mean absolute error (MAE) between current and reference, the switching frequency, and the total harmonic distortion (THD) on steady state and evaluating the transient response. The NN-FCS approach was tested using varying number of levels, weighting parameters, number of hidden neurons, and prediction horizon.

1) Different Numbers of Levels N: In order to analyze the generalization of the NN approach for different levels, three CHB inverters were considered, with $N = 5$, 10, and 20 H-bridges. The weighting coefficients (w_i, w_s) were fixed to (1, 0.1), and $\eta = 8$ hidden neurons are considered with a prediction horizon $h = 1$. Fig. 4 shows the switching frequency of the controllers for different steady-state conditions of the quadrature current reference. The frequency is not fixed due to the absence of a modulator and varies according to reference changes.

Fig. 5 presents the MAE, while Fig. 6 shows the THD. Fig. 7 presents the transient state, showing the dynamic response to a Δi_q step. It turned out that the NN-MPC controller provides a slightly lower frequency and harmonic distortion, compatible reference errors in steady state, and a slightly slower dynamic response. Moreover, it follows the optimal controller trend, irrespective of the number of levels of the converter, making the NN a general solution and a promising alternative for real-time implementation. Table I summarizes the average performances.

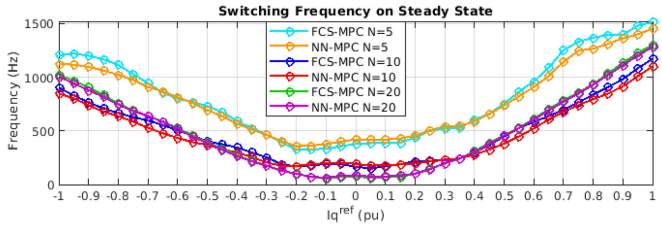

Fig. 4. Steady-state switching frequency at $N = 5$, 10, 20.

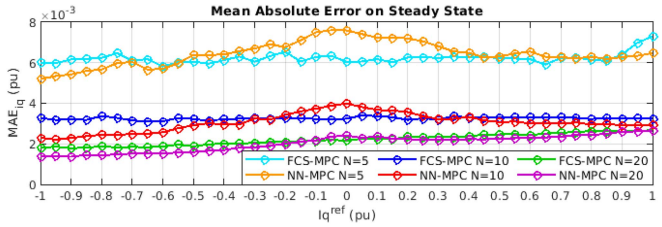

Fig. 5. Steady-state MAE at $N = 5$, 10, 20.

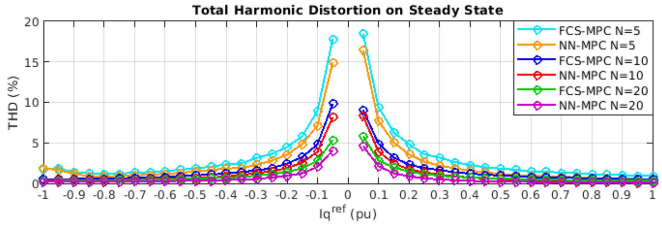

Fig. 6. Steady-state THD at $N = 5$, 10, 20.

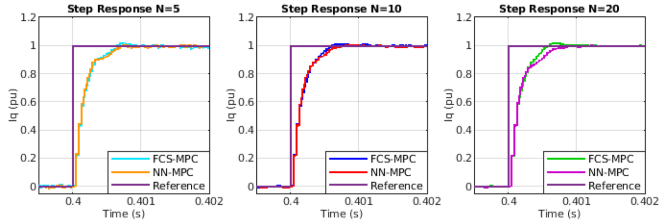

Fig. 7. Transient operations at $N = 5$, 10, 20.

TABLE I
AVERAGE PERFORMANCES FOR DIFFERENT LEVELS

	Frequency	MEA	THD	Transient MAE
FCS $N = 5$	809 Hz	0.0062 p.u.	3.3445	0.0715 p.u.
NN $N = 5$	783 Hz	0.0064 p.u.	2.6287	0.0704 p.u.
FCS $N = 10$	489 Hz	0.0033 p.u.	1.7212	0.0685 p.u.
NN $N = 10$	457 Hz	0.0031 p.u.	1.2918	0.0698 p.u.
FCS $N = 20$	498 Hz	0.0022 p.u.	1.0205	0.0682 p.u.
NN $N = 20$	491 Hz	0.0020 p.u.	0.6540	0.0725 p.u.

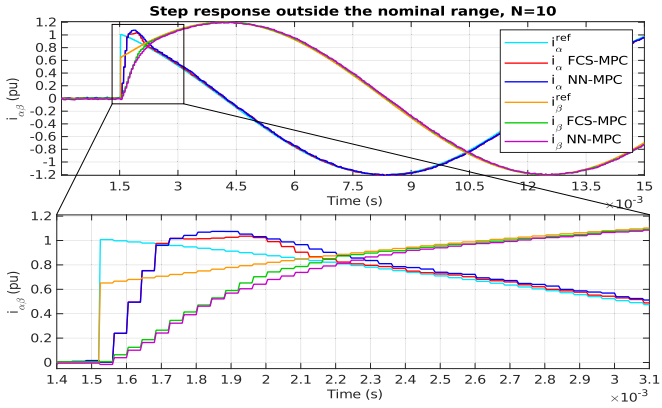

Fig. 8. Step response outside the nominal range; $N = 10$.

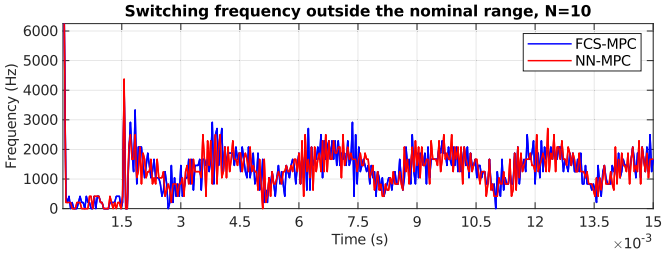

Fig. 9. Step response frequency outside the nominal range, $N = 10$.

Further tests were carried out to analyze the system outside the nominal conditions to evaluate the performance of the NN controller in a region outside the training dataset. In particular, a voltage 20% higher than the nominal value was applied, and a current 20% higher than the rated value was supplied. The grid voltage was 1.4 p.u., and the reference quadrature current step was 1.2 p.u. Fig. 8 shows the step response for the $N = 10$ inverter. A zoom of the $\alpha\beta$ currents is shown highlighting the satisfactory dynamic response. Fig. 9 shows the switching frequencies when applying the two controllers. The NN was able to generalize the control law even on operational conditions not considered in the training dataset. This result is still valid for $N = 5$ and $N = 20$ STATCOMs, but the relative plots are omitted for space reasons.

2) Different Weighting Coefficients (w_i, w_s)*:* The weighting factors of the FCS-MPC cost function in (4) allow a tradeoff between tracking performances and switching losses. By increasing the ratio w_i/w_s, the controller gives higher priority to reference error minimization at the expense of the switching frequency. Once the weighting factors are suitably tuned, it is possible to train the NN that will approximate the desired controller behavior. In order to evaluate the NN performance under different tuning parameters, tests were carried out by varying the FCS-MPC weighting coefficients (w_i, w_s) between (1, 0), (1, 0.1), (1, 0.5), (1, 1), and (0.5, 1). Thus, different NNs were trained to learn the controllers with different tuning parameters. The following tests refer to the $N = 10$ case, with $\eta = 8$ and $h = 1$. Figs. 10– 12 show the switching frequencies, the MAE, and the THD at steady state for the different control

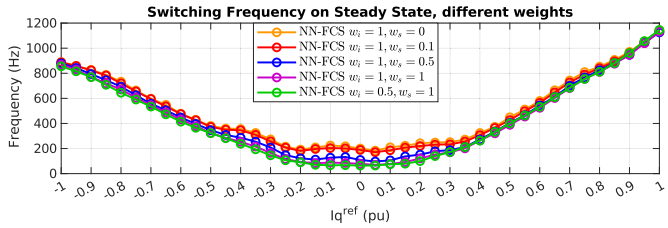

Fig. 10. Steady-state switching frequency at different weights.

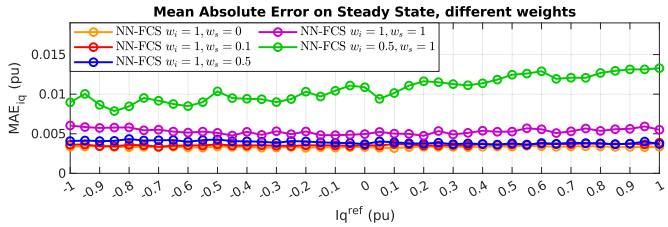

Fig. 11. Steady-state MAE at different weights.

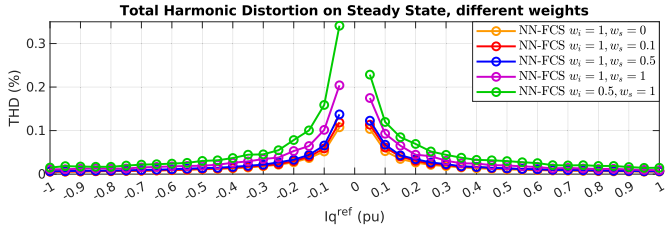

Fig. 12. Steady-state THD at different weights.

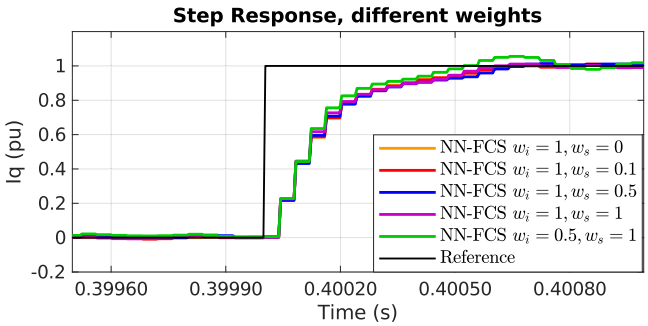

Fig. 13. Transient operations at different weights.

tunings, respectively. As the ratio increases, the MAE and the THD decrease at the expense of a higher switching frequency. Conversely, as the ratio decreases, the switching frequency becomes lower but MAE and THD increase. The step response in Fig. 13 shows that a low ratio also impacts the transient state, increasing both settling time and overshoot, as evident in the case $(w_i, w_s) = (0.5, 1)$.

3) Different Hidden Neurons η*:* The tradeoff between the performance and complexity of the NN is a key factor for the online implementation. In order to find the best tradeoff, tests were carried out by training NNs with a different number of hidden neurons. Starting from the data collected for the case in which

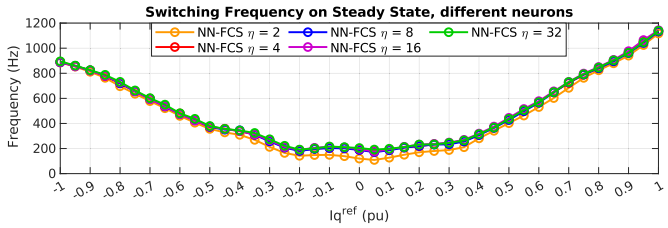

Fig. 14. Steady-state switching frequency at different hidden neurons.

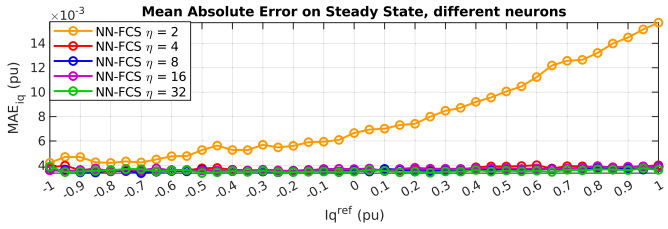

Fig. 15. Steady-state MAE at different hidden neurons.

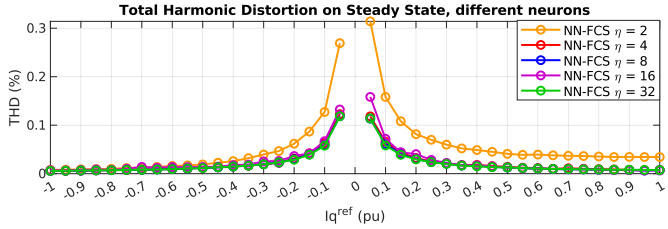

Fig. 16. Steady-state THD at different hidden neurons.

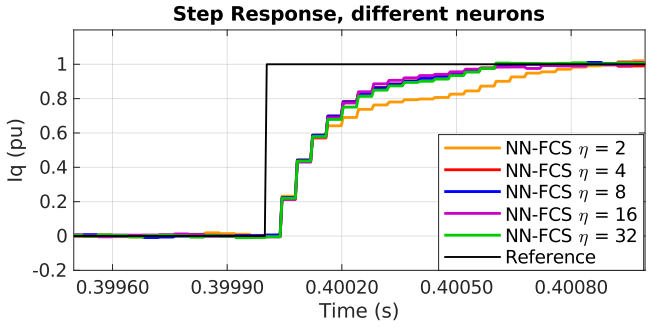

Fig. 17. Transient operations at different hidden neurons.

$N = 10$, $(w_i, w_s) = (1, 0.1)$, and $h = 1$, different NNs were trained with hidden neurons equal to 2, 4, 8, 16, and 32. Figs. 14–16 show the performance of the different NN architectures, while Fig. 17 presents the transient state. The tests above showed that even an NN with just two hidden neurons was enough to stabilize the system. However, its performance was poor if compared to NNs with more neurons. The performance analysis suggested that $\eta = 4$ was sufficient to guarantee satisfactory results. At the same time, $\eta = 8$ was the best option since it generated the lowest switching frequency, and its performance was compatible in terms of THD and MAE with respect to $\eta = 4, 16, 32$.

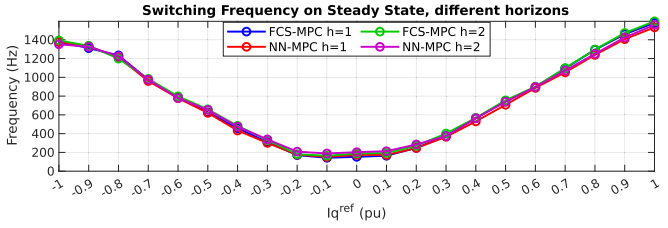

Fig. 18. Steady-state switching frequency at different prediction horizons.

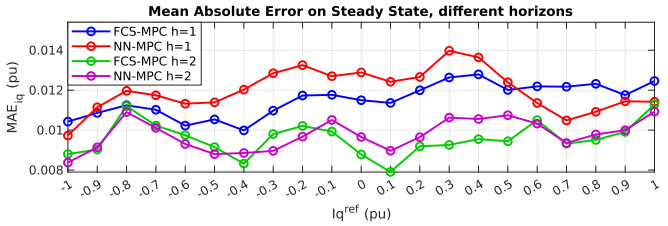

Fig. 19. Steady-state MAE at different prediction horizons.

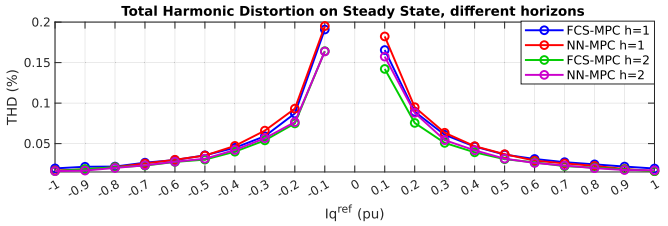

Fig. 20. Steady-state THD at different prediction horizons.

4) Different Prediction Horizons h: By increasing the prediction horizon of the FCS-MPC, it is possible to improve the overall performance of the controller [7]. In order to test the ability of the NN approach to learn multiple horizons, FCS-MPC tests were carried out by changing the prediction horizon. Since the increase of the control horizon led to a dramatic increment in computations, horizons larger than three were not tested. It turned out that horizon $h = 2$ led to better performances with respect to the case in which $h = 1$. Choosing $h = 3$, the one-step prediction controller was improved, but no significant improvements were found by changing h from 2 to 3. For this reason, NN approximations were tested just for $h = 1$ and 2. Figs. 18–20 show the steady-state switching frequency, MAE, and THD, respectively, for the case in which $N = 5$, $(w_i, w_s) = (1, 5)$, approximated with $\eta = 8$ NNs for horizon $h = 1, 2$. The NN-MPC tended to have slightly lower switching frequency and slightly higher THD when compared with the original FCS-MPC. Fig. 21 presents the transient response of the controllers, showing a slightly lower settling time of the NN approximations compared with the FCS-MPC. Table II summarizes the average performance for horizons $h = 1, 2$.

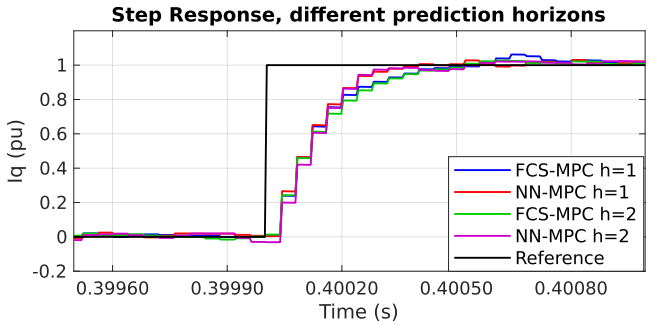

Fig. 21. Transient operations at different prediction horizons.

TABLE II
AVERAGE PERFORMANCE FOR DIFFERENT HORIZONS

	Frequency	MAE	THD	Transient MAE
FCS h=1	760.58 Hz	0.0115 p.u.	0.0529	0.1042 p.u.
NN h=1	745.66 Hz	0.012 p.u.	0.0542	0.1000 p.u.
FCS h=2	774.75 Hz	0.0096 p.u.	0.0456	0.1030 p.u.
NN h=2	766.63 Hz	0.0098 p.u.	0.0475	0.0921 p.u.

VI. HARDWARE-IN-THE-LOOP SIMULATIONS

Hardware-in-the-loop tests were carried out for $N = 5$, 10, and 20 cases. The weighting parameters were fixed to $(w_i, w_s) = (1, 0.1)$. The number of hidden neurons was $\eta = 8$: the best tradeoff between accuracy and cost complexity for the discussed implementation. The prediction horizon was $h = 1$ to allow a fair comparison with the FCS-MPC, whose classical solution is impractical for larger horizons with a large number of levels.

A. RTL Implementation

In this section, the RTL implementation is discussed, presenting the algorithmic state machines (ASMs) that realize the sequential steps of the algorithm and giving insights into the combinatorial part. In the ASMs, the notation $<=$ is a signal assignment, while $\leftarrow$ means that a register is updated, implying that the value is changed in the next clock cycle. The ASM that implements the NN calculation for $\eta = 8$ is shown in Fig. 22. The pre- and postprocessing calculations computed by the MATLAB NN are implemented in the corresponding states. The layer computations are implemented by using eight adders and eight multipliers opportunely multiplexed to create a pipeline structure to speed up the algorithm: the red lines in the figure underline the different stages of the pipeline. The activation function of the hidden layer is computed using a lookup table (LUT in the figure). The NN output is transformed into $S^{\pi/3}_{\alpha\beta}$ in order to compute the feasible switching vector. It is, then, transformed into the abc coordinate considering null zero-sequence obtaining S^0_{abc}. The cluster voltage balancing problem is solved by explicitly computing the cost for all the possible abc vectors that ensure the $\alpha\beta$ computed values. Starting from S^0_{abc}, the maximum and minimum possible values of the zero-sequence component are obtained, namely, $s^{\max}_\gamma$ and $s^{\min}_\gamma$. The costs are computed for all the admissible values, that are $S^i_{abc} = S^0_{abc} + i$, $s^{\min}_\gamma \le i \le s^{\max}_\gamma$, and the abc vector that

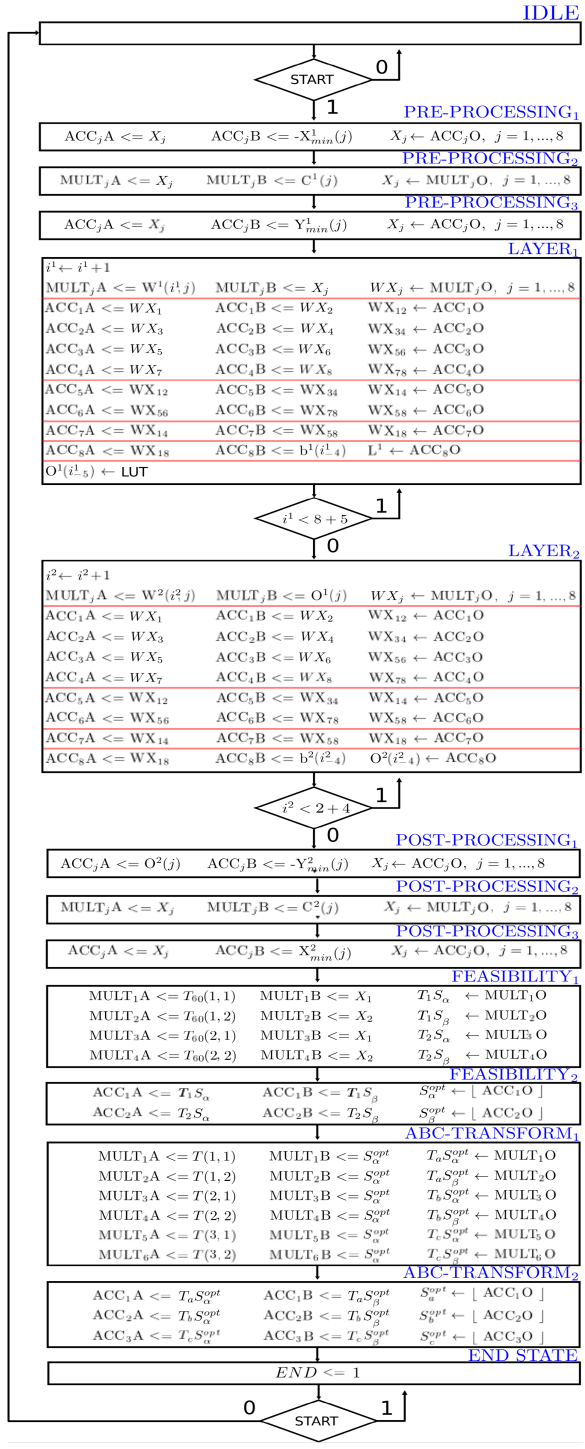

Fig. 22. NN algorithmic state machine.

minimizes the cost is selected. In the ASM in Fig. 23, the cost is initialized to be the maximum stored value, while the optimum value is initialized to be S^0_{abc}. The $const_{\text{abc}}$ value includes the term that is constant during the operations in the same sampling interval (INIT_1). The minimum and maximum values between S^0_a, S^0_b, and S^0_c are evaluated in order to compute the maximum and minimum zero-sequence admissible voltages (INIT_2 and INIT_3). The costs are individually calculated by employing 12

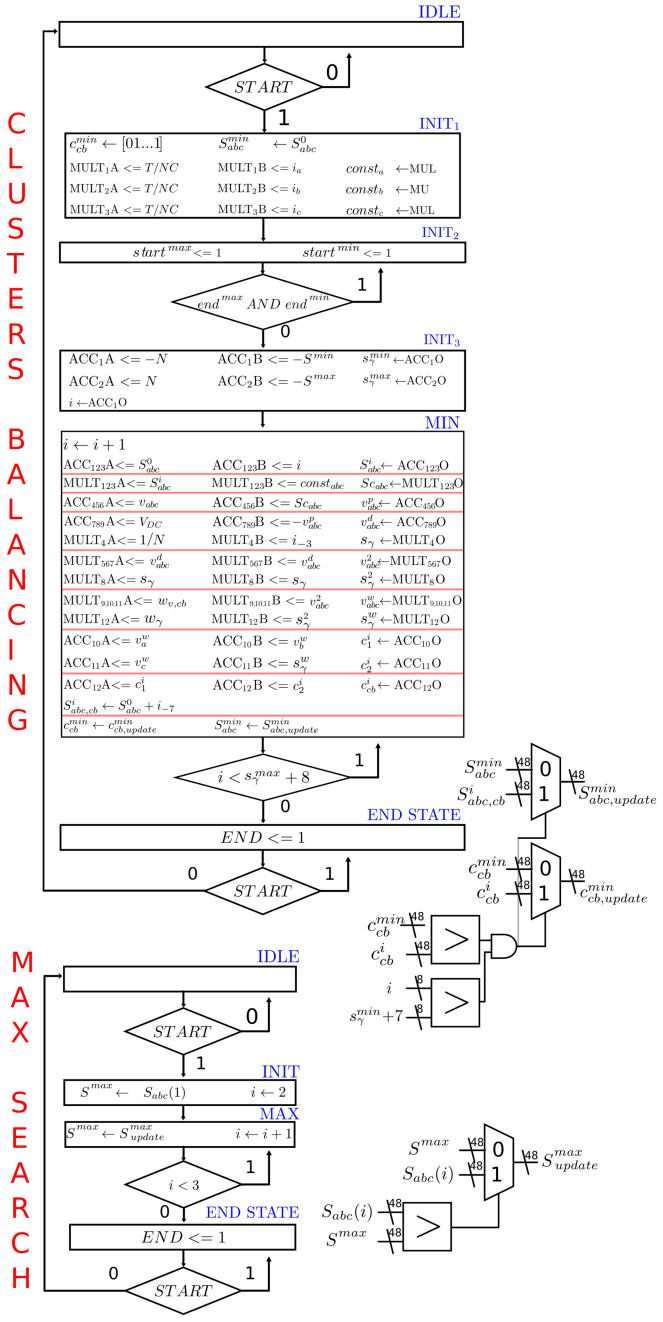

Fig. 23. Cluster voltage balancing ASM.

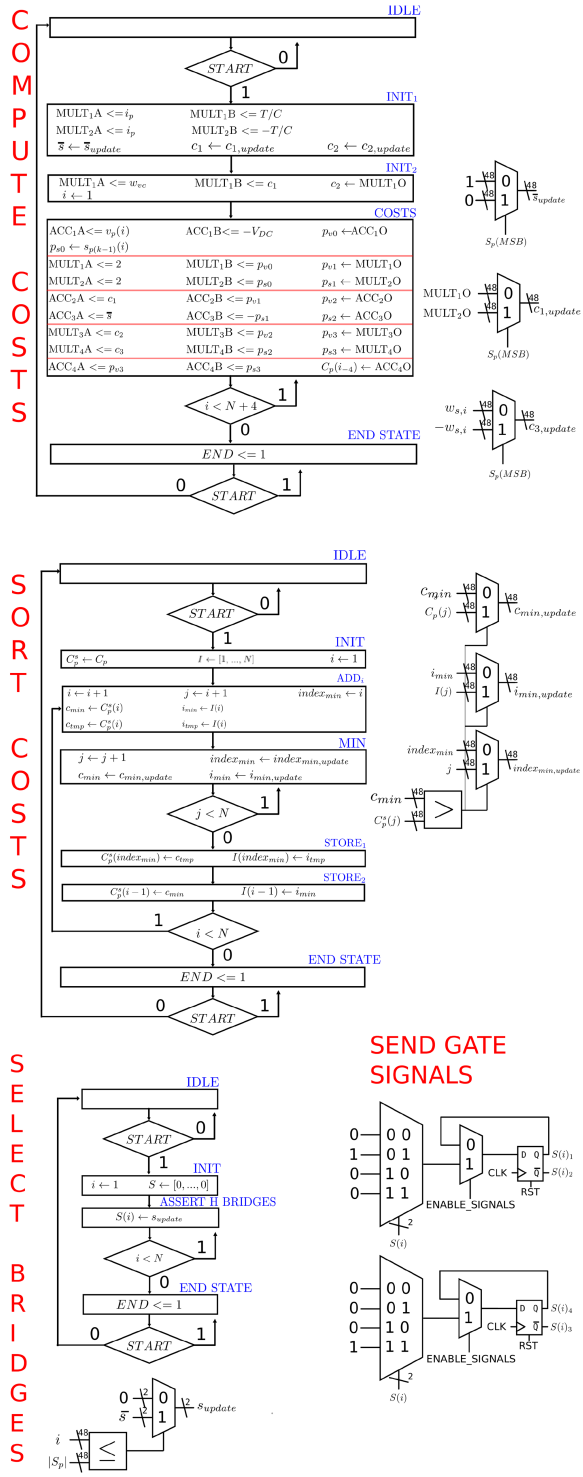

Fig. 24. Individual voltage balancing ASM.

adders and 12 multipliers multiplexed in an eight-level pipeline structure. The minimum cost and the optimal vector are iteratively updated by employing the comparator circuits in the figure.

The individual voltages balancing problem is solved for each abc phase starting from the optimal value $S^{\min}_{abc}$. It proceeds as follows: the cost for each H-bridge is computed; then, the cost array is sorted in increasing order. The first $|S^{\min}_{abc}|$ H-bridges are selected to be 1 or -1 according to the sign of $S^{\min}_{abc}$, while the others are set to 0. Fig. 24 presents the implementation of the different steps, computed concurrently for each phase.

The first ASM is related to the computation of the costs: in INIT_1 and INIT_2, those variables that are constant during the overall computation are calculated, depending on the sign of $S_p, p \in \{a, b, c\}$, computed by the cluster voltages problem. If the most significant bit (MSB) of S_p is 0, the cost difference in

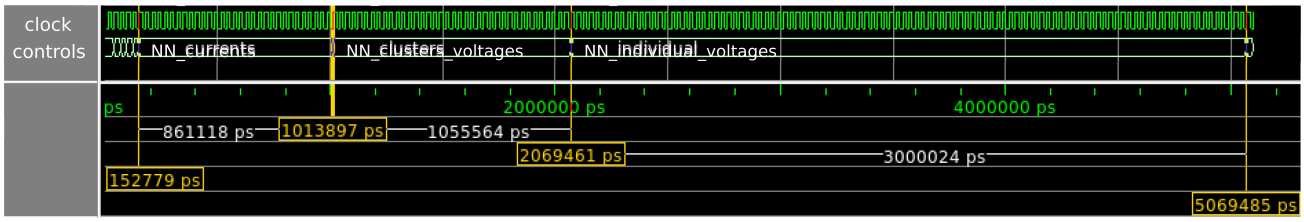

Fig. 25. NN-MPC simulation in ModelSim for $N = 10$.

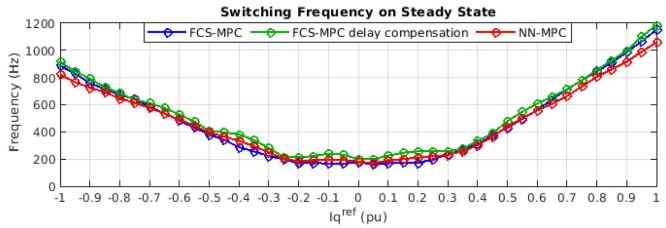

Fig. 26. Steady-state switching frequency HIL for $N = 10$.

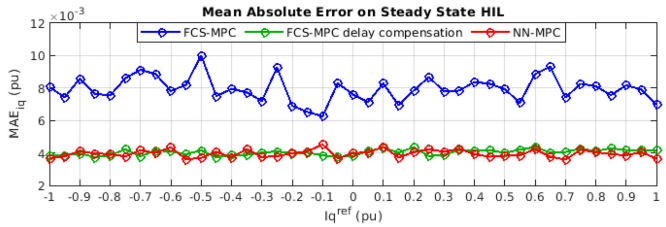

Fig. 27. Steady-state MAE HIL for $N = 10$.

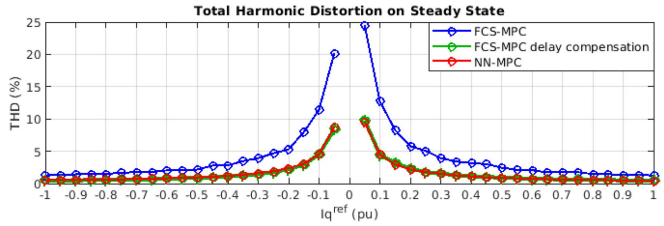

Fig. 28. Steady-state THD HIL for $N = 10$.

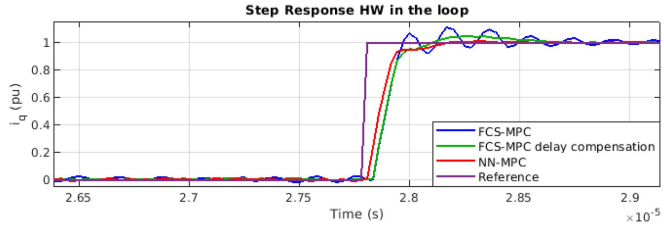

Fig. 29. Step response HIL for $N = 10$.

choosing 1 rather than 0 is computed. Otherwise, if the MSB is 1, the cost difference in choosing -1 rather than 0 is evaluated.

The N costs are evaluated and stored by employing a five-level pipeline and utilizing four adders and four multipliers: it makes it possible to use the 12 total adders and multipliers for computing the three phases in parallel. The second ASM implements the sorting algorithm: a selection sort algorithm is chosen for simplicity. The third ASM selects the best H-bridges setting them to 1 or -1, leaving the remaining values equal to 0. These switching variables are converted into gate signals.

On the other hand, FCS-MPC implementation employs the same architecture for the clusters and voltage balancing problems but differs for the current control loop. The ASM that implements the FCS current control consists of an initialization step, the cost computation, and the evaluation of all the possible vectors (similar to cluster voltage balancing ASM) and the abc transform. The cost computation is realized using the same adders and multipliers multiplexed in an eight-level pipeline. Considering the steps needed by the exhaustive search, the pipeline, the initialization, and the transform, the total number of steps needed by the control loop is $12N^2 + 6N + 14$.

B. Hardware-in-the-Loop Results

The two algorithms were implemented using a Terasic DE-10 Nano board equipped with an FPGA SoC Intel Cyclone V (5CSEBA6U23I7). Each control algorithm was tested via hardware-in-the-loop setup, in which the STATCOM model was simulated in Simulink environment, while the algorithm computations were implemented on the FPGA. The FPGA-in-the-loop app in Simulink was used to generate a Quartus project that embedded the hardware implementation. More in detail, it provided the Universal Asynchronous Receiver–Transmitter connection that permitted the communication between the Simulink model and the FPGA control. The simulation step was set according to the actual clock frequency needed to guarantee the correct execution of the algorithm. With the described implementation, with the register sized to 48 bits and the sharing of the arithmetic resources, the Quartus Time Analyzer ensured correct execution with a clock frequency of 36 MHz. The clock cycles needed for the two algorithms and the execution times for different values of N are reported in Table III.

It can be seen that the time delay of the NN is the same, independently of the number of levels. This result leads the NN-MPC to have a much shorter execution time than the classic FCS-MPC. The time spent in clusters and individual voltage balancing is the same for the two controllers since the neural network replaced just the current control. For $N = 5$, FCS-MPC takes 11.75 μs, while NN-MPC takes 3.03 μs. Thus, the classic control is implementable in real time, and it is possible to compute the input within the 40-μs sampling interval. By the way, NN-MPC takes an execution time nearly four times smaller. For $N = 10$,

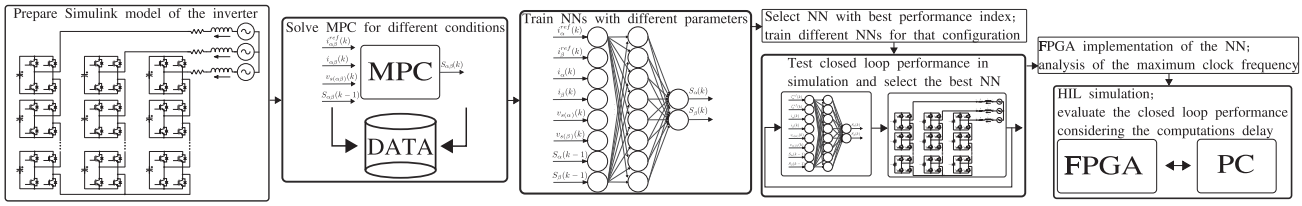

Fig. 30. Schematic diagram flow of the overall procedure.

TABLE III
COMPUTATIONAL COST AND TIME DELAYS OF FCS AND NN MPCS

	clock cycles	μs, $N = 5$	μs, $N = 10$	μs, $N = 20$
Current FCS	$12N^2$+6N+14	≈ 9.58	≈35.42	≈137.08
Current NN	31	≈0.86	≈0.86	≈0.86
Clusters	2N+19	≈0.80	≈1.08	≈1.64
Voltages	$\frac{1}{2}N^2+\frac{9}{2}N+14$	≈1.36	≈3.03	≈8.44
FCS-MPC	$\frac{25}{2}N^2+\frac{25}{2}N+39$	≈11.75	≈39.52	≈147.17
NN-MPC	$\frac{1}{2}N^2+\frac{9}{2}N+45$	≈3.03	≈4.97	≈10.94

FCS-MPC takes 39.52 μs, which is about the overall sampling interval. Using the FCS-MPC algorithm, it is just possible to calculate the control for the next sampling time.

The performance of the standard FCS-MPC is degraded due to the one sampling interval delay. Because of this, it is a common solution to use a delay compensation as in [9]. Using NN-MPC, the control loop is completed in 4.97 μs, which is one order of magnitude less than the conventional FCS-MPC, and it makes it possible to apply the input in the same sampling interval. Fig. 25 shows the algorithm steps simulated via ModelSim, whereas Figs. 26–29 shows the HIL simulations for $N = 10$. Compared with the FCS-MPC with delay compensation, NN-MPC has similar performance on the steady-state error and harmonic distortion, as seen in Figs. 27 and 28. However, in Fig. 26, the steady-state switching frequency of the NN-MPC is still slightly lower, and in Fig. 29, the dynamic performances of the proposed approach are superior since it has faster response and does not suffer from the inaccuracy of the two-step forward prediction that leads, in any case, to a deterioration of the performances.

For $N = 20$, the standard control is impractical unless the sampling interval is increased. The proposed one takes 10.94 μs, which is small enough to permit the real-time implementation. It is interesting to notice that the time spent for the NN with $\eta = 8$, 4, 2 hidden neurons was similar for the described implementation since 12 adders and multipliers were used in parallel (the cluster voltages balancing requires four of them for each phase). In particular $\eta = 4$ requires 27 iterations, while $\eta = 2$ requires 25 iterations, resulting in 0.75 and 0.69 μs. With the described implementation, without adding extra hardware resources, the NN with $\eta = 16$ required to solve the two loops in the ASM in Fig. 22 twice, resulting in 50 iterations. Analogously, the $\eta = 32$ NN required to solve the loops four times, and 88 iterations were needed. The execution time spent was about 1.39 and 2.44 μs for the two cases. The FCS-MPC for $N = 10$ and horizons $h = 2$ and 3 required about 89.9 and 852 us. In contrast, the time spent by the NN-MPC is always the same as Table III, regardless of the prediction horizon. Fig. 30 presents a schematic diagram flow that summarizes the whole procedure, which embeds: 1) the simulation of the FCS-MPC to control the CHB inverter for collecting data; 2) the training of the NN; 3) the selection of the best NN in the closed-loop performance; 4) the implementation on the FPGA; and, finally, 5) the HIL simulation used to analyze the effect of computation delay.

VII. CONCLUSION

In this article, an NN-MPC for a cascaded H-bridge converter was presented. A shallow NN was trained to learn the FCS-MPC solution. The control was tested on a CHB-STATCOM in simulation to analyze the performance of the proposed controller in comparison to the standard FCS-MPC. The controllers were tested for different numbers of levels, and the results suggested that the NN well follows the optimal behavior in each case. The standard FCS-MPC and the proposed algorithm were implemented on the FPGA, and the time spent by the computations of the controllers was derived. It turned out that the standard FCS-MPC is only implementable for a CHB with ten H-bridges with a delay compensation technique and becomes impractical for 20 H-bridges.

On the other hand, the time delay of the proposed approach allows the control input to be computed within the same sampling intervals, even for a CHB inverter with 20 H-bridges per phase.

REFERENCES

[1] M. Malinowski, K. Gopakumar, J. Rodriguez, and M. A. Pérez, "A survey on cascaded multilevel inverters," *IEEE Trans. Ind. Electron.*, vol. 57, no. 7, pp. 2197–2206, Jul. 2010.
[2] H. Akagi, "Multilevel converters: Fundamental circuits and systems," *Proc. IEEE*, vol. 105, no. 11, pp. 2048–2065, Nov. 2017.
[3] A. M. Saif, C. Buccella, V. Patel, M. Tinari, and C. Cecati, "Design and cost analysis for STATCOM in low and medium voltage systems," in *Proc. IEEE 44th Annu. Conf. Ind. Electron. Soc.*, 2018, pp. 3938–3943.
[4] F. Borrelli, A. Bemporad, and M. Morari, *Predictive Control for Linear and Hybrid Systems*. Cambridge, U.K.: Cambridge Univ. Press, 2017.
[5] E. F. Camacho and C. B. Alba, *Model Predictive Control*. Berlin, Germany: Springer, 2013.
[6] S. Vazquez, J. Rodriguez, M. Rivera, L. G. Franquelo, and M. Norambuena, "Model predictive control for power converters and drives: Advances and trends," *IEEE Trans. Ind. Electron.*, vol. 64, no. 2, pp. 935–947, Feb. 2017.
[7] T. Geyer and D. E. Quevedo, "Performance of multistep finite control set model predictive control for power electronics," *IEEE Trans. Power Electron.*, vol. 30, no. 3, pp. 1633–1644, Mar. 2015.
[8] B. Stellato, T. Geyer, and P. J. Goulart, "High-speed finite control set model predictive control for power electronics," *IEEE Trans. Power Electron.*, vol. 32, no. 5, pp. 4007–4020, May 2017.

[9] P. Cortes, J. Rodriguez, C. Silva, and A. Flores, "Delay compensation in model predictive current control of a three-phase inverter," *IEEE Trans. Ind. Electron.*, vol. 59, no. 2, pp. 1323–1325, Feb. 2012.
[10] P. Cortes, A. Wilson, S. Kouro, J. Rodriguez, and H. Abu-Rub, "Model predictive control of multilevel cascaded H-bridge inverters," *IEEE Trans. Ind. Electron.*, vol. 57, no. 8, pp. 2691–2699, Aug. 2010.
[11] C. D. Townsend, T. J. Summers, and R. E. Betz, "Multigoal heuristic model predictive control technique applied to a cascaded H-bridge STATCOM," *IEEE Trans. Power Electron.*, vol. 27, no. 3, pp. 1191–1200, Mar. 2012.
[12] B. Gutierrez and S.-S. Kwak, "Modular multilevel converters (MMCs) controlled by model predictive control with reduced calculation burden," *IEEE Trans. Power Electron.*, vol. 33, no. 11, pp. 9176–9187, Nov. 2018.
[13] M. R. Nasiri, S. Farhangi, and J. Rodriguez, "Model predictive control of a multilevel CHB STATCOM in wind farm application using diophantine equations," *IEEE Trans. Ind. Electron.*, vol. 66, no. 2, pp. 1213–1223, Feb. 2019.
[14] Z. Ni and M. Narimani, "A new fast formulation of model predictive control for CHB STATCOM," in *Proc. IEEE 45th Annu. Conf. Ind. Electron. Soc.*, 2019, vol. 1, pp. 3493–3498.
[15] R. Baidya, R. P. Aguilera, P. Acuña, S. Vazquez, and H. D. T. Mouton, "Multistep model predictive control for cascaded H-bridge inverters: Formulation and analysis," *IEEE Trans. Power Electron.*, vol. 33, no. 1, pp. 876–886, Jan. 2018.
[16] Y. Zhang, X. Wu, X. Yuan, Y. Wang, and P. Dai, "Fast model predictive control for multilevel cascaded H-bridge STATCOM with polynomial computation time," *IEEE Trans. Ind. Electron.*, vol. 63, no. 8, pp. 5231–5243, Aug. 2016.
[17] A. Dekka, B. Wu, V. Yaramasu, and N. R. Zargari, "Dual-stage model predictive control with improved harmonic performance for modular multilevel converter," *IEEE Trans. Ind. Electron.*, vol. 63, no. 10, pp. 6010–6019, Oct. 2016.
[18] J. Huang et al., "Priority sorting approach for modular multilevel converter based on simplified model predictive control," *IEEE Trans. Ind. Electron.*, vol. 65, no. 6, pp. 4819–4830, Jun. 2018.
[19] Y. Zhang, X. Wu, and X. Yuan, "A simplified branch and bound approach for model predictive control of multilevel cascaded H-bridge STATCOM," *IEEE Trans. Ind. Electron.*, vol. 64, no. 10, pp. 7634–7644, Oct. 2017.
[20] Y. Zhang, X. Yuan, X. Wu, Y. Yuan, and J. Zhou, "Parallel implementation of model predictive control for multilevel cascaded H-bridge STATCOM with linear complexity," *IEEE Trans. Ind. Electron.*, vol. 67, no. 2, pp. 832–841, Feb. 2020.
[21] L. Diao, J. Tang, P. C. Loh, S. Yin, L. Wang, and Z. Liu, "An efficient DSP–FPGA-based implementation of hybrid PWM for electric rail traction induction motor control," *IEEE Trans. Power Electron.*, vol. 33, no. 4, pp. 3276–3288, Apr. 2018.
[22] T. Atalik et al., "Multi-DSP and -FPGA-based fully digital control system for cascaded multilevel converters used in FACTS applications," *IEEE Trans. Ind. Informat.*, vol. 8, no. 3, pp. 511–527, Aug. 2012.
[23] Z. Mynar, L. Vesely, and P. Vaclavek, "PMSM model predictive control with field-weakening implementation," *IEEE Trans. Ind. Electron.*, vol. 63, no. 8, pp. 5156–5166, Aug. 2016.
[24] O. Gulbudak and E. Santi, "FPGA-based model predictive controller for direct matrix converter," *IEEE Trans. Ind. Electron.*, vol. 63, no. 7, pp. 4560–4570, Jul. 2016.
[25] P. Martin Sanchez, O. Machado, E. J. Peña, F. J. Rodriguez, and F. J. Meca, "FPGA-based implementation of a predictive current controller for power converters," *IEEE Trans. Ind. Informat.*, vol. 9, no. 3, pp. 1312–1321, Aug. 2013.
[26] Z. Zhang, F. Wang, T. Sun, J. Rodríguez, and R. Kennel, "FPGA-based experimental investigation of a quasi-centralized model predictive control for back-to-back converters," *IEEE Trans. Power Electron.*, vol. 31, no. 1, pp. 662–674, Jan. 2016.
[27] I. S. Mohamed, S. Rovetta, T. D. Do, T. Dragicević, and A. A. Z. Diab, "A neural-network-based model predictive control of three-phase inverter with an output *LC* filter," *IEEE Access*, vol. 7, pp. 124737–124749, 2019.
[28] M. Novak and T. Dragicevic, "Supervised imitation learning of finite-set model predictive control systems for power electronics," *IEEE Trans. Ind. Electron.*, vol. 68, no. 2, pp. 1717–1723, Feb. 2021.
[29] M. Novak and F. Blaabjerg, "Supervised imitation learning of FS-MPC algorithm for multilevel converters," in *Proc. IEEE 23rd Eur. Conf. Power Electron. Appl.*, 2021, pp. P-1–P-10.
[30] D. Wang et al., "Model predictive control using artificial neural network for power converters," *IEEE Trans. Ind. Electron.*, vol. 69, no. 4, pp. 3689–3699, Apr. 2022.
[31] S. Wang, T. Dragicevic, Y. Gao, and R. Teodorescu, "Neural network based model predictive controllers for modular multilevel converters," *IEEE Trans. Energy Convers.*, vol. 36, no. 2, pp. 1562–1571, Jun. 2021.
[32] F. Simonetti, G. D. Di Girolamo, A. D'Innocenzo, and C. Cecati, "Machine learning for model predictive control of cascaded H-bridge inverters," in *Proc. IEEE 21st Mediterranean Electrotech. Conf.*, 2022, pp. 1241–1246.
[33] F. Simonetti, G. D. Girolamo, A. D'Innocenzo, and C. Cecati, "A neural network approach for efficient finite control set MPC of cascaded H-bridge STATCOM," in *Proc. IEEE 47th Annu. Conf. Ind. Electron. Soc.*, 2021, pp. 1–6.
[34] P. Giroux, G. Sybille, and H. Le-Huy, "Modeling and simulation of a distribution STATCOM using Simulink's power system blockset," in *Proc. IEEE 27th Annu. Conf. Ind. Electron. Soc.*, 2001, vol. 2, pp. 990–994.

Francesco Simonetti (Student Member, IEEE) received the master's degree in control systems and computer engineering in 2020 from the University of L'Aquila, L'Aquila, Italy, where he is currently working on optimal control and machine learning techniques for multilevel converters toward the Ph.D. degree with the Department of Engineering and Information Science and Mathematics.

His current research interests include optimal control and machine learning techniques for multilevel converters.

Alessandro D'Innocenzo (Member, IEEE) received the Ph.D. degree in electrical and information engineering from the University of L'Aquila, L'Aquila, Italy, in 2007, and the International Curriculum Option of Doctoral Studies in hybrid control for complex, distributed and heterogeneous embedded systems, in 2007.

He is currently an Associate Professor with the Department of Engineering and Information Sciences and Mathematics, University of L'Aquila, where he was a Postdoctoral Researcher with the Department of Electrical and Information Engineering in 2007 and 2009. He was a Postdoctoral Fellow with the Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA, USA, in 2008. His research interests include system identification, optimal control, and fault detection through interdisciplinary approaches across control theory and artificial intelligence, with applications to networked control systems, building automation, and power systems.

Dr. D'Innocenzo received the Filauro Foundation Award for Ph.D. students in 2005 and the "Best Application Paper Award" of the European Control Conference in 2015. He was invited plenary speaker in the 9th International Workshop on Reachability Problems in 2015. He obtained the National Scientific Qualification for the role of Full Professor in Control Theory in 2019.

Carlo Cecati (Fellow, IEEE) received the Dr.Ing. degree in electrotechnical engineering from the University of L'Aquila, L'Aquila, Italy, in 1983.

Since 1983, he has been with the University of L'Aquila, where he has been a Professor of Industrial Electronics and Drives since 2006. He was a Qianren Talents Professor with the Harbin Institute of Technology, Harbin, China, from 2015 to 2017. He is the Chief Technical Officer at DigiPower Ltd., an R&D company active in the field of power electronics. His research interests include power electronics, distributed generation, e-transportation, smart grids, and related technologies.

Dr. Cecati was a co-Editor-in-Chief (from 2010 to 2012) and Editor-in-Chief (from 2013 to 2015) for the IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS. He was a corecipient of the 2012 and 2013 Best Paper Awards from the IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, the 2012 Best Paper Award from the *IEEE Industrial Electronics Magazine*, and the 2019 Outstanding Paper Award from the IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS. He received the Antony J. Hornfeck Award from the IEEE Industrial Electronics Society in 2017, the title of "Commander of the Republic of Italy" from the President of the Republic of Italy, and the Eugene Mittlemann Achievement Award from the IEEE in 2021.