(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
22 December 2016 (22.12.2016)

WIPO | PCT

(10) International Publication Number
WO 2016/202623 A1

(51) International Patent Classification:
*G05B 13/04* (2006.01) *H02M 7/487* (2007.01)
*H02M 7/483* (2007.01)

(21) International Application Number:
PCT/EP2016/062807

(22) International Filing Date:
6 June 2016 (06.06.2016)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
15172241.0 16 June 2015 (16.06.2015) EP

(71) Applicant: **ABB SCHWEIZ AG** [CH/CH]; Brown Boveri Strasse 6, CH-5400 Baden (CH).

(72) Inventors: **GEYER, Tobias**; Seestrasse 113, 8002 Zürich (CH). **ROHR, Eduardo**; Ziegelhaustrasse 12, 5400 Baden (CH). **VALLONE, Joel**; Ch. de Cocagne 23, 1030 Bussigny (CH).

(74) Agent: **ABB PATENT ATTORNEYS**; ABB Schweiz AG, Intellectual Property (CH-IP), Brown Boveri Strasse 6, 5400 Baden (CH).

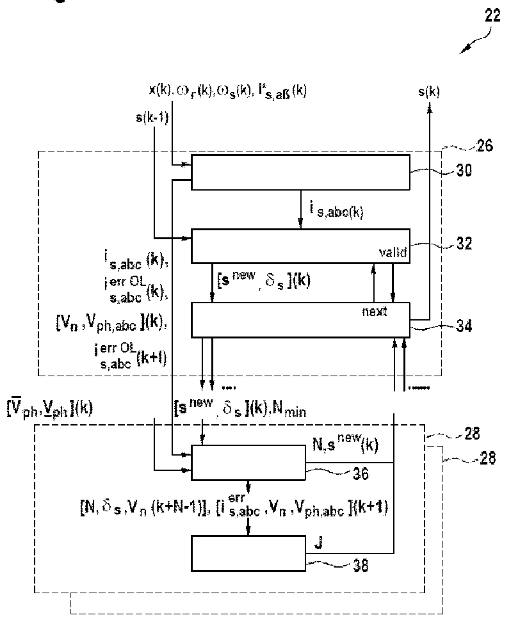
(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

**Published**:
— *with international search report (Art. 21(3))*

(54) Title: FPGA-BASED MODEL PREDICTIVE CONTROL

Fig. 2


22
$x(k), \omega_r(k), \omega_s(k), i^*_{s,\alpha\beta}(k)$
$s(k)$
$s(k-1)$
26
30
$i_{s,abc}(k)$
32
valid
$i_{s,abc}(k)$, $i^{err\,OL}_{s,abc}(k)$, $[V_n, V_{ph,abc}](k)$, $i^{err\,OL}_{s,abc}(k+l)$
$[s^{new}, \delta_s](k)$
next
34
$[\bar{V}_{ph}, \underline{V}_{ph}](k)$
$[s^{new}, \delta_s](k), N_{min}$
28
28
$N, s^{new}(k)$
36
$[N, \delta_s, V_n(k+N-1)], [i^{err}_{s,abc}, V_n, V_{ph,abc}](k+1)$
J
38

(57) **Abstract**: An FPGA (22) for controlling an electrical converter (12) comprises an enumeration block (32) adapted for generating possible next switch positions (*$s^{new}(k)$*) for semiconductor switches of the electrical converter (12) based on an actual applied switch position (*s(k-1)*); a plurality of explorer blocks (28), each explorer block (28) adapted for calculating a cost value (*J*) for a possible next switch position of the semiconductor switches by: receiving a possible next switch position (*$s^{new}(k)$*); calculating system variables at future time instants from system variables at a current time instant of the electrical converter (12) and the load (24) based on the possible next switch position, wherein the system variables at future time instants are calculated from the system variables at the current time instant with differential equations modelling the electrical converter (12) and the load (24); determining a cost value (*J*) from the system variables at future time instants by evaluating a cost function with the system variables at future time instants; an arbiter block (34) for selecting the next switch position (*s(k)*) to be applied to the electrical converter (12) from the possible next switch positions by: receiving possible next switch positions (*$s^{new}(k)$*) from the enumeration block (32); selecting a non-operating explorer block (28) and sending a received possible next switch position to the non-operating explorer block; receiving a cost value (*J*) for the respective possible next switch position from a finished explorer block; when all possible next switch positions received from the enumeration block (32) have been processed, selecting the next switch position (*s(k)*) as the possible next switch position (*$s^{new}(k)$*) with the lowest cost value (*J*). Each explorer block (28) is further adapted for determining a prediction horizon (N) for the possible next switch position (snew(k)) at which at least one of the calculated system variables at future time instants has a deviation from a reference for the system variable, which is bigger than a predefined deviation for the system variable. A prediction horizon (N) for a system variable at future time instants is determined via a linear extrapolation, in which the system variable at the future time instants is calculated from the system variable at the current time instant, and the prediction horizon (N) is determined based on an intersection point of the linearly extrapolated system variable between the current time instant and the future time instant with a maximal possible deviation from a reference of the system variable. Moreover, the intersection point is determined iteratively by a binary search.

WO 2016/202623 A1

# FPGA-based model predictive control

## FIELD OF THE INVENTION

The invention relates to the field of model predictive control or electrical converters. In particular, the invention relates to an FPGA for controlling an electrical converter and to an electrical converter comprising a controller with such an FPGA.

## BACKGROUND OF THE INVENTION

Electrical converters for converting an input voltage into an output voltage of different frequency and/or different voltage magnitude comprise controllable semiconductors which are switched by a control method to achieve the desired output voltage.

One such control method is model predictive control, in which the switch position of the switches is determined by solving an optimization problem that is based on a physical model of the converter and optionally a connected load. The model is usually stated in discrete-time differential equations. However, such type of control method is usually computational demanding and the control hardware has to be fast to solve the specific model predictive control problem within the short time available before the next switch position is needed.

For example, instead of performing the control method in software, WO 2009/080 407 A1 proposes a method for operating an electrical machine, in which a model predictive control method is performed by an FPGA (field programmable gate array) interconnected with a DSP (digital signal processor) to reduce the computation time of the method.

WO 2013/110 532 A1 relates to a model predictive control method for an electrical converter, which is parallelized and executed in a multi-core processor to avoid an FPGA-based solution.

WO 2014/006200 A1 discloses a FPGA for controlling an electrical converter.

In "Model Predictive Direct Torque Control of a variable speed drive with a five-level inverter", Tobias Geyer et al, IECON 2009 - 35TH ANNUAL CONFERENCE OF IEEE INDUSTRIAL ELECTRONICS (IECON 2009) - 3-5 NOV. 2009 - PORTO, PORTUGAL, IEEE, PISCATAWAY, NJ, USA, 3 November 2009 (2009-11-03), pages 1203-1208, X P031 629326, ISBN: 978-1-4244-4648-3 the principles of the Model Predictive Direct Torque Control (MPDTC), which reduces the converter's switching losses and improves the torque's Total Harmonic Distortion (THD) with respect to standard Direct Torque Control (DTC) are disclosed, while maintaining DTC's favorable dynamic and robustness properties. The MPDTC is adapted and applied to a five-level convert-er driving a high frequency induction machine.

## DESCRIPTION OF THE INVENTION

It is an objective of the invention to provide a controller that is adapted to perform model predictive control for an electric converter in a fast and efficient way.

This objective is achieved by the subject-matter of the independent claims. Further exemplary embodiments are evident from the dependent claims and the following description.

An aspect of the invention relates to an FPGA for controlling an electrical converter. The FPGA may be a component of a controller of the electrical converter, which receives measured or estimated system variables such as input currents and/or output currents and/or voltages and/or fluxes and/or speeds and/or frequencies, and which determines switch positions for semiconductor switches of the electrical converter with the aid of a model predictive control method.

According to an embodiment of the invention, the FPGA comprises an enumeration block, a plurality of explorer blocks and an arbiter block. Usually, such blocks and their configuration may be described with a hardware description language, which then may be applied to the logic blocks of the FPGA to form these blocks in hardware.

The enumeration block is adapted for generating possible next switch positions for semiconductor switches of the electrical converter based on an actual applied switch position. A switch position of the electrical converter may comprise all of the individual switch positions (such as on/off) of the semiconductor switches. The actual applied switch position may be the actual switching state of the semiconductor switches and, for example, may be provided by the FPGA during the previous control cycle at the previous time step.

The explorer blocks are used for calculating a cost value for each of the possible next switch positions based on model predictive control. This calculation may be performed in parallel in different explorer blocks for different possible next switch positions.

Each explorer block is adapted for calculating a cost value for a possible next switch position of the semiconductor switches by: receiving a possible next switch position from the arbiter block; calculating system variables at future time instants from system variables at the current time instant of the electrical converter based on the possible next switch position, wherein the system variables at future time instants are calculated from the system variables at the current time instant with differential equations modelling the electrical converter and the load; and determining a cost value from the system variables at future time instants by evaluating a cost function with the system variables at future time instants.

The system variables may be interpreted as the state of the electrical converter and the load, and, for example, may comprise input currents and/or voltages, output currents and/or voltages, physical properties of a load connected to the electrical converter, such as currents, voltages, fluxes, torque, speed, frequency, etc. From the assumption that the possible next switch position to be evaluated by the explorer block will be applied during the next sampling instant, the explorer block calculates the future behavior of the system variables (i.e. the system variables at future time instants) by the use of discrete-time differential equations. These equations model the electrical converter and optionally a load supplied by the electrical converter. The evolution of the system variables at future time instants may be determined for a maximal prediction horizon of a predefined length.

In the end, a cost value is determined from the system variables at future time instants with a cost function, which, for example, encodes switching losses of the electrical converter and/or a switching frequency and/or penalizes a deviation of a capacitor voltage from a desired voltage and/or penalizes a deviation of a load current from a reference current, etc.

The arbiter block is used for distributing the possible next switch position determined by the enumeration block to the explorer blocks and for finding the possible next switch position with the best (usually lowest) cost value. This possible next switch position is then applied to the semiconductor switches of the electrical converter.

The arbiter block is adapted for selecting the next switch position to be applied to the electrical converter from the possible next switch position by: receiving possible next switch position from the enumeration block; selecting a non-operating explorer block and sending a received possible next switch position to the non-operating explorer block; receiving a cost value for the respective possible next switch position from a finished explorer block (which then becomes a non-operating explorer block); when all possible next switch positions received from the enumeration block have been processed, selecting the next switch position as the possible next switch position with the best (for example lowest) cost value.

During each cycle of the controller (which is usually  performed within a shorter time interval than the sampling interval), the enumeration block determines all possible next switch positions, which are then sent to the arbiter block. Whenever the arbiter block receives a new possible next switch position (that may need to be valid), it looks for a non-operating explorer block, sends this possible next switch position to this explorer block, which then calculates the respective cost value. Meanwhile, the arbiter block may receive further possible next switch positions and may distribute them to further non-operating explorer blocks. Thus, the enumeration block, the arbiter block and the explorer blocks may operate in parallel.

When an enumeration block is ready, it sends the cost value to the arbiter block, which then may save it together with the corresponding possible next switch position as a potential next switch position, when the cost value is better than a previously saved cost value.

According to the invention, each explorer block is further adapted for determining a time interval (such as a number of time steps or sampling intervals) for the possible next switch position at which at least one of the calculated system variables at future time instants has a deviation from a reference for the system variable, which is bigger than a predefined deviation for the system variable. This time interval may be interpreted as a prediction horizon of a possible next switch position. To make sure that the system variables at future time instants stay within bounds around reference system values, the explorer block may calculate the deviation of the system variable at future time instants (which may be predicted up to a maximal prediction horizon) from the reference. When the future system variable leaves one of its bounds, its prediction horizon is set to the future time instant after which it leaves the bounds.

This may be imagined as a trajectory of the system variable at the future time instants leaving a band enclosing the reference, with this band having a width that is based on the predefined deviation.

The prediction horizon of the possible next switch position may be the minimal prediction horizon of all system variables at future time instants.

According to an embodiment of the invention, the cost function depends on the inverse of the prediction horizon of the possible next switch position. The cost function may be weighted with the number of switching transitions. In such a way, the cost value for a possible next switch position with a long horizon is usually better than the cost value of one with a short horizon. Thus, the FPGA will select switch positions for which it is likely that there are no switching transitions for the next time steps/sampling intervals.

According to the invention, a prediction horizon for a future system variable is determined via a linear extrapolation, in which the future system variable at a future time instant (for example the maximal prediction horizon) is calculated from the system variable at the current time instant. The prediction horizon is determined based on an intersection point of the linearly extrapolated system variable between the current time instant and the future time instant with a maximal possible deviation from a reference of the system variable.

According to the invention, the intersection point is determined iteratively by a binary search method. In this search, the intersection point may be found by dividing a search interval into two new search intervals and by proceeding with the interval containing the intersection point. This interval may be determined by comparing the value of the future system variable, which may be determined by interpolation with a boundary value. With a binary search, the computational costs of a division may be avoided.

According to an embodiment of the invention, the cost function is only evaluated, when the prediction horizon is longer than one time step (or sampling interval). When the prediction horizon is shorter than one sampling interval, the cost value is determined based on a predicted violation of the bounds, which is the difference between the predicted system variable and the violated bound. When the corresponding possible next switch position were applied to the converter, this would result in the physical system variables deviating from the references by more than the predefined bounds. However, sometimes it is possible that no possible next switch position is found with a prediction horizon longer than one time step. In this case, to avoid a deadlock, the arbiter block chooses the possible next switch position with the lowest violations of the bounds. This may be achieved by calculating a cost value based on the degree of the violation of a system variable at the next time step of its bound for possible next switch positions. This cost value should be rather high compared to cost values determined from the cost function.

According to an embodiment of the invention, the cost function is only evaluated, when the prediction horizon is longer than a given number of time steps.

According to an embodiment of the invention, the cost function is based on a number of switching transitions between the actual applied switch position and the possible next switch position. In such a way, the FPGA selects switch positions with a low number of switching transitions, which also results in lower switching losses.

According to an embodiment of the invention, the cost function is based on the switching losses that are predicted to occur when switching from the actual applied switch position to the possible next switch position. The switching losses often depend on the switching transition, the commutated phase current and the DC link voltage. Assuming that the DC link voltage is effectively constant, the switching losses can be predicted based on the commutated phase current and the switching transition. This information can be stored in a look-up table. In such a way, the FPGA selects switch positions that result in lower switching losses.According to an embodiment of the invention, the enumeration block comprises a look-up table providing a possible next switch position based on the actual applied switch position. It may be possible that the set of possible next switch positions is dependent on the actual applied switch position. The look-up table may provide these sets indexed by an actual applied switch position.

Furthermore, in case the electrical converter has more than one phase, there may be one look-up table that may be used for each phase of the electrical converter.

According to an embodiment of the invention, the enumeration block is adapted for determining, whether a possible next switch position is valid with respect to actual applied switch position and/or constraints on the system variables at the current time instant. For example, there may be constraints on the possible next switch positions based on a sign of an actual current. The enumeration block may determine, whether a possible next switch position is valid and may provide this information to the arbiter block. The arbiter block then only sends valid possible next switch positions to an explorer block.

According to an embodiment of the invention, the arbiter block is adapted for: sending the actual applied switch position to an explorer block for determining, whether the actual applied switch position has a prediction horizon longer than one time step; and selecting the actual applied switch position as the next switch position, when the prediction horizon is longer than the next time instant. The arbiter then sends the possible next switch position to the explorer block, only when the prediction horizon of the actual applied switch position is shorter than the next time step.

Before the possible next switch positions from the enumeration block are evaluated, the arbiter may check, whether it is possible to keep the actual applied switch position. This may be the case, when the prediction horizon of the actual applied switch position is longer than one time step/sampling interval. In this situation it is not necessary to switch at all, resulting in no switching losses, which is desirable.

According to an embodiment of the invention, the FPGA further comprises a pre-computation block adapted for calculating commonly used values and predictions from the system variables at the current time instant, wherein the commonly used values and predictions are used by the explorer blocks during the calculation of the system variables at future time instants. For example, these variables may comprise currents and/or voltages in the *abc* coordinate system, predicted reference currents and/or voltages, predicted rotor flux variables, and/or error currents with respect to a reference current.

According to an embodiment of the invention, the number of explorer blocks is smaller than a maximal number of possible next switch positions. In this case, the arbiter block may have to wait for explorer blocks to finish before a further switch position can be evaluated by the enumeration block. The number of explorer blocks may be selected based on the time an explorer block requires to evaluate a possible next switch position, and the maximal number of possible next switch positions to be evaluated during one cycle of the controller. In such a way, the capacity of the available logic blocks of the FPGA may be used in an optimal way.

According to an embodiment of the invention, by predicting the number of time steps a next switch position can be applied before one of the system variables violates a bound, by predicting the cost value associated with this next switch position and choosing the next switch position with the best (i.e. usually lowest) cost value, a sequence of future switch positions is established from the current time step until the time step at which the first system variable violates a bound. Out of this sequence of future switch positions, only the first element, i.e. the switch position at the current time step, is applied to the converter, in order to apply closed-loop feedback. At the next time-step, new measurements and/or estimates are obtained and the optimization problem described above is solved again, albeit it over a shifted horizon. This policy is referred to as receding horizon control and is a distinguishing feature of model predictive control. In summary, model predictive control combines (open-loop) constrained optimal control with the receding horizon policy that provides feedback and closes the control loop.

A further aspect of the invention relates to an electrical converter system, which comprises an electrical converter comprising a plurality of semiconductor switches for converting a DC or AC input voltage into a DC or AC output voltage of a different frequency, and a controller with an FPGA applying the next switch position determined by the FPGA to the semiconductor switches based on measurements and/or estimates of voltages and/or currents in the electrical converter provided to the FPGA. This FPGA may be designed as described above and in the following.

For example, the semiconductor switches may be IGBTs or thyristors, which are switched on and off by gate signals from the controller based on the next switch position.

It has to be understood that the converter may be a DC-DC, AC-DC, DC-AC or AC-AC converter. It may comprise one or more phases and/or may have two or more output levels. It may be a direct converter or an indirect converter with a DC link.

For example, the electrical converter may be an active neutral point clamped voltage source inverter with five output levels.

The load may be an electrical machine, such as a three-phase induction machine or a three-phase synchronous machine. Alternatively, the load may be a grid or a point of common coupling or a transformer connected to such a point of common coupling.

These and other aspects of the invention will be apparent from and elucidated with reference to the embodiments described hereinafter.

## BRIEF DESCRIPTION OF THE DRAWINGS

The subject-matter of the invention will be explained in more detail in the following text with reference to exemplary embodiments which are illustrated in the attached drawings.

Fig. 1 schematically shows a converter system according to an embodiment of the invention.

Fig. 2 schematically shows an FPGA according to an embodiment of the invention.

Fig. 3 schematically shows a component of the FPGA of Fig. 2.

Fig. 4 schematically shows a further component of the FPGA of Fig. 2.

Fig. 5 schematically shows a further component of the FPGA of Fig. 2.

Fig. 6 schematically shows a further component of the FPGA of Fig. 2.

Fig. 7 shows a diagram illustrating a binary search performed in the FPGA of Fig. 2.

The reference symbols used in the drawings, and their meanings, are listed in summary form in the list of reference symbols. In principle, identical parts are provided with the same reference symbols in the figures.

## DETAILED DESCRIPTION OF EXEMPLARY EMBODIMENTS

Fig. 1 shows an electrical converter system 10 comprising an electrical converter 12 with three phase legs 14. Each of the phase legs 14 comprises a five-level active neutral point clamped inverter 16, which is adapted to convert the DC voltage of a DC link 18 into a phase voltage of a three-phase system with the current $i_{s,abc}$. The inverter comprises eight semiconductor switches (here transistors with anti-parallel diodes) S1 to S8, which may be either opened “0” or closed “1”. The semiconductor switches S1 to S8 may encompass multiple series-connected semiconductor switches.

The converter system 10 furthermore comprises a controller 20 with an FPGA 22, which will be described below, and a load 24, such as an electrical machine.

### *Physical model of the converter*

The model of the converter 12 may be formulated in the orthogonal $\alpha\beta$ frame or in the three-phase $abc$ frame with the per unit system. A vector in the three-phase $abc$ system may be transformed to orthogonal stationary coordinates $\alpha\beta$ by applying $\xi_{\alpha\beta} = \mathbf{K}\xi_{abc}$. Here, **K** is set in such a way that the $\alpha$-axis and the $a$-axis are aligned.

$$\mathrm{K} = \frac{2}{3}\begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix}, \mathrm{K}^{-1} = \begin{bmatrix} 1 & 0 \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} \\ -\frac{1}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} \quad (1)$$

The state of the converter 12 corresponds to its actual applied switch position $s_a, s_b, s_c \in \{0,1,\dots,7\}$, its neutral point potential $\upsilon_n$ and its three phase capacitors voltages $(\upsilon_{pha}, \upsilon_{phb}, \upsilon_{phc})$, which may be interpreted as inverter state variables. We define the inverter state vector as $x_{inv} = [\upsilon_n \quad \upsilon_{pha} \quad \upsilon_{phb} \quad \upsilon_{phc}]^T$.

The continuous-time model is discretized with forward Euler discretization and the sampling interval $T_s$. The discrete-time representation is denoted with the variable $k \in \mathbb{N}$. The discrete-time equation of the phase capacitor voltage dynamic is given by

$$\upsilon_{phx}(k+1) = \upsilon_{phx}(k) + T_s\delta_{\upsilon_{ph}}(i_{s,x}(k), s_x(k)), x \in \{a,b,c\} \quad (2)$$

Similarly, the discrete-time equation of the neutral point voltage dynamic is

$$\upsilon_n(k+1) = \upsilon_n(k) + T_s\delta_{\mathrm{v_{inv}}}(\mathrm{i}_{s,abc}(k), \mathrm{s}_{abc}(k)) \quad (3)$$

In the discrete-time equations, $\delta_{\upsilon_{ph}}$ and $\delta_{\upsilon_{inv}}$ denote scalar functions with the phase currents and switch positions as arguments at time step *k*. The three-phase phase current is defined as $\mathbf{i}_{s,abc}$ = $[i_{sa} \quad i_{sb} \quad i_{sc}]^T$ and the switch position, i.e. the switch positions in each of the three *abc* phase legs, is defined as $\mathrm{s}_{abc} = [s_a \quad s_b \quad s_c]^T \in \{0,1,\dots,7\}^3$. An implementation of the controller 20 may use a sampling interval of 25*μs* or 50*μs*, for example.

*Physical model of the electrical machine*

The electrical machine 24 may be a squirrel-cage induction machine. The state-space model of a squirrel-cage induction machine in the $\alpha\beta$ reference frame is described in the following. For a model predictive direct current control method, it is convenient to choose the machine state variable

$x_m = [i_{s\alpha} \quad i_{s\beta} \quad \psi_{r\alpha} \quad \psi_{r\beta}]^T$ with the stator current $\mathbf{i}_{s,\alpha\beta}$ and the rotor flux linkage $\psi_{r,\alpha\beta}$. The model inputs are the rotor's angular velocity $\omega_r$ and the inverter switch position $\mathbf{s}_{abc}$.

The differential equation for the electrical machine 24 is stated in matrix notation. It is discretized using the forward Euler method:

$$\mathrm{x_m}(k+1) = \mathrm{A}\mathrm{x_m}(k) + \mathrm{B}\mathrm{v}(\mathrm{s}_{abc}(k)) \tag{4}$$

where

$$A = I_4 + T_s \begin{bmatrix} -\frac{1}{t_s} & 0 & \frac{X_m}{Dt_r} + T_s p \omega_r^2 & \frac{X_m}{D}\omega_r \\ 0 & -\frac{1}{t_s} & -\frac{X_m}{D}\omega_r & \frac{X_m}{Dt_r} + T_s p \omega_\mathrm{r}^2 \\ \frac{X_m}{t_r} & 0 & -\frac{1}{t_r} & -\omega_r \\ 0 & \frac{X_m}{t_r} & \omega_r & -\frac{1}{t_r} \end{bmatrix}$$

$$B = \begin{bmatrix} T_s \frac{X_r}{D} K \\ 0_{2x3} \end{bmatrix}$$

The term v($\mathbf{s}_{abc}$(k)) maps the inverter switch position into the three-phase *abc* voltage applied to the stator windings of the machine.The constants in matrix **A** are summarized in the following table:

| Parameter | Symbol |
|---|---|
| Stator resistance | $R_s$ |
| Rotor resistance | $R_r$ |
| Stator leakage reactance | $X_{ls}$ |
| Rotor leakage reactance | $X_{lr}$ |
| Magnetizing reactance | $X_m$ |
| Stator self reactance | $X_s = X_{ls} + X_m$ |
| Rotor self reactance | $X_r = X_{lr} + X_m$ |
| Determinant | $D = X_s X_r - X_m^2$ |
| Transient stator time constant | $t_s = \frac{X_r D}{R_s X_r^2 + R_r X_m^2}$ |
| Rotor time constant | $t_r = \frac{X_r}{R_r}$ |
| Model correction factor | $p = \frac{0.016569}{78.5398 * 10^{-3}}$ |

The $T_s p \omega_r^2$ term is a quadratic correction term to compensate for the error introduced by the forward Euler discretization method. It has been numerically derived by comparing forward Euler with exact Euler discretization. It is a machine-dependent parameter. Forward Euler has been chosen as a discretization method because it provides sufficient accuracy and avoids the computationally expensive exponentiation of the exact Euler approach.

*Model predictive current control method*

In the model predictive direct current control method, the selection of the switch position may be stated as a discrete optimization problem. The objective is to minimize the switching frequency (and/or the switching power losses) of the converter 12 while constraining its phase capacitor voltages, the neutral point potential and the stator current error with respect to a reference. We will now define in more detail the optimization problem to be solved.

The state vector of the converter system 10 is defined as the concatenation of the induction machine states and the converter states $\mathrm{x} = [\mathrm{x}_{\mathrm{m}}^T \quad \mathrm{x}_{inv}^T]^T$, where $x_m = [i_{s\alpha} \quad i_{s\beta} \quad \psi_{r\alpha} \quad \psi_{r\beta}]^T$ and $x_{inv} = [\upsilon_n \quad \upsilon_{pha} \quad \upsilon_{phb} \quad \upsilon_{phc}]^T$. The components of the state vector may be interpreted as system variables.

In general, system variables may be either measured or determined for the current time instant (system variables at the current time instant) and/or they may be predicted or calculated for future time instants (system variables at future time instants).

The system variables to be kept within bounds are captured by the output vector $\mathrm{y} = [i_{sa} \quad i_{sb} \quad i_{sc} \quad \upsilon_n \quad \upsilon_{pha} \quad \upsilon_{phb} \quad \upsilon_{phc}]^T$. The input vector is defined as the switch position $\mathbf{s}_{abc}$. It is assumed that the rotor and stator speed $\omega_r$ and $\omega_s$ respectively vary slowly, allowing one to consider them to beconstant within the prediction horizon.

In the remainder, to simplify the notation, the switch position $\mathbf{s}_{abc}$, which is a vector composed of “0” and “1”, will often be reduced to $\mathbf{s}$.

In this control method, the current is directly controlled by explicitly bounding it in the optimization problem statement. By setting the width of the bound on the stator current error, we can directly control the Total Harmonic Distortion (THD) of $\mathbf{i}_{s,abc}$. The following equations detail the discrete optimization problem to be solved.

$$J^* = \min_{\mathrm{S(k)}} \frac{1}{NT_s} \sum_{l=k}^{k+N-1} \delta_s(\mathrm{s}(l-1), \mathrm{s}(l), \mathrm{i}_s(l)) + \lambda_n (\upsilon_n(N))^2 \tag{5a}$$

$$subjto\, \mathrm{x}_{\mathrm{m}}(l+1) = \mathrm{A}\mathrm{x}_{\mathrm{m}}(l) + \mathrm{B}\mathrm{v}(\mathrm{s}_{abc}), \tag{5b}$$

$$x_{inv}(l+1) = x_{inv}(l) + T_s\delta_\upsilon(i_{s,abc}(l), s_{abc}(l)), \quad (5c)$$

$$y(l) = Cx(l) \quad (5d)$$

$$|y^*(l) - y(l)| \leq y_{err}, \quad (5e)$$

$$s(l) \in S(l) \subset \{0,1,\ldots,7\}^3 \quad (5f)$$

$$\forall l = k,\ldots, k+N-1 \quad (5g)$$

Where,

$$C = \begin{bmatrix} K^{-1} & 0_{3x6} \\ 0_{4x4} & I_4 \end{bmatrix}, \in \mathbb{R}^{7x8}$$

To evaluate the short-term switching frequency or switching power losses, the cost function sums up the number of switching transitions or switching energy losses $\delta_s()$ between two successive time steps over all the switching sequence **S**($k$) = [**s**$^T$($k$) **s**$^T$($k$ + 1)…**s**$^T$($k$ + $N$ – 1)]$^T$. This sum is then divided by the prediction horizon length $N$ of **S**($k$).

The last term in the cost function is used to reduce the likelihood of deadlocks. Indeed, when the neutral point potential deviates too much from zero, the risk of deadlocks may increase quickly. A deadlock occurs when no switching sequence **S**($k$) exists that meets (5e). The weight $\lambda_n$ is used to fine tune the deadlock avoidance.

The two first constraints (5b), (5c) define the state evolution of the system 10 (the converter and the electrical machine) as explained before. The sampled system state **x**($k$) and the actual switch position **s**($k$ – 1) of the converter 12 are used as a starting point for the predictions.

The third constraint (5d) builds the output vector from a subset of the system vector and the appropriate transformation.

The fourth constraint (5e) bounds the output vector **y**($l$) around its reference **y**$^*$($l$). The reference **y**$^*$($l$) is time varying for the phase capacitor voltages and the stator current. Hence, they are also an input to the optimization problem. **y**$_{err}$ determines the width of the error bounds around the references.

In the fifth constraint (5f), $S(l)$ is the set of allowed switch positions at time instant $l$.

*Concepts implemented in the FPGA*

In (5a), the optimization problem may become computationally intractable as $N$ increases. This is because at each time step a large number of new switch positions is available that needs to be evaluated by the algorithm. Several heuristics may allow one to limit the computational burden without significantly impacting the performance.

The following concepts have been implemented in the FPGA 22 to increase its performance and to reduce the computational burden:

*1) Lazy evaluation:*

A switching transition should be triggered only when at least one of the output variables exceeds its bound or is predicted to exceed its bound within the sampling interval. That is, at time-step *k*, if the output vector is predicted to remain within its bounds $|y^*(k+1)-y(k+1)| \leq y_{err}$ when applying the previously applied switch position again, then maintain the previous switch position and set **s**(*k*) = **s**(*k*-1). Thus it is possible to minimize the switching frequency (and thus the switching losses) by delaying switching as much as possible.

*2) Switching horizon:*

A switching sequence may be interpreted as a succession of switching decisions at discrete time-steps. The switching horizon is a string composed of "E" (extension) steps and "S" (switching) steps. The extension step maintains the previous switch position until a bound is violated. The trajectory of the system output vector y is predicted by (2), (3) and (4). When a bound is hit, the controller 20 may proceed to a switching step. That is, the controller 20 needs to enumerate all the admissible switching transitions from the no longer feasible switch position to a new candidate switch position. An example, which will be explained in more detail below, is the "SE" string. Here, the controller 20 first enumerates all valid switch positions **s**(k) as new switching sequences and then extends each of them until they hit their respective boundaries. Commonly used switching horizons include SE, SSE, SESE, SSESE and so on. SSESE for example, implies that switching is considered at time steps *k* and *k*+1, followed by an extension step, another switching transition and a second extension step.

*3) Extrapolation*

One approach is to predict the system evolution by incrementally evaluating the system model at each time step. The advantage of doing so is the high accuracy of the procedure. However, it is computationally expensive to evaluate it when the prediction horizons are long. An alternative is, for a given switch position, to evaluate the system model at several points in time, such as *k, k + l, k + 2l,* where *l* is a positive integer, and to then approximate the underlying function by quadratic or linear interpolation. This method requires far less computations and is sufficiently accurate for the prediction horizons typically observed.

*FPGA implementation*

Fig. 2 shows an overview of the blocks of the FPGA 22. As stated above, the "SE" prediction horizon is implemented in the FPGA 22. Thus, the FPGA 22 must first enumerate all possible next switch positions at time step *k* and then extend each of them.

To address the goal of scalability and simplicity, the architecture of the FPGA 22 is based on dynamic scheduling. The FPGA 22 comprises a central scheduler block 26 that samples the system state $\mathbf{x}(k)$ into registers and also enumerates and distributes possible next switch positions $s^{new}$ to a number of $n$ explorer blocks 28.

The scheduler block comprises a pre-computation block 30, which pre-computes values that, for example, are needed in the explorer blocks 28, an enumeration block 32, which generates the possible next switch positions $s^{new}$ and an arbiter block 34, which distributes the possible next switch positions $s^{new}$ to the explorer blocks 28.

The system state $\mathbf{x}(k)$ together with a switching sequence may be defined as a node in the solution space to be explored. The explorer blocks 28 evaluate these nodes, by assessing their prediction horizon $N$ through an extension step performed by an extension block 36 and then evaluate their cost value $J$ in a cost evaluation block 38.

Once the exploration has completed its computations for a node, it is returned to the scheduler 26, in particular to the arbiter block 34. Only the switch position of the node with the lowest cost value $J^*$ is maintained by the arbiter block 34. At the end of the enumeration and evaluation process, the scheduler returns the best switch position $\mathbf{s}(k)$ to the external system.

The following table summarizes the hardware characteristics of each block.

| Block Name | Mult-16 | Add-16 | Max. cycles | FPGA Slices |
|---|---|---|---|---|
| **Scheduler 26** | **6** | **7** | **47+64·34/n** | **1500** |
| Pre-computation 30 | 4 | 6 | 13 | |
| Enumeration 32 | 0 | 0 | 64 | |
| Arbiter 34 | 2 | 1 | 64·34/$n$ | |
| **Explorer 28** | **4** | **14** | **34** | **3300** |
| Extension 36 | 4 shared | 7 | 28 | |
| Cost evaluation 38 | 4 shared | 7 | 6 | |

The total hardware cost of the FPGA 22 depends on the number $n$ of explorer blocks *28* instantiated. The worst case execution time in FPGA cycles is linearly dependent on the number of explorer blocks 28: $13+34+64 \cdot 34/n$ cycles.

Indeed, the FPGA 22 will first pre-compute the model, operate the lazy evaluation and then evaluate all the possible nodes through the $n$ explorer blocks 28. The explorer blocks 28 are expensive in terms of area usage. This is mainly due to the complexity involved in the extension step as explained below.

*Scheduler block: model pre-computation*

Fig. 3 shows the pre-computation block 30. The FPGA 22 can spare execution time by pre-computing some elements in common to all explorer blocks 28 with the pre-computation block 30.

The discrete state-space model of formulas (5b) and (5c) is divided into two parts. The part in common for all explorer blocks 28 is computed in the pre-computation block 30. The part of the model specific to each node is evaluated by the explorer blocks 28 themselves.

The input to the pre-computation block 30 is the system state **x**($k$), the stator and rotor rotational speeds $\omega_s(k)$, $\omega_r(k)$ and the current reference $i^*_{s,\alpha\beta}$ ($k$) at time instant $k$. The following equations describe how the output variables of the pre-computation block 30 are computed.

First, the open loop part $\mathrm{i}^{errOL}_{s,abc}(k+l)$ of the current error is computed. This equation is detailed in several steps: As a first step, the stator current and its reference at instant $k$ + $l$ are computed in the $\alpha\beta$ reference frame:

$$\mathrm{i}_{s,\alpha\beta}(k+l) = \mathrm{A}_1\,\mathrm{i}_{s,\alpha\beta}(k) + \mathrm{A}_2\,\psi_{r,\alpha\beta}(k) + \frac{X_r}{D} lT_s\,\mathrm{Kv}(s(k)) \tag{6}$$

$$\mathrm{i}^*_{s,\alpha\beta}(k+l) = \mathrm{A}_0\mathrm{i}^*_{s,\alpha\beta}(k) \tag{7}$$

Where,

$$\mathrm{A}_0 = \mathrm{I}_2\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}\omega_s lT_s, \mathrm{A}_1 = (1-\frac{1}{t_s}lT_s)\mathrm{I}_2,$$

$$\mathrm{A}_2 = \begin{bmatrix} \frac{1}{t_r} & \omega_r \\ -\omega_r & \frac{1}{t_r} \end{bmatrix}\frac{X_m}{D}lT_s + p(lT_s\omega_r)^2\,\mathrm{I}_2$$

The stator current reference $i^*_{s,\alpha\beta}$ is rotated forward with the stator frequency $\omega_s$. $\mathbf{I}_2$ is the identity matrix of size two.

The stator current error is then stated in *abc* coordinates:

$$\begin{aligned}\mathrm{i}^{err}_{s,abc}(k+l) &= \mathrm{K}^{-1}(\mathrm{i}^*_{s,\alpha\beta}(k+l) - \mathrm{i}_{s,\alpha\beta}(k+l)) \\ &= \underbrace{\mathrm{i}^{errOL}_{s,abc}(k+l)}_{Commonpart} + \underbrace{\mathrm{B}_0\mathrm{v}(\mathrm{s}(k))}_{Individualpart}\end{aligned} \tag{8}$$

$$\mathrm{B}_0 = \mathrm{K}^{-1}(\frac{X_r}{D}lT_s)\mathrm{K}$$

With the chosen system model, the $\mathrm{i}^{errOL}_{s,abc}(k+l)$ part of the current error is common to the explorer blocks 28. The second part of the equation relates to the possible next switch position **s**($k$) specific to the node. Hence this part is computed in the explorer blocks 28. The first part of the equation is computed in block 30. It may be called open-loop current error at time step $k$+$l$. It is computed in $\alpha\beta$ and then transformed into *abc* coordinates:

$$\mathrm{i}^{err\,OL}_{s,abc}(k+l) = \mathrm{K}^{-1}(\mathrm{A}_0\,\mathrm{i}^*_{s,\alpha\beta} - \mathrm{A}_1\,\mathrm{i}_{s,\alpha\beta}(k) - \mathrm{A}_2\,\psi_{r,\alpha\beta}(k)) \tag{9}$$

The second equation to be pre-computed is the stator current error at time step *k*:

$$\mathrm{i}_{s,abc}^{err}(k) = \mathrm{K}^{-1}(\mathrm{i}_{s,\alpha\beta}^{*}(k) - \mathrm{i}_{s,\alpha\beta}(k)) \tag{10}$$

The third and last variable to be pre-computed is the stator current at time step *k* in *abc* coordinates:

$$\mathrm{i}_{s,abc}(k) = \mathrm{K}^{-1}\,\mathrm{i}_{s,\alpha\beta}(k) \tag{11}$$

From the implementation point of view, a minimal number of adders and multipliers was used, while the number of cycles to compute the aforementioned equations was minimized.

*Scheduler block: enumeration*

The second way to leverage centralization is achieved by the scheduler 26, which manages enumeration and distribution of possible next switch positions $s^{new}$. This is done with the help of the two blocks 32, 34 operating in parallel.

The enumeration block 32 enumerates and filters the allowed switch positions $s^{new}$.

The enumeration block 32 takes as input the actual applied switch position **s**(*k*–1) of the converter 12. With this information, it is possible to enumerate all the admissible next switch positions $S_x(k)$ for each phase leg 14 separately. Thus, each phase leg 14 has a set of possible next switch positions $S_x(k) \subset \{0,1,..7\}, x \in \{a,b,c\}$. These sets are stored in look-up tables 40 and selected depending on $s_x(k–1)$. By construction of the converter 12, none of the sets has more than four elements. The elements in the sets of each phases are then combined in combination block 42 to form any possible switch positions $\mathbf{s}^{new}(k)$.

Due to the clamping constraints of the converter 12, several switching transitions are disallowed. The clamping constraints depend on the direction of the current $\mathbf{i}_{s,abc}(k)$ on each phase leg 24 as well as the switching transition $\mathbf{s}(k–1) \rightarrow \mathbf{s}^{new}(k)$. A constraint detector block 44, which may be implemented as a combinatorial circuit, detects any invalid transition given the previous information. The constraint detector block 44 outputs a corresponding binary value “valid”.

The enumeration block 32 is managed by a finite state machine, which increments the counter controlling an enumeration multiplexer until a valid possible next switch position $s^{new}$ is detected by the constraint detector block 44. The finite state machine starts searching when the “next” signal is asserted.

Fig. 5 shows the relationship between the enumeration block 32 and the arbiter block 34. The arbiter block 34 arbitrates the exploration of the enumerated switch positions $s^{new}$. The arbiter block 34 sequentially distributes the enumerated switch positions $s^{new}$ to the explorer blocks 28, gathers their results and maintains the best switch position **s**(*k*) according to its cost $J^*$.

*Scheduler block: arbiter*

The arbiter block comprises two finite state machines 46, 48 running in parallel.

The first finite state machine 46 operates in two stages. In a first stage, the finite state machine 46 makes a “lazy evaluation” by sending $\mathbf{s}(k\text{–}1)$ to an explorer block 28 as a possible next switch position $s^{new}(k)$ to be extended. If the prediction horizon $N$ of this possible next switch position is greater or equal to one sampling interval, the arbiter block 34 stops and returns $\mathbf{s}(k\text{–}1)$ as the next switch position $\mathbf{s}(k)$ for the new time instant. Otherwise, the second stage of the finite state machine 46 starts. In this stage, the finite state machine 46 samples possible next switch positions $\mathbf{s}^{new}(k)$ from the enumeration block 32 and sends the $\mathbf{n}_i = [\mathbf{s}^{new}(k),\ \delta_s(k),\ N_{min}]$ information to any available (i.e. non-operating) explorer block 28 (indexed by $i$ in Fig. 5). The finite state machine 46 continues until all the $\mathbf{s}^{new}(k)$ have been evaluated.

The second finite state machine 48 polls each explorer block 28 for available results $\mathbf{r}_i = [\mathbf{s}^{new}(k),\ N,\ J]$. If the cost $J$ of $\boldsymbol{r}_i$ is smaller than the current best $J^*$, $\mathbf{r}_i$ is saved as the best next switch position $\mathbf{s}(k)$. Otherwise, the result is dropped. Once all the results have been gathered, the best switch position $\mathbf{s}(k)$ can be returned as the switching decision for the new time instant.

*Explorer block*

Fig. 6 shows one explorer block 28. An explorer block 28 can be considered as a computation unit. For the input data, the scheduler block 26 provides the node specific information $\mathbf{n}_i = [\mathbf{s}(k),\ \delta_s(k),\ N_{min}]$ and also the pre-computed model shared by all nodes $i_{s,abc}(k), i^{err}_{s,abc}(k), i^{errOL}_{s,abc}(k+l), \upsilon_n(k), \mathrm{v}_{ph,abc}(k)$. The explorer block 28 is composed of the two blocks 36, 38 each one having a computation step to accomplish. The extension block 36 first computes the prediction horizon $N$. Then, the cost evaluation block 38 applies the cost function $J$ on the explored node. To optimize hardware usage, the two blocks 36, 38 may share 4 common multipliers.

*Explorer block: extension with linear extrapolation*

To apply the extension step, the extension block 36 goes through several steps. It first evaluates the system state evolution at time step $k + l$ by computing the stator current error (8) and the inverter voltages (2) and (3). After this step, the explorer knows the system output y at time steps $k$ and $k + l$. Thus, it can now build a linear approximation of the system evolution:

$$y(k+j) = \frac{j\delta_y}{l} + y(k),\ \ \delta_y = y(k+l) - y(k) \qquad (12)$$

Fig. 7 shows the linear approximation of one variable as dashed line.

Now that the model is fully available, the extension step can evaluate the time step *N* when one of the 7 components first hits its boundary:

$$N = \min_{i \in \{1,2,\ldots 7\}}(N_i) \tag{13}$$

$$N_i = l\frac{y_{i,bnd} - y_i(k)}{\delta_{yi}} \tag{14}$$

$$y_{i,bnd} = \begin{cases} \overline{y_i} & if\ \delta_{yi} \geq 0 \\ \underline{y_i} & else \end{cases}$$

In here, we use the index *i* to denote the *i*th component of the output vector y. The output vector **y** is composed of the current error $\mathrm{i}_{s,abc}^{err}$ and the inverter voltages $\upsilon_n, \mathrm{v}_{ph,abc}$. The variables $\mathrm{i}_{s,abc}^{err}$ and $\upsilon_n$ are bounded by constant values, while the capacitor voltages $\mathbf{v}_{ph,abc}$ might have time-varying boundaries that depend on $\upsilon_n$. Thus, $\overline{\mathrm{v}}_{ph}$ and $\underline{\mathrm{v}}_{ph}$ are parameters that are provided as input. The upper or lower bounds are enforced depending on the signs of the individual component slopes $\delta_{yi}$. The output vector **y** is composed of 7 components with each one having their own linear model to be evaluated. The objective is to find the time instant at which $y_i(N_i) = y_{i,bnd}$. One possibility is to use (14) for evaluating the prediction horizon $N_i$ of each component. However, an integer divider may be rather expensive in terms of area and computation delay. Another possibility is to use a binary search method, which exploits the shared multipliers and requires only two more adders.

The binary search, which is described with respect to Fig. 7, first evaluates the linear function at $N_{max}$, which may be a predefined prediction horizon. If the result is within the bound $y_{i,bnd}$, $N_{max}$ is returned as the prediction horizon $N_i$. Otherwise the search starts within the interval between the actual time instant *k* and $k$+$N_{max}$. At each step, the binary search divides the time interval into two subintervals and continues with the subinterval that comprises the point at which the bound equals the linearly approximated output variable. When the time interval is less than $N_{min}$, the computation is aborted and the prediction horizon $N_i$ for the variable indexed with *i* is set to 0, wherein $N_{min}$ is a minimal horizon length as explained below.

For example, in Fig. 7, the linear approximation (dashed line) is evaluated four times, before the boundary crossing is located.

The binary search has the advantage that it relies only on one multiplier and two adders to compute (14) because it evaluates the linear model (12) at different points in time instead of using a division. The binary search reduces the size of its search interval by two at each iteration. In theory, the worst case of the number of iterations required is $O(\log_2(N_{max}))$.

The binary search may furthermore use upper and lower bounds to reduce the computation time.

For example, an upper bound $N_{max}$ on the interval to be explored can be imposed. $N_{max}$ may be set to the current shortest prediction horizon among the components $y_i$ that have already been evaluated. If the evaluated component $y_i$ has a horizon of $N_i > N_{max}$, then the binary search on $y_i$ stops since the *i*th component is not critical (i.e. the violation of its bound is predicted to occur after another component violates its bound). Before any evaluation, $N_{max}$ may be initialized to 32, for example, thus providing real time guarantees even in the worst case.

Furthermore, a lower bound $N_{min}$ may be imposed, which may be provided by the scheduler block 26. It is the minimum horizon length $N_{min}$ to be accomplished by the evaluated possible next switch position on the explorer block 28 in order to become the best candidate. During the extension process, the explorer block 28 can find out that the minimum length cannot be met while performing the binary search. If this is the case, the computation is aborted and the evaluated possible next switch position is signalled as abandoned to the scheduler block 26, for example by setting its cost to a negative value $J = -1$ or setting the prediction horizon to 0. $N_{min}$ may be determined for each possible next switch position based on $J^*$ of the current best possible next switch position observed by the scheduler block 26 and the specific switching transition $\mathbf{s}(k-1) \rightarrow \mathbf{s}(k)$ to explore:

$$N \geq N_{\min} = \frac{\delta_s(\mathrm{s}(k-1), s(k), \mathrm{i}_s(l))}{J^*} \tag{15}$$

The previous equation ignores the neutral point voltage $\upsilon_n$ of (5a) for ease of computation. Note that the $N_{min}$ heuristic can speed-up the average computation time but does not give any guarantees about the worst case execution time.

The objective is to find the greatest integer $N_i$ for each of the linear models to be evaluated (by $i = 1,2,\dots,7$) such that the three conditions below hold:

$$N_{min} \leq N_i \leq l \tag{15a}$$

$$y_i(k) + N_i\delta_{yi} \leq \overline{y_i} \tag{15b}$$

$$y_i(k) + N_i\delta_{yi} \geq \underline{y_i} \tag{15c}$$

The binary search method is used to find $N_i$. Denote the $m$-th bit of $N_i$ as $N_i[m]$, with $N_i[1]$ being the least significant bit. Let $a$, $b$, $c$, and $s$ be temporary variables to be used in the binary search procedure. The $m$-th bit of $s$ is denoted by $s[m]$, with $s[1]$ being the least significant bit. In the algorithm $a$ is used as a candidate solution, $b$ is used as a lower bound of the solution, $c$ is used to store the results of a division by 2 (implemented as a right shift), and $s$ is an upper bound of the solution. The algorithm to obtain $N_i$ uses $M = \log_2 l$ iterations, described below.

Initialization:

$$N_i = 0, \qquad b = 0, \qquad c = \delta_y, \qquad s = l - 1$$

Loop:

```
for $m = M, \dots, 1$
    $c := \frac{c}{2}$
    $a := b + c$
    if $(y_i(k) + a \leq \overline{y_i})$ and $(y_i(k) + a \geq \underline{y_i})$, then
        $b := a$
        $N_i[m] := 1$
    else
        $s[m] := 0$
        if $s < N_{min}$, then
            stop for loop
        end if
    end if
end for
```

Notice that if the loop is stopped because $s < N_{min}$, then no integer greater than $N_{min}$ satisfies (15b) and (15c).

The standard solution for the problem (13) would involve a numeric long division to solve (14). The division requires $X$ iterative steps, where $X$ is the number of bits of the numerator, and therefore is performed over at least $X$ FPGA clock cycles.

The proposed method presents the advantage that fewer clock cycles are necessary to obtain the result. The proposed algorithm takes at most $\log_2 l$ iterations to execute. Since $\log_2 l < X$, fewer iterations are required in the proposed method. Furthermore, the algorithm features an early detection of when a solution for the problem does not exist in the cases that the upper bound of the solution $s$ is lesser than $N_{min}$. Fewer clock cycles means that the solution for the problem (13) is found faster.

The maximum efficiency is achieved when the maximum horizon length $l$ is a power of 2. In several places we say that we evaluate 7 components or the problem. The number 7 applies to this specific topology. Perhaps we can generalise this number 7.

*Explorer block: cost function evaluation*

After the extension step, the cost value *J* may be computed by the cost evaluation block 38 in two different ways.

If the evaluated next best switch position has a prediction horizon $N \geq 1$, all its output components $y_i$ are either within their boundaries or their trajectories are approaching the reference. Thus, (5a) is used for calculating the cost value *J*.

Otherwise, if $N = 0$, a deadlock has occurred and a possible next switch position needs to be found that resolves this deadlock. Its cost value *J* may then be defined as the normalized component with the maximum boundary violation at the time step $k$+1:

$$J_{dl} = \max_{i\in\{1,2,\ldots 7\}} \left( \frac{\left| y_{i,bnd} - y_i(k+1) \right|}{\overline{y_i} - \underline{y_i}} \right. \tag{16}$$

In this second situation we say that $J_{dl}$ is a deadlock cost. By minimizing $J_{dl}$ the worst case violation is minimized.

For the FPGA 22, a deadlock situation would occur when all the evaluated next switch positions have a prediction horizon $N = 0$. That is, they all have failed to fulfil condition (5e) at time step $k + 1$. Nevertheless, the FPGA 22 still needs to take a switching decision. In this case, the policy is to choose as the next switch position the one that is predicted to yield the lowest boundary violation at time step $k + 1$. By construction, the FPGA 22 is always able to choose a switch position. Furthermore, even in the deadlock case, the best deadlock switching is monitored by the scheduler block 26.

As output, the explorer block 28 provides the switch position $\mathbf{s}(k)$ that was investigated together with its prediction horizon *N* and the related cost value *J*.

While the invention has been illustrated and described in detail in the drawings and foregoing description, such illustration and description are to be considered illustrative or exemplary and not restrictive; the invention is not limited to the disclosed embodiments. Other variations to the disclosed embodiments can be understood and effected by those skilled in the art and practising the claimed invention, from a study of the drawings, the disclosure, and the appended claims. In the claims, the word “comprising” does not exclude other elements or steps, and the indefinite article “a” or “an” does not exclude a plurality. A single processor or controller or other unit may fulfil the functions of several items recited in the claims. The mere fact that certain measures are recited in mutually different dependent claims does not indicate that a combination of these measures cannot be used to advantage. Any reference signs in the claims should not be construed as limiting the scope.

LIST OF REFERENCE SYMBOLS

| | |
|---|---|
| 10 | converter system |
| 12 | converter |
| 14 | converter phase |
| 16 | ANPC inverter |
| 18 | DC link |
| 20 | controller |
| 22 | FPGA |
| 24 | load |
| 26 | scheduler block |
| 28 | explorer block |
| 30 | pre-computation block |
| 32 | enumeration block |
| 34 | arbiter block |
| 36 | extension block |
| 38 | cost evaluation block |
| 40 | look-up table |
| 42 | combination block |
| 44 | constraint detector block |
| 46 | first finite state machine |
| 48 | second finite state machine |

## CLAIMS

1. An FPGA (22) for controlling an electrical converter (12), the FPGA (22) comprising:
an enumeration block (32) adapted for generating possible next switch positions ($s^{new}(k)$) for semiconductor switches of the electrical converter (12) based on an actual applied switch position ($s(k-1)$);
a plurality of explorer blocks (28), each explorer block (28) adapted for calculating a cost value (*J*) for a possible next switch position of the semiconductor switches by:
receiving a possible next switch position ($s^{new}(k)$);
calculating system variables at future time instants from system variables at a current time instant of the electrical converter (12) based on the possible next switch position, wherein the system variables at future time instants are calculated from the system variables at the current time instant with differential equations modelling the electrical converter (12) and the load (24);
determining a cost value (*J*) from the system variables at future time instants by evaluating a cost function with the system variables at future time instants;
an arbiter block (34) for selecting the next switch position ($s(k)$) to be applied to the electrical converter (12) from the possible next switch positions by:
receiving possible next switch positions ($s^{new}(k)$) from the enumeration block (32);
selecting a non-operating explorer block (28) and sending a received possible next switch position to the non-operating explorer block;
receiving a cost value (*J*) for the respective possible next switch position from a finished explorer block;
when all possible next switch positions received from the enumeration block (32) have been processed, selecting the next switch position ($s(k)$) as the possible next switch position ($s^{new}(k)$) with the lowest cost value (*J*);
characterized in
that each explorer block (28) is further adapted for:
determining a prediction horizon (*N*) for the possible next switch position ($s^{new}(k)$) at which at least one of the calculated system variables at future time instants has a deviation from a reference for the system variable, which is bigger than a predefined deviation for the system variable;
that a prediction horizon (N) for a system variable at future time instants is determined via a linear extrapolation, in which the system variable at the future time instants is calculated from the system variable at the current time instant, and the prediction horizon (N) is determined based on an intersection point of the linearly extrapolated system variable between the

current time instant and the future time instant with a maximal possible deviation from a reference of the system variable; and
that the intersection point is determined iteratively by a binary search.

2. The FPGA (22) of claim 1,
wherein the cost function depends on the inverse of the prediction horizon ($N_p$).

3. The FPGA (22) of one of claims 1 or 2,
wherein the cost function is only evaluated, when the prediction horizon ($N$) is longer than one time step;
wherein, when the prediction horizon ($N$) is not longer than one time step, the cost value ($J$) is determined based on a violation of a maximal allowed deviation from a reference of the system variable at future time instants.

4. The FPGA (22) of one of the preceding claims,
wherein the cost function is based on a number of switching transitions between the actual applied switch position and the possible next switch position ($s^{new}(k)$); and/or
wherein the cost function is based on the switching losses incurred when switching between the actual applied switch position and the possible next switch position ($s^{new}(k)$).

5. The FPGA (22) of one of the preceding claims,
wherein the enumeration block (32) comprises a look-up table (40) providing a possible next switch position ($s^{new}(k)$) based on the actual applied switch position.

6. The FPGA (22) of one of the preceding claims,
wherein the enumeration block (32) is adapted for determining, whether a possible next switch position ($s^{new}(k)$) is valid with respect to the actual applied switch position and/or constraints on the system variables at the current time instant; and/or
wherein the arbiter block (34) only sends valid possible next switch positions to an explorer block (28).

7. The FPGA (22) of one of the preceding claims,
wherein the arbiter block (34) is adapted for:
sending the actual applied switch position ($s(k-1)$) to an explorer block (28) for determining, whether the actual applied switch position ($s(k-1)$) has a prediction horizon ($N$) of at least one time step;

selecting the actual applied switch position (s*(k-1)*) as the next switch position (s*(k)*), when the prediction horizon (*N*) is longer than one time step.

8. The FPGA (22) of one of the preceding claims, further comprising:
a pre-computation block (30) adapted for calculating commonly used values and predictions based on the system variables at the current time instant, wherein the commonly used values and predictions are used by the explorer blocks (28) during the calculation of the system variables at future time instants.

9. The FPGA (22) of one of the preceding claims,
wherein the number of explorer blocks (28) is smaller than a maximal possible number of next switch positions ($s^{new}(k)$).

10. An electrical converter system (10), comprising:
an electrical converter (12) comprising a plurality of semiconductor switches for converting a DC or AC input voltage into a DC or AC output voltage of a different frequency;
a controller (20) with an FPGA (22) according to one of the preceding claims for applying the next switch position determined by the FPGA (22) to the semiconductor switches.

11. The electrical converter system (10) of claim 10,
wherein the electrical converter (12) comprises an active neutral point clamped voltage source inverter.

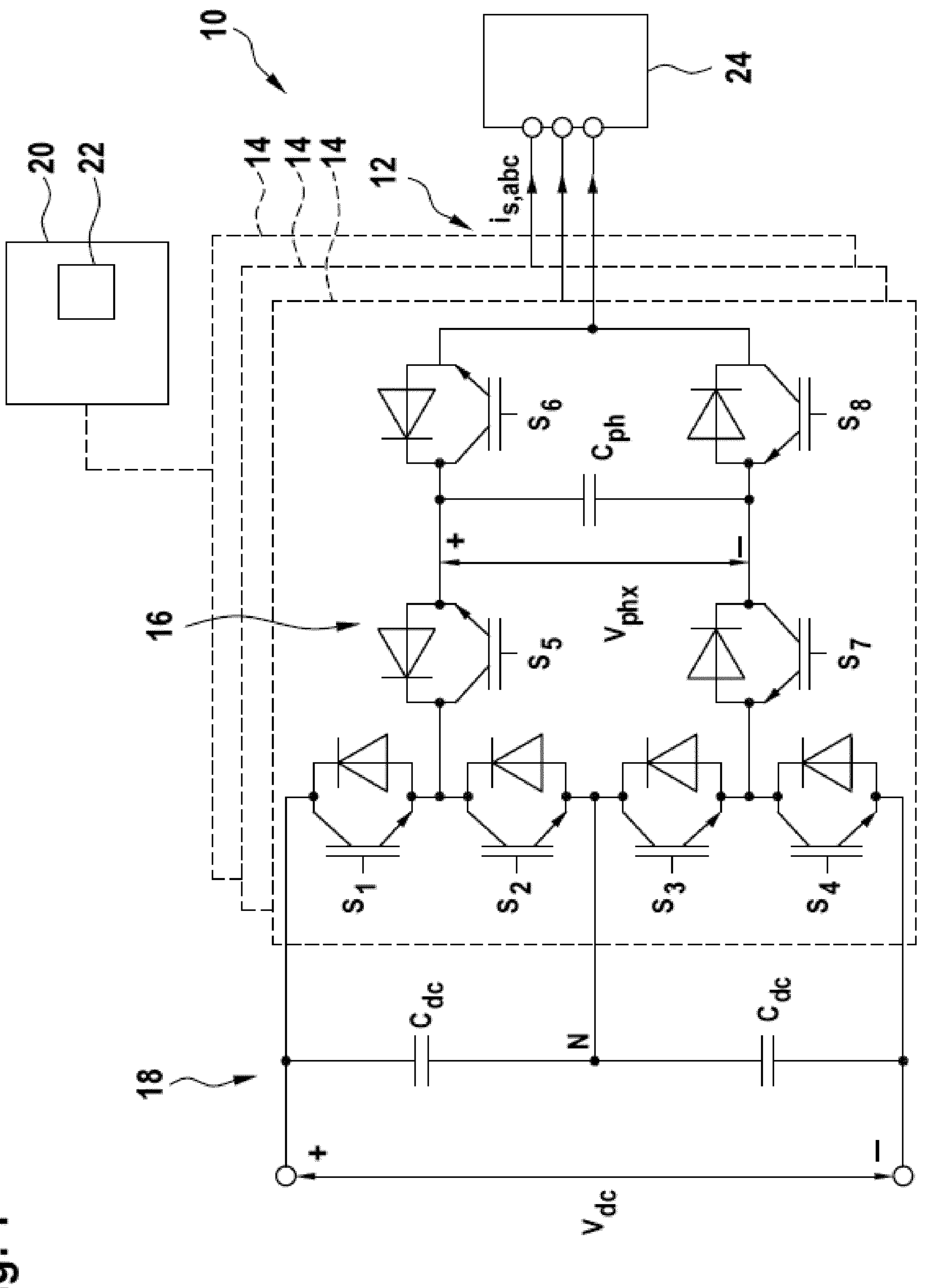

10
12
14
14
14
16
18
20
22
24
$i_{s,abc}$
$S_1$
$S_2$
$S_3$
$S_4$
$S_5$
$S_6$
$S_7$
$S_8$
$C_{ph}$
$V_{phx}$
$C_{dc}$
$C_{dc}$
N
$V_{dc}$
+
–
+
–

Fig. 1

Fig. 2

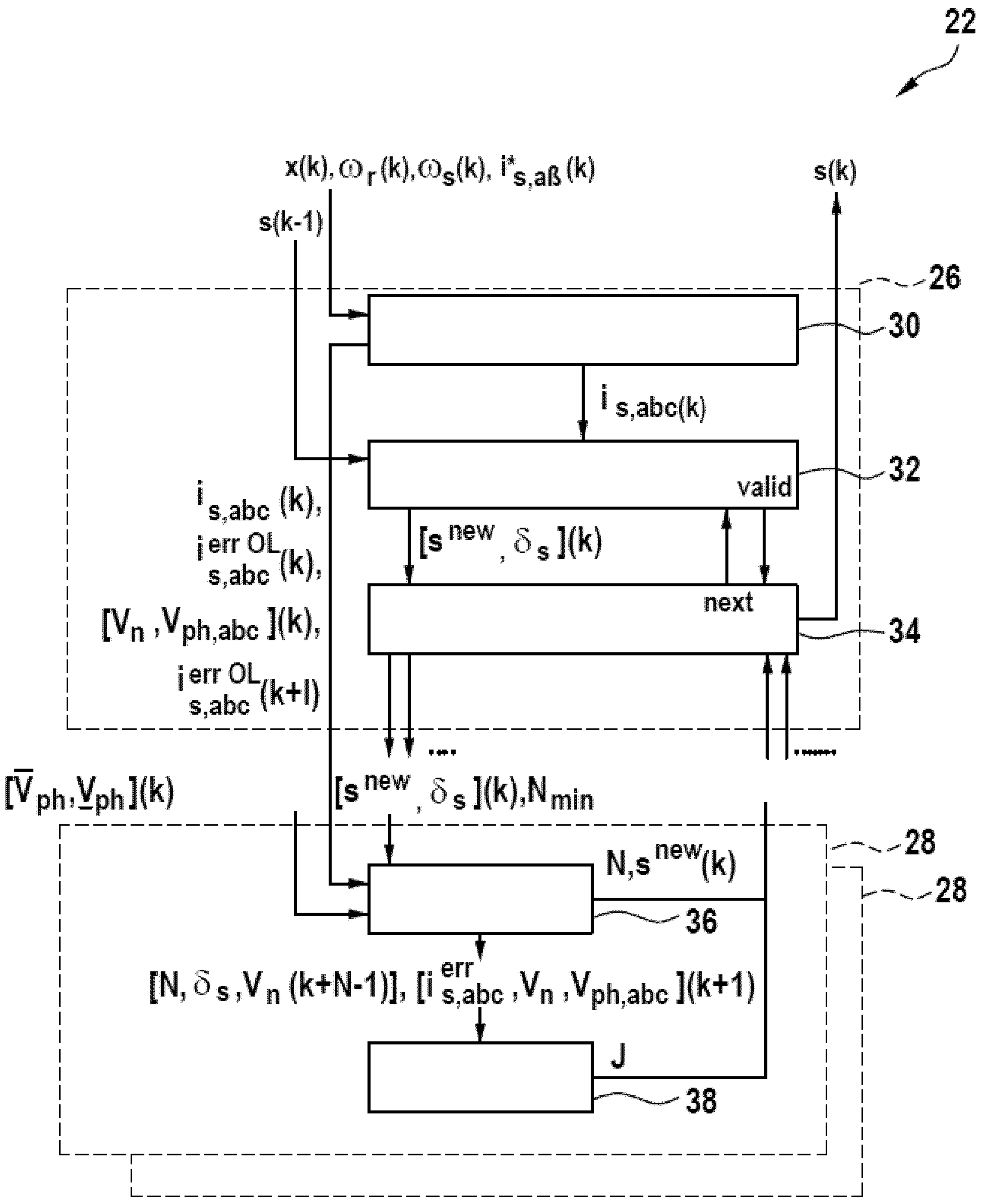

22
$x(k), \omega_r(k), \omega_s(k), i^*_{s,a\beta}(k)$
$s(k)$
$s(k-1)$
26
30
$i_{s,abc(k)}$
32
valid
$i_{s,abc}(k)$,
$i^{err\,OL}_{s,abc}(k)$,
$[V_n, V_{ph,abc}](k)$,
$i^{err\,OL}_{s,abc}(k+l)$
$[s^{new}, \delta_s](k)$
next
34
$[\bar{V}_{ph}, \underline{V}_{ph}](k)$
$[s^{new}, \delta_s](k), N_{min}$
28
28
$N, s^{new}(k)$
36
$[N, \delta_s, V_n(k+N-1)], [i^{err}_{s,abc}, V_n, V_{ph,abc}](k+1)$
J
38

Fig. 3

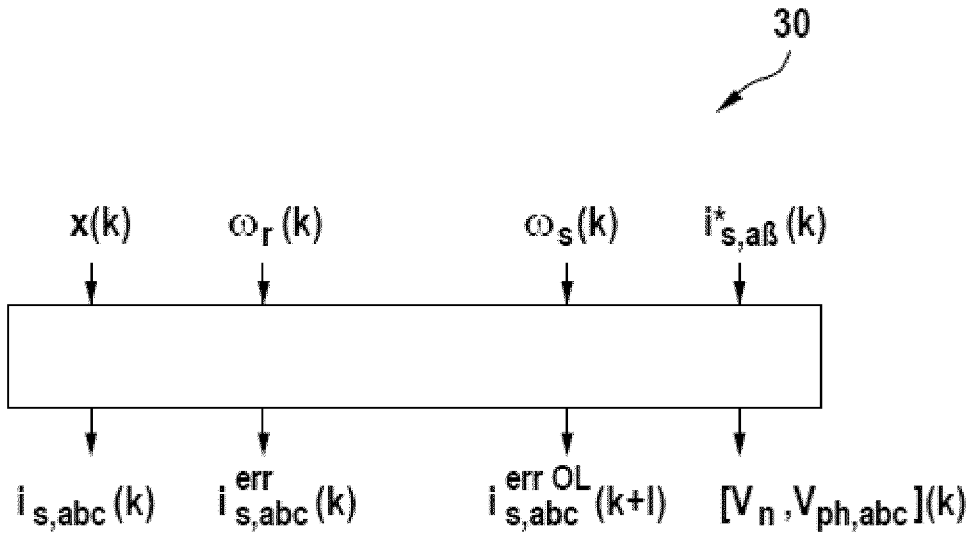

30
$x(k)$
$\omega_r(k)$
$\omega_s(k)$
$i^*_{s,a\beta}(k)$
$i_{s,abc}(k)$
$i^{err}_{s,abc}(k)$
$i^{err\,OL}_{s,abc}(k+l)$
$[V_n, V_{ph,abc}](k)$

Fig. 4

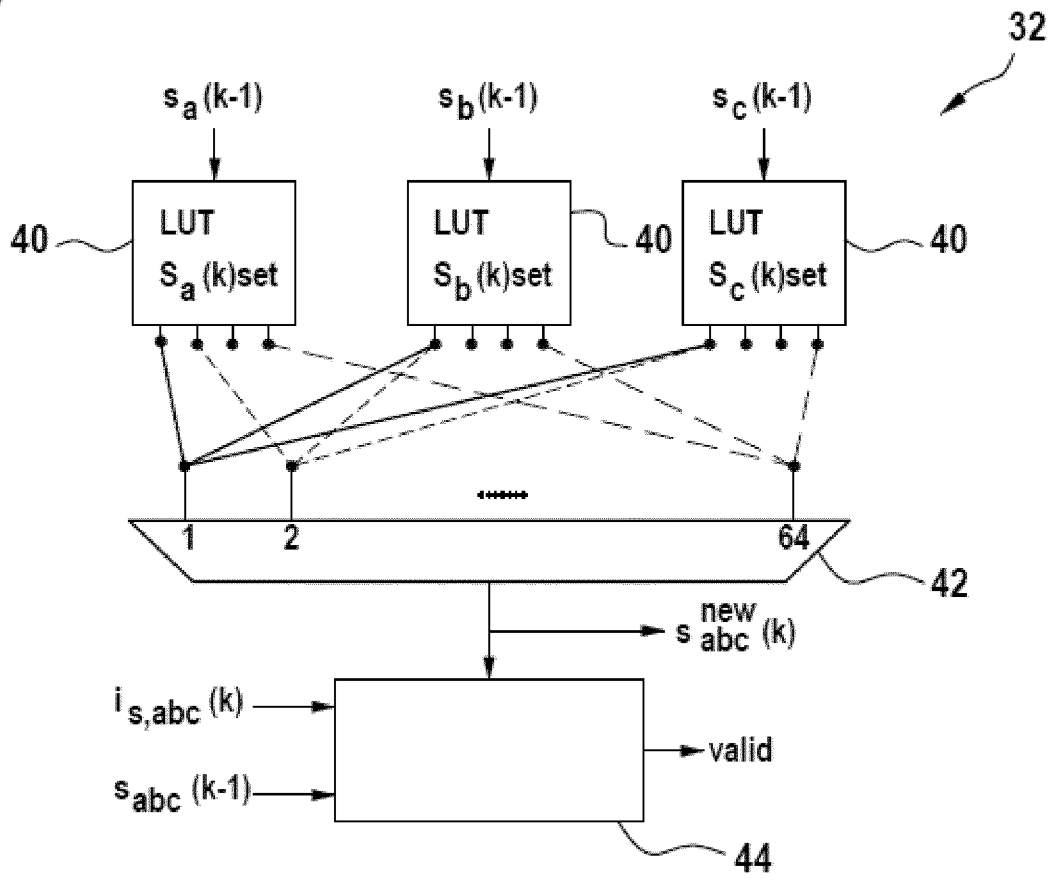

32
$s_a(k\text{-}1)$
$s_b(k\text{-}1)$
$s_c(k\text{-}1)$
LUT $S_a(k)$set
LUT $S_b(k)$set
LUT $S_c(k)$set
40
40
40
1
2
64
42
$s^{new}_{abc}(k)$
$i_{s,abc}(k)$
$s_{abc}(k\text{-}1)$
valid
44

Fig. 5

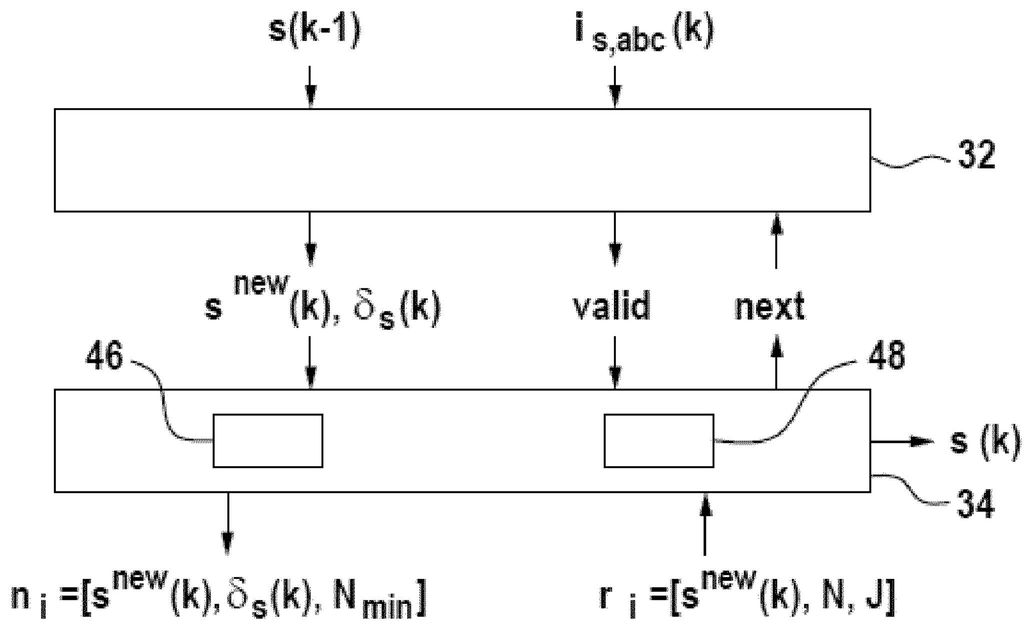

$s(k-1)$
$i_{s,abc}(k)$
32
$s^{new}(k), \delta_s(k)$
valid
next
46
48
$s(k)$
34
$n_i = [s^{new}(k), \delta_s(k), N_{min}]$
$r_i = [s^{new}(k), N, J]$

Fig. 6

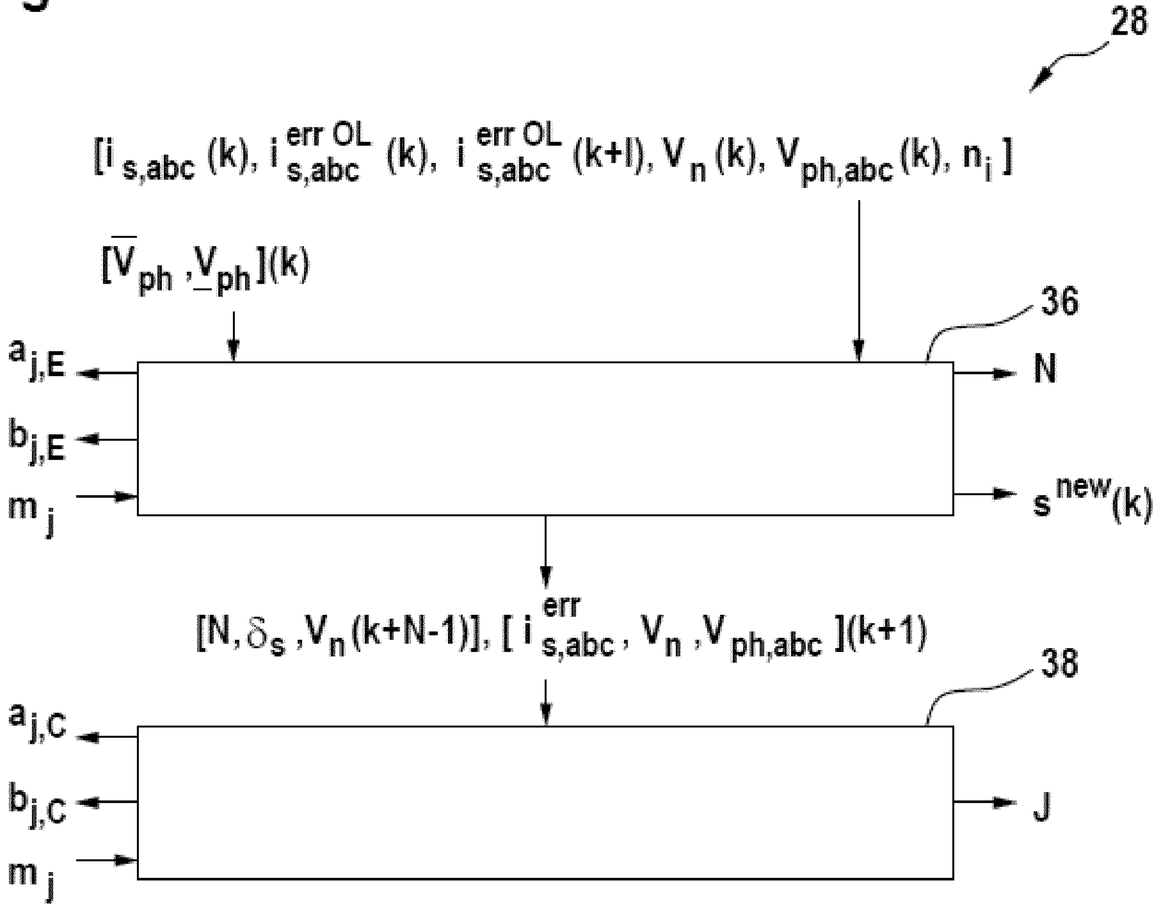
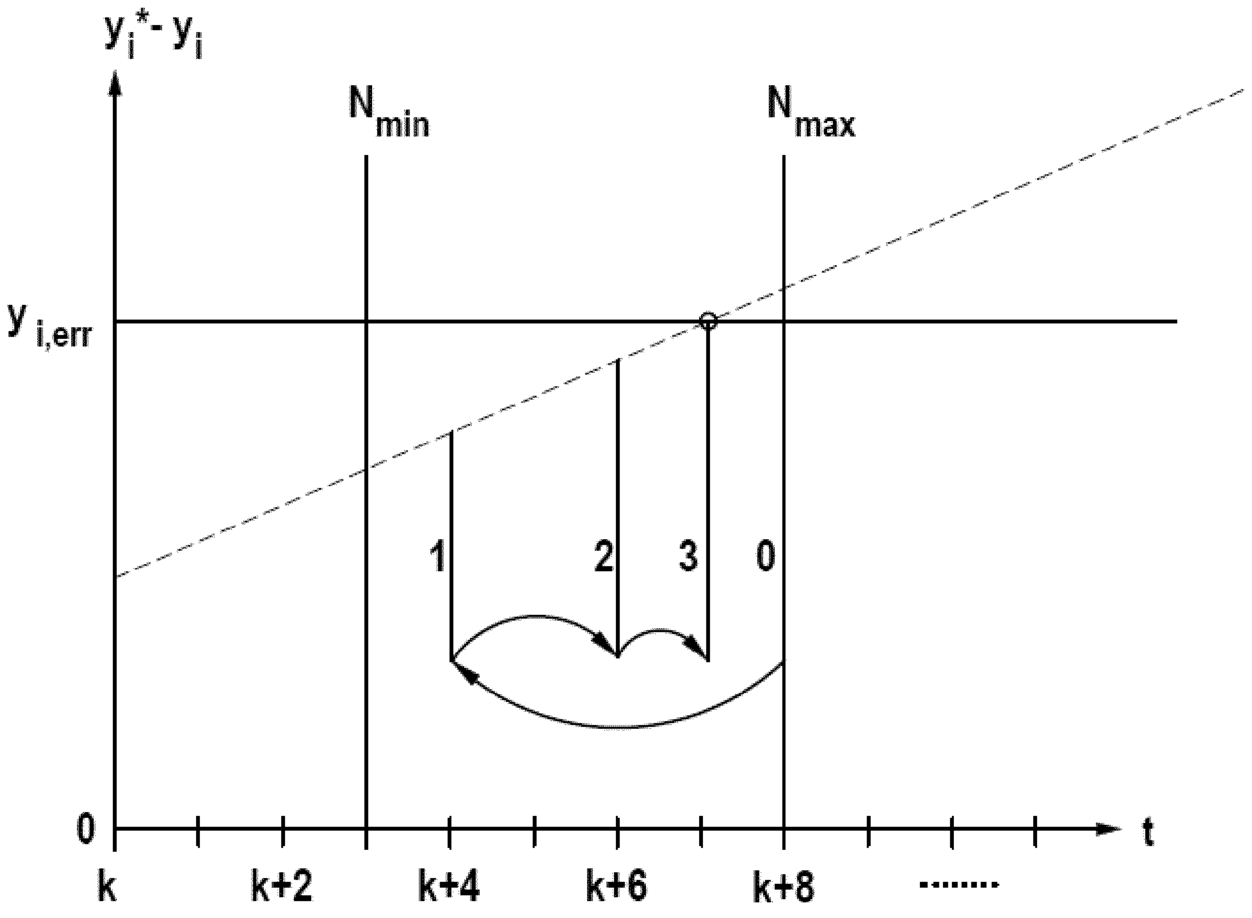

28
$[i_{s,abc}(k), i^{err\,OL}_{s,abc}(k), i^{err\,OL}_{s,abc}(k+l), V_n(k), V_{ph,abc}(k), n_i]$
$[\overline{V}_{ph}, \underline{V}_{ph}](k)$
36
$a_{j,E}$
$b_{j,E}$
$m_j$
N
$s^{new}(k)$
$[N, \delta_s, V_n(k+N-1)], [i^{err}_{s,abc}, V_n, V_{ph,abc}](k+1)$
38
$a_{j,C}$
$b_{j,C}$
$m_j$
J

Fig. 7



# INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2016/062807

**A. CLASSIFICATION OF SUBJECT MATTER**
INV. G05B13/04 H02M7/483 H02M7/487
ADD.

According to International Patent Classification (IPC) or to both national classification and IPC

**B. FIELDS SEARCHED**

Minimum documentation searched (classification system followed by classification symbols)
H02M G05B

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

EPO-Internal, WPI Data

**C. DOCUMENTS CONSIDERED TO BE RELEVANT**

| Category* | Citation of document, with indication, where appropriate, of the relevant passages | Relevant to claim No. |
|---|---|---|
| X | WO 2014/006200 A1 (ABB TECHNOLOGY AG [CH]) 9 January 2014 (2014-01-09) page 2, line 28 - page 2, line 32 page 3, line 15 - page 3, line 27 page 4, line 1 - page 4, line 12 page 5, line 24 - page 5, line 28 page 6, line 1 - page 6, line 16 page 7, line 24 - page 7, line 25 page 11, line 16 - page 15, line 13 ----- -/-- | 1-11 |

[X] Further documents are listed in the continuation of Box C.

[X] See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

| Date of the actual completion of the international search | Date of mailing of the international search report |
|---|---|
| 25 August 2016 | 02/09/2016 |

| Name and mailing address of the ISA/ | Authorized officer |
|---|---|
| European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016 | Madouroglou, E |

1

# INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2016/062807

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

| Category* | Citation of document, with indication, where appropriate, of the relevant passages | Relevant to claim No. |
|---|---|---|
| X | TOBIAS GEYER ET AL: "Model Predictive Direct Torque Control of a variable speed drive with a five-level inverter", IECON 2009 - 35TH ANNUAL CONFERENCE OF IEEE INDUSTRIAL ELECTRONICS (IECON 2009) - 3-5 NOV. 2009 - PORTO, PORTUGAL, IEEE, PISCATAWAY, NJ, USA, 3 November 2009 (2009-11-03), pages 1203-1208, XP031629326, ISBN: 978-1-4244-4648-3 paragraphs [000I], [00IV] ----- | 1-11 |
| A | WO 2009/080407 A1 (ABB RESEARCH LTD [CH]; PAPAFOTIOU GEORGIOS [CH]; ZURFLUH FRANZ [CH]) 2 July 2009 (2009-07-02) cited in the application page 4, line 3 - page 4, line 20 page 6, line 5 - page 6, line 28 page 7, line 10 - page 7, line 16 page 8, line 30 - page 9, line 15 ----- | 1-11 |
| A | WO 2009/016113 A1 (ABB RESEARCH LTD [CH]; PAPAFOTIOU GEORGIOS [CH]; HARNEFORS LENNART [SE) 5 February 2009 (2009-02-05) page 2, line 17 - page 2, line 25 page 3, line 13 - page 3, line 19; claim 1; figure 2 ----- | 1-11 |

1

# INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No

PCT/EP2016/062807

| Patent document cited in search report | Publication date | Patent family member(s) | Publication date |
|---|---|---|---|
| WO 2014006200 A1 | 09-01-2014 | CA 2877440 A1 | 09-01-2014 |
| | | CN 104396136 A | 04-03-2015 |
| | | EP 2870689 A1 | 13-05-2015 |
| | | KR 20150028282 A | 13-03-2015 |
| | | US 2015171726 A1 | 18-06-2015 |
| | | WO 2014006200 A1 | 09-01-2014 |
| WO 2009080407 A1 | 02-07-2009 | AT 522978 T | 15-09-2011 |
| | | CA 2705721 A1 | 02-07-2009 |
| | | CN 101904086 A | 01-12-2010 |
| | | EP 2223426 A1 | 01-09-2010 |
| | | ES 2371803 T3 | 10-01-2012 |
| | | JP 5474818 B2 | 16-04-2014 |
| | | JP 2011519253 A | 30-06-2011 |
| | | KR 20100092957 A | 23-08-2010 |
| | | PL 2223426 T3 | 31-01-2012 |
| | | RU 2010130263 A | 27-01-2012 |
| | | US 2010253269 A1 | 07-10-2010 |
| | | WO 2009080407 A1 | 02-07-2009 |
| WO 2009016113 A1 | 05-02-2009 | NONE | |