Real-time Simulation of 3 Parallel PWM Rectifiers

Michal Kopecký [1)], Jan Švanda [2)] and Martin Vlček [3)]

ŠKODA ELECTRIC, a.s./SW2 Prague Department, Plzeň, Czech Republic

[1)] e-mail: *michal.kopecky@skoda.cz*

[2)] e-mail: *jan.svanda@skoda.cz*

[3)] e-mail: *martin.vlcek1@skoda.cz*

Abstract — **This paper describes the development of a real-time model up to 3 parallel PWM rectifiers and its implementation on FPGA using LabVIEW development environment. The main benefit of this real-time model is the fact that there is no need for a real device or a test stand for debugging of traction drive control SW. The Hardware-in-the-Loop testing with similar RT model of an induction machine has already brought large financial and time savings. Moreover, destructive states or states difficult to evoke can be tested using such a real-time model.**

Keywords — PWM rectifier, real-time simulation, Hardware-in-the-Loop (HIL), LabVIEW FPGA, traction drives control SW.

I. Introduction

A. Simulated System

The simulated system (see Fig. 1) consists of three parallel branches with one-phase PWM rectifiers connected to the common DC link circuit. Usage of this AC/DC conversion system brings following main advantages: possibility of energy recovery, consumption of sinusoidal current and DC voltage stabilization.

Each one-phase PWM rectifier consists of 4 IGBT transistors which form a full bridge. The system is controlled by generation of PWM signals at these transistors [1]. The use of three parallel branches allows reduction of power transmitted by each branch. Moreover, the 3 parallel PWM rectifiers can be controlled with offset to each other. Such a control can damp desired frequency band in the input AC current spectrum in order to fulfil the EMC limits.

B. Model Adaptation on Less Parallel PWM Rectifiers

A very important thing to be mentioned is that the model of 3 parallel PWM rectifiers could be easily used to simulate a system with only two or one parallel branch. One of the branches is not considered if we set resistance R_{as} to a huge magnitude (e.g. 10^{10} Ω) and the transformation ratio p to zero (this causes zero secondary voltage u_{as}) and ensure that the transistors in the disconnected rectifier are switched off. For example, the configuration with two secondary windings was used in the electric multiple unit Škoda 7Ev-RegioPanter. The AFE control SW for this unit was the first application which was tested with the real-time model.

C. Connection System Simulation

The model provides also the simulation of the system connection in the right order as well as failure states (see TABLE I). For this purpose there are 4 switches introduced into the model: S_0 (Start Voltage), S_1 (Charging Contactor), S_2 (Line Contactor) and S_3 (Load Contactor). It should be mentioned that the switches in parallel branches are coupled, i.e. they can be either on or off in all parallel branches.

TABLE I.
SWITCHING LOGIC FOR SIMULATION OF SYSTEM CONTROL

Switches States				System State
S_0	S_1	S_2	S_3	
0	0	0	0	Nothing is simulated.
1	0	0	0	Primary voltage connected, all secondary windings open.
1	1	0	0	Connected over charging resistance, Capacitor started charging.
1	1	1	0	Charging resistance bridged.
1	1	1	1	Current load connected. Operational state.
1	*0*	*1*	*0*	*Charging over small resistance. Huge currents. Failure state*
1	*1*	*0*	*1*	*Charging with connected load. Overcurrents. Failure state.*

From the point of view of simulation, all system states with S_1 switched on lead to the same model only with different parameters (of resistance or current load). For this reason, we will discuss below only how the operational state with all switches on can be simulated.

II. State Space Model

A. Transformer

The first issue to be considered is the separation of the primary circuit from the rest of the system.

The model of the secondary side requires secondary voltages as input. These could be computed from the primary input voltage easily using the transformation ratios:

$$\begin{aligned} u_{as1} &= p_{11} u_{ap} \\ u_{as2} &= p_{12} u_{ap} \\ u_{as3} &= p_{13} u_{ap} \end{aligned} . \qquad (1)$$

On the other hand, the information about primary current i_{ap} has to be provided to the system controller. Once all secondary currents are computed in the state

space model, the primary current can be obtained by formula:

$$i_{ap} = p_{11} i_{as1} + p_{12} i_{as2} + p_{13} i_{as3}\,. \tag{2}$$

Equations (1) and (2) allow elimination of primary circuit from simulation. Therefore, the only secondary side can be considered below.

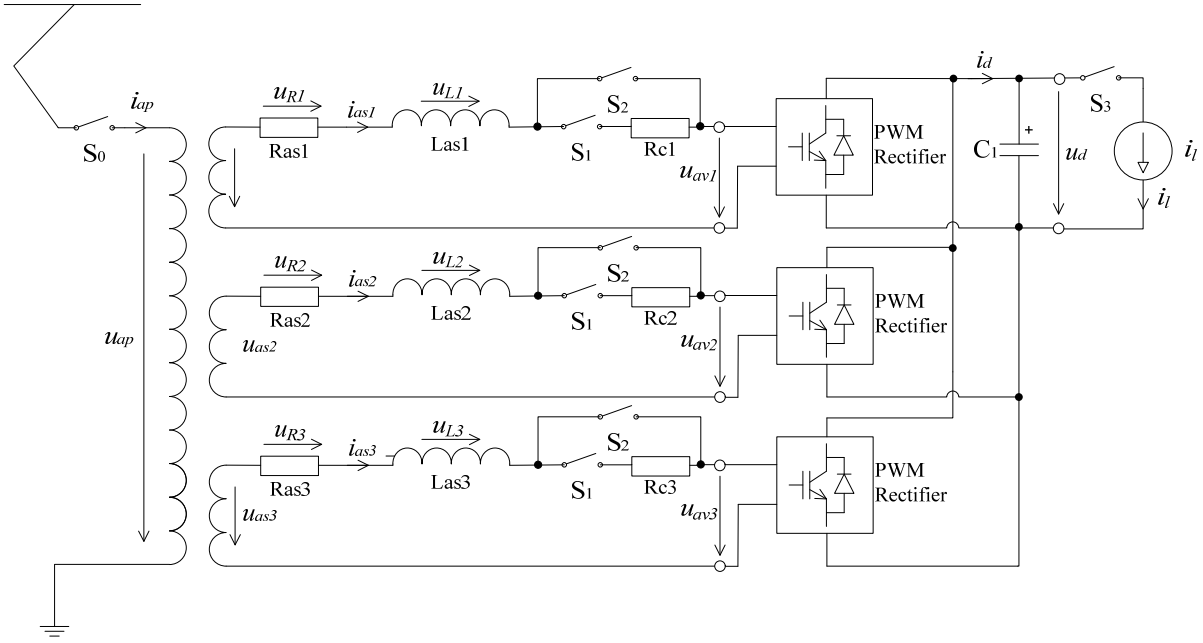

Fig. 1. Block diagram of the simulated system

B. State Space Representation

The mathematical model of the simulated system relies on the state space representation:

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned}\,. \tag{3}$$

Because the 3 parallel PWM rectifiers are connected to the same DC link, it is impossible to solve equations for each branch independently, but it is necessary to solve system describing the whole circuit with matrix *A* of dimension 4x4.

There are four state variables – inductor currents and capacitor DC voltage, forming the state vector:

$$x = \begin{pmatrix} i_{as1} \\ i_{as2} \\ i_{as3} \\ u_d \end{pmatrix}. \tag{4}$$

The input vector (5) contains 3 secondary voltages, DC load current and also diode and transistor forward voltages. These values are constant but the dependence of state and output variables on them varies in different system states.

$$u = \begin{pmatrix} u_{as1} \\ u_{as2} \\ u_{as3} \\ i_l \\ U_D \\ U_T \end{pmatrix}. \tag{5}$$

The output vector consists of all other quantities in the circuit, i.e.:

$$y = \begin{pmatrix} u_{av1} \\ u_{R1} \\ u_{L1} \\ u_{av2} \\ u_{R2} \\ u_{L2} \\ u_{av3} \\ u_{R3} \\ u_{L3} \\ i_d \end{pmatrix}. \tag{6}$$

Unlike the most common problems, the state matrices are not constant in time in our case. But they change according to the individual states of each parallel circuit. These states are determined by the semiconductor elements which current flows through.

Because solving of the state space system is relatively easy and well known problem, the crucial point of the model development is to invent the switching algorithm which selects the appropriate state matrices in the current simulation step.

C. State Matrices Construction

Before analysing the possible circuit states, the part-by-part matrices construction (see TABLE II) is introduced. This construction, which substantially simplifies the whole switching algorithm, is based on the independence of individual parallel branches on each other. This is due to the fact that the state of each parallel branch depends only on the quantities in this branch and DC link voltage but this state is independent of the other branch quantities.

TABLE II.
OVERALL STATE MATRICES COMPOSING OF ROWS CORRESPONDING TO INDIVIDUAL PWM RECTIFIERS

Branch	Matrices, Rows	Corresponding variables
1st PWM Rectifier branch	1st row of matrix *A*	i_{as1}
	1st row of matrix *B*	
	1st to 3rd row of matrix *C*	u_{av1}, u_{R1}, u_{L1}
	1st to 3rd row of matrix *D*	
2nd PWM Rectifier branch	2nd row of matrix *A*	i_{as2}
	2nd row of matrix *B*	
	4th to 6th row of matrix *C*	u_{av2}, u_{R2}, u_{L2}
	4th to 6th row of matrix *D*	
3st PWM Rectifier branch	3rd row of matrix *A*	i_{as3}
	3rd row of matrix *B*	
	7th to 9th row of matrix *C*	u_{av3}, u_{R3}, u_{L3}
	7th to 9th row of matrix *D*	

Further, the i_d current is computed as a linear combination of i_{as} currents according to which parallel branches are connected to the DC link and with orientation of which in the current time step:

$$i_d = k_1 i_{as1} + k_2 i_{as2} + k_3 i_{as3}, \tag{7}$$

where coefficients k_1, k_2, k_3 are assumed to be *0* or *±1*. The last rows of the state matrices can be constructed using these coefficients:

$$\begin{aligned} A_{4,\cdot} &= \left(\frac{k_1}{C_1} \quad \frac{k_2}{C_1} \quad \frac{k_3}{C_1} \quad 0 \right) \\ B_{4,\cdot} &= \left(0 \quad 0 \quad 0 \quad -\frac{1}{C_1} \quad 0 \quad 0 \right) \\ C_{10,\cdot} &= \left(k_1 \quad k_2 \quad k_3 \quad 0 \right) \\ D_{10,\cdot} &= \left(0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \right) \end{aligned} \tag{8}$$

III. STATE SWITCHING ALGORITHM

Because of the independence of each parallel branch state on other branches we can consider only the state switching for one PWM rectifier. Once the next states in all parallel branches are determined we can construct the state index vector of 3 elements and according to the TABLE II also the state matrices for the next time step computation.

A. Possible States of One PWM Rectifier

We will follow the algorithm development history describing also the problems we came across during it.

Theoretically, there are $2^4 = 16$ possible states needed to be involved for the full description of one PWM rectifier considering all possible combinations of semiconductor elements conduction. In fact, the number of all possible states can be reduced to 7 by excluding prohibited states (short-circuited branch – causes error of application) and by merging some switching combination into one (in terms of external behavior it does not matter e.g. the sequence of transistor and diode conducting).

B. Basic State Switching Model

This simplest model was used only for simulation of the uncontrolled single phase rectifier (i.e. diode bridge). In this case there are only 3 system states:

Open circuit (No. 1),

Positive branch conduction (No. 2) and

Negative branch conduction (No. 3).

However as we shall see later, it is not sufficient in a more complex situation.

The main idea of this model is very simple. Before each step of the state space model a sequence of conditions is evaluated and the first fulfilled condition determines an appropriate state.

It is important to mention that here as well as in the whole paper the algorithm depends on the sequence of condition evaluation. More than one condition can lead to the given state, e.g. the state No. 3 (see Fig. 3) can be chosen if one of the following conditions is fulfilled.

```
if (ias <= -Ias_Min & direction < 0)
%Current is flowing in negative
direction and was flowing in this
direction also in the previous step
elseif (-uas+uR+uL-ud) >= 2*UD
%Zero current in circuit and there is
sufficient voltage to diode D3 and D2
begin conducting.
```

C. Correction of Negative Current i_d

The calculated AC current i_{as} can change its sign between two consecutive simulation steps. As a consequence, also the DC current i_d changes its sign from plus to minus. However, this is not physically possible.

The global variable `direction` used by decision on the next state, avoids simulation to go to a wrong state on the basis of that established configuration only. However, already computed solution at that simulation step would stay wrong (blue curve in Fig. 2).

A very simple additional condition was introduced into the model, which returns i_d up to 0, whenever $i_d < 0$ occurs (green curve in Fig. 2).

D. Infinitesimal Occurrence of Open State

We observed that the system cannot immediately change its state by the current i_{as} zero crossing – the state No. 1 (open circuit) always has to occur at least for one time step.

This treatment works fine for an uncontrolled rectifier (with nonzero diode threshold voltage U_D). However, it brings problems in the simulation of PWM controlled rectifiers

. We will demonstrate them on our model situation. Let us consider situation with transistor T2 constantly closed (e.g. pwm = [0 1 0 0]) and system being in the state No. 5, when T2 and D4 conduct (see Fig. 4).

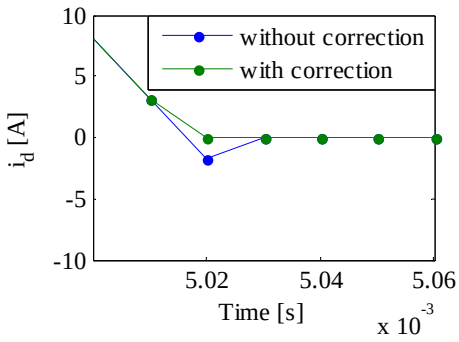

Fig. 2: Correction of the current i_d

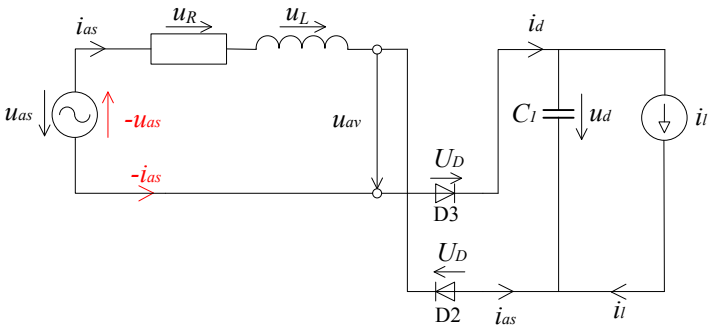

Fig. 3: State No. 3 diagram (diode D3 and D2 conducting)

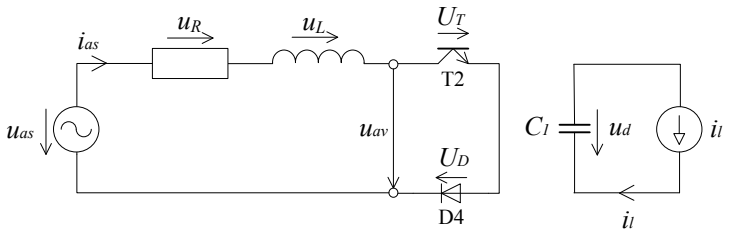

Fig. 4: State No. 5 diagram (transistor T2 and diode D4 conducting)

The value of short-circuit current i_{as} is continuously decreasing until it changes its sign in one particular simulation step. In this case the system should go to the already known state No. 3 which would also correspond to the physical reality. Unfortunately, the global variable `direction=1` does not allow smooth state transition and the intermediate state No. 1 (open circuit) is set for one simulation step. As a consequence, the step change of voltages u_{av} and u_L occurs in this step (see Fig. 5).

Let us remark that the solution does not bring omission of the global variable `direction` from the algorithm switching. In this case, smooth transition from the state No. 5 to the state No. 3 occurs without any problems. However, the very next state transition fails. In that case the system in the state No. 3 should not move to the state No. 5 but the state No. 1 should be established. Unfortunately, the switching algorithm does not catch this, because of missing condition with the global variable `direction`. As a consequence, the wrong state No. 5 is established and this wrong decision results into oscillation of the numerical solution (see Fig. 6).

From the description above we can conclude that the key issue of the whole switching algorithm is to determine if the i_{as} current zero crossing is or is not physically realistic. This reality is done by voltages relationships in the simulated circuit.

The final solution of the problem described above has brought infinitesimal occurrence of open state philosophy which relies on the following idea: if the current i_{as} changes its sign between two consequential steps, we can suppose that the open circuit state (No. 1) always occurs for infinitesimally short moment (shorter than the simulation step!). The consequence of this occurrence are settings $i_{as} = 0$, $u_R = 0$ and $u_L = 0$. While a reason for the zero resistance voltage is obvious, by inductance voltage u_L we can imagine that the through-going current is for very short time, but still constantly zero and therefore also its derivative is zero. After this infinitesimal occurrence of open circuit state the voltages conditions for transition to the states with different i_{as} current sign can be evaluated and if some of them is fulfilled, the system could go to the appropriate state. It is important to remark that the values of quantities by infinitesimal open circuit state occurrence do not appear in numerical solution, in contrast to Fig. 5.

By the way, this approach has introduced the current state knowledge into our state switching algorithm, which will be necessary in all state switching managers listed below.

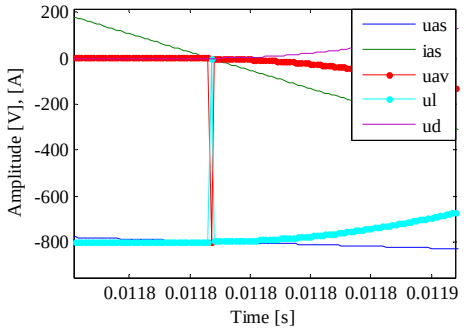

Fig. 5: Vault of voltages u_{av} and u_L by setting of the intermediate state No. 1

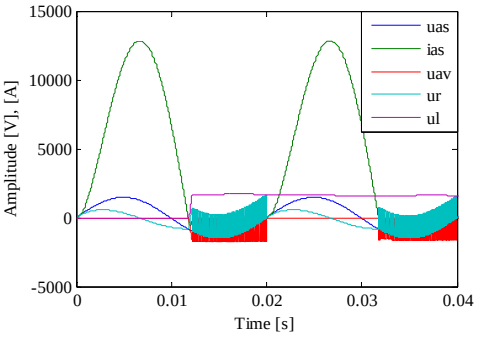

Fig. 6: Oscillation of u_{av} and u_L caused by omitting the global variable `direction` condition

E. Model with Groups According the i_{as} Current Direction

According considerations above we introduced the concept of state partition into two groups according to the sign of the current i_{as}. The default state No. 1 with zero i_{as} current stands out of both groups. Whereas a transition between two states in the same group proceeds immediately, a transition between two states from different groups is possible only with infinitesimal occurrence of the open circuit state.

For practical implementation of such a switching algorithm, we have left the separate condition principle. Now, the state conditions are evaluated together in two functions:

```
function state_index =
Group1_conditions(ias, uas, uR, uL,
ud, pwm),
function state_index =
Group2_conditions(ias, uas, uR, uL,
ud, pwm).
```

These functions start with the state No. 1 (i.e. `state_index = 1`) and process conditions for individual states bottom-up according to the state index. Finally, the function returns the state with the highest index in the appropriate group where the conditions are met. If the function returns the default value `1` it means that no conditions for any state in the group are fulfilled.

The next state alone is chosen in

```
function new_index =
choose_state_using_Groups(index, ias,
uas, uR_o, uL_o, ud, pwm).
```

First, there are evaluated conditions for the state in the same group in the function `Group`*`i`*`_conditions` (appropriate function is determined by group index *`i`* according to the state in the previous time step). As inputs to this function, the just computed state variables values, the current value of the voltage u_{as} and the values of the voltages u_R and u_L from the previous time step are sent. If none of these conditions is met (i.e. function `Group`*`i`*`_conditions` returns one), the second function belonging to the other group is called as well – but in the inputs there are the values of i_{as}, u_R a u_L replaced by zeros (because of the infinitesimal open state circuit occurrence).

In case the previous state index is equal to 1, there are called both function with the classic inputs and for the next state index maximum of both returned values is chosen.

We attach the MATLAB source code of the switching manager described above:

```
switch index   %Decision-making
according the previous state index
  case {2,5,7}   %Group1 (ias > 0)
     new_index = Group1_conditions(ias,
     uas, uR_o, uL_o, ud, pwm);
     if new_index == 1   %No state
     conditions in group 1 fulfilled
        new_index =
        Group2_conditions(0, uas, 0, 0,
        ud, pwm);
     end
  case {3,4,6}   %Group2 (ias < 0)
     new_index = Group2_conditions(ias,
     uas, uR_o, uL_o, ud, pwm);
     if new_index == 1   %No state
     conditions in group 2 fulfilled
        new_index =
        Group1_conditions(0, uas, 0, 0,
        ud, pwm);
     end
  case 1   %Open circuit
     new_index =
     max(Group1_conditions(ias, uas,
     uR_o, uL_o, ud, pwm),
     Group2_conditions(ias, uas, uR_o,
     uL_o, ud, pwm));
end
```

The advantage of this switching manager is not only the solution of both problems described above but also the relatively easy portability on FPGA because the switching algorithm takes the same amount of time in every step. Moreover, the evaluation of both functions `Group1_conditions` and `Group2_conditions` can execute in parallel.

F. Correction in Case of Open State Circuit Choice in Model with 3 Parallel PWM Rectifiers

If the next state chosen in the function `choose_state_using_Groups` is the open circuit state No. 1 in certain parallel branch, non-zero i_{as} current flowing through this branch is physically impossible.

For this reason, it has to be corrected to zero. As a consequence, we should recalculate also the voltage u_d. This is done using the reverse approach to the Euler method for the state space numerical solution, i.e. from the already computed new value of u_d we subtract the contribution of the appropriate i_{as} current (the one that should be zero). The use of the coefficient vector k defined by (7) brings advantages to do that.

Be aware that values of coefficients k_1, k_2, k_3 do not correspond to the groups according to the sign of i_{as} current. However, their values depend on addition of individual currents charging capacitor in the DC link (see TABLE III).

TABLE III.
VALUES OF COEFFICIENTS ACCORDING TO THE STATE INDEX

Coefficient k_i value ($i = 1, 2, 3$)	State index of the parallel branch i	Semiconductor element conducting
1	2	D1&D4
	6	T4&T1
-1	3	D3&D2
	7	T2&T3
0	1	None
	4	D3&T1
	5	T2&D4

The final MATLAB implementation of the switch manager for 3 parallel PWM rectifiers combines both choosing of the next state and correction in the case of open state circuit in one execution of the for loop. The source code is provided below.

```
k_o = k; %Storing "old" vector value
used in state space numerical solution
for i=1:PPR %PPR = 3 (#PWM Rectifiers)
   %Choosing of a new state index of
   the i-th PWM Rectifier
   index(i) =
   choose_state_using_Groups(index(i),
   ias(n,i),uas(n,i), ur(n-1,i), ul(n-
   1,i), ud(n), pwm(n,i,:));
   %Correction in case of state No. 1 –
   open circuit.
   if index(i) ==1
      ias(n,i) = 0;
      ud(n) = ud(n)-ko(i)/C1*h*ias(n-
      1,i);
   end;
end;
```

G. Final Implementation of 3 Parallel PWM Rectifiers Model

We recapitulate the whole model of 3 parallel PWM rectifiers implementation below.

In each simulation steps:

1. Numerical solution of the state equation $\dot{x} = Ax + Bu$ using Euler method.
2. Choosing new state using the state switching manager and state matrices updating according to the new state
3. Evaluating the outputs from the equation $y = Cx + Du$

The fact that the outputs are computed with already new state matrices brings not only higher accuracy but also no need of additional correction of the current i_d (see Fig. 2). Because i_d is computed from already corrected currents i_{as} it is ensured that it will be non-negative.

IV. THE REAL-TIME MODEL

A. SW and HW resources

In order to run the model described above in real-time we need to move on different platforms than PC: real-time system and especially field-programmable gate array (FPGA).

TABLE IV.
SW AND HW RESOURCES FROM NATIONAL INSTRUMENTS

Resource	Usage
NI PXI-7854R Multifunction RIO with Virtex-5 LX110 FPGA	Numerical model, Inputs and outputs operation
NI PXIe-8133 1.73 GHz Quad-Core Real-Time Controller	Communication with FPGA model and with PC Interface
NI PXI-4110 Triple-Output Programmable DC Power Supply	HW interfaces power supply
NI PXIe-1065 18-Slot 3U PXI Express Chassis	Chassis, Backplane communication
LabVIEW 2012 SP1 f3	SW used by model implementation
LabVIEW Real-Time 12.0.0	
LabVIEW FPGA 12.0.0	

We make demand of the highest possible computational repeat on the FPGA model. This should be at least 20 times higher than switching frequency (900 Hz in most applications). However, there is only a limited number of FPGA resources. We will describe our approach how to fulfil these contradictory requirements by reaching the highest model accuracy as possible later.

B. Numerical Format

For implementation of the computational VI on FPGA in floating point the IP Xilinx CORE Generator Blocks (see Fig. 7) were used. The advantages of this approach are easy portability of the model for systems with (totally) different parameters and high model precision. On the other hand this approach increases significantly the FPGA sources usage.

Not many operators in double precision can be placed in our code to avoid FPGA resources overuse. In our case, there were only two multipliers and two adders used. The crucial part of our work is then to invent how to order the input data and the intermediate results as the inputs of individual operators to get the results in the shortest time.

This can be done using FPGA high-throughput 2-wire protocol (see TABLE V).

TABLE V.
LOGICAL SIGNALS OF 2-WIRE HANDSHAKING PROTOCOL

LabVIEW standard name	Xilinx name [2]	Description
input valid	operation_nd	Determinates if there is new valid input data to process.
output valid	rdy	Determinates if the received result of operation is valid.

C. FPGA Model Performace Considerations

Because of the requirements to reduce the model space and timing, we introduced some more or less substantial simplifications.

First, we rewrite Euler method (9).

$$x_{n+1} = x_n + h(Ax_n + Bu_n). \tag{9}$$

We can save 4 multiplications for evaluating x_{n+1}, when we use the multiplied matrices *hA* and *hB* instead of the matrices *A* and *B* on FPGA The equation (9) will be transformed into shape:

$$x_{n+1} = x_n + hAx_n + hBu_n. \tag{10}$$

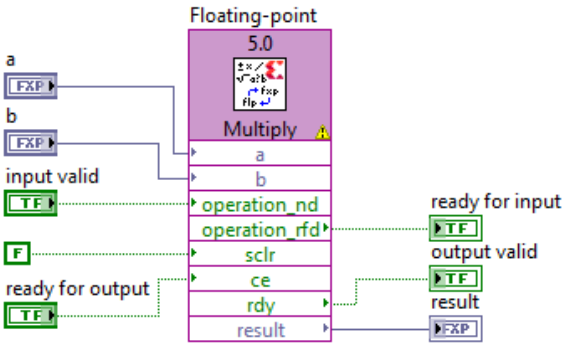

Fig. 7: Using of Xilinx Floating-Point Operator to multiply 2 numbers in double precision on FPGA

Substantially significant resources and time savings can be reached using the sparse matrices concept. In general, all matrices *hA*, *hB*, *C* and *D* have together 16 + 24 + 40 + 60 = 140 elements. However, there are only 52 elements which can be non-zero in some combination of individual states (see TABLE VI). Only these non-zero elements are involved in the computation. We only have to keep in mind the sparse matrices structure (see TABLE VI) by model implementation.

TABLE VI.
COORDINATE-WISE REPRESENTATION OF SPARSE MATRICES

Matrix	Row index of non-zero elements / Column indexes of non-zero elements																	#
hA	1	1	2	2	3	3	4	4	4									9
	1	4	2	4	3	4	1	2	3									
hB	1	1	1	2	2	2	3	3	3	4								10
	1	5	6	2	5	6	3	5	6	4								
C	1	2	3	3	4	5	6	6	7	8	9	9	10	10	10			15
D	4	1	1	4	4	2	2	4	4	3	3	4	1	2	3			18
	1	1	1	3	3	3	4	4	4	6	6	6	7	7	7	9	9	9	
	1	5	6	1	5	6	2	5	6	2	5	6	3	5	6	3	5	6	

The numerical solver virtual instrument (VI) solving one simulation step contains 3 parallel computational threads, switch state manager VI and building matrices VI placed in single cycle timed loop (SCTL).

The main FPGA VI consists of the solver loop, digital inputs (PWM) processing loop and ten times slower analog output generation loop and loop which sends data to the RT controller (from where they are forwarded to PC). The compiled code of this main VI consumes almost all FPGA resources (see TABLE VII).

TABLE VII.
DEVICE UTILIZATION OF COMPILED MAIN FPGA VI

Total Slices	91.9 % (15880 out of 17280)
Slice Registers	68.2 % (47166 out of 69120)
Slice LUTs	66.5 % (45937 out of 69120)
DSP48s	71.9 % (46 out of 64)
Block RAMs	7.8 % (10 out of 128)

Sparse matrices usage, involving multiplication with simulation step *h* and parallel executions decrease the number of 40 MHz SCTL executions down to 182. By code benchmarking we discovered that due to the overhead whole loop in main FPGA VI is running with tact of 208 ticks of the 40 MHz clock. To ensure that the simulation will be actually running in real-time we decided to choose simulation step $h = 6.25\ \mu s$, corresponding to 250 ticks of the 40 MHz clock.

D. Communication

The model receives the PWM pulses from the electronic control unit (ECU) via optical-electrical converter and digital inputs of FPGA.

The output analog data are adjusted and generated in slower output generation loop and provided using SW and HW interface to ECU (see Fig. 8).

Besides, the FPGA model communicates with an application on PXIe-8133 RT controller. This communication is provided by two DMA FIFOs. Host-To-Target FIFO sends input quantities, i.e. secondary voltages u_{as} and current consumption i_l. All computed quantities are sent back using Target-To-Host FIFO to RT target application. However, this application only sends data forward via Ethernet to PC. In PC, the received data are presented and also could be logged.

V. MODEL RESULTS

A. Model validation

We made a simple test to validate the model of one PWM rectifier. The input data for this testing was sinusoidal secondary voltage u_{as} (of the frequency 50 Hz and amplitude 250 V) and approximately constant load current i_l of 6.65 A. These quantities were measured by an experiment with the laboratory pulse rectifier.

The results of the RT model of one pulse rectifier with input data described above are shown in Fig. 10 and Fig. 12. This simulation results were compared with the real measured data to prove that the model behavior corresponds to the behavior of a real device.

B. Rectifiers offset testing

The data sent from model into PC are sampled with 16 kHz. Therefore, we can make the spectral analysis theoretically up to 8 kHz and use the model for testing electromagnetic compatibility of device by different configurations.

Primary current spectra are shown in Fig. 11 and Fig. 13. We can see that the cluster of spectral lines nearby frequency 1600 Hz can be eliminated by the offset control of 2 parallel PWM rectifiers.

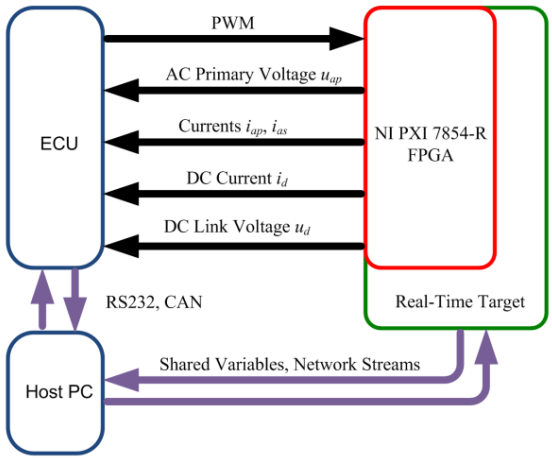

Fig. 8: Communication diagram of real-time model

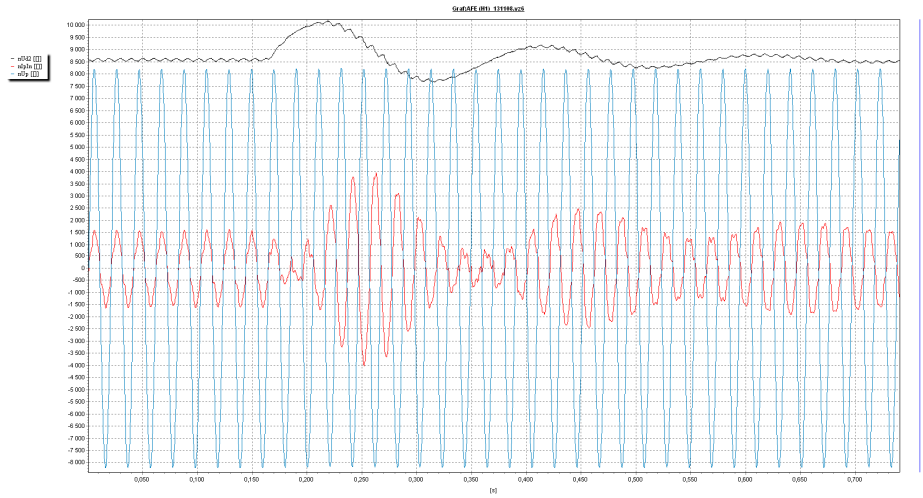

Fig. 9: Off-line graph of transition into energy recovery state from the diagnostic environment DISMON. Primary voltage u_{ap} in blue, primary current i_{as} in red and DC link voltage u_d in black (all in computer units).

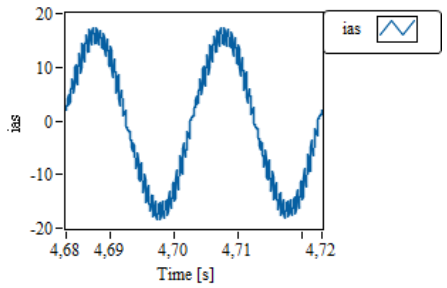

Fig. 10: Secondary current i_{as}

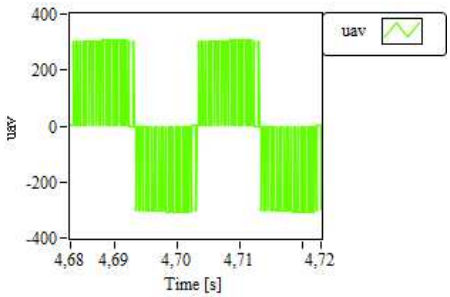

Fig. 12: Bridge input voltage u_{av}

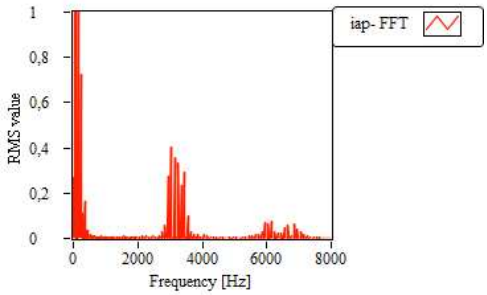

Fig. 11: Spectrum of primary current i_{ap} by two PWM rectifiers running

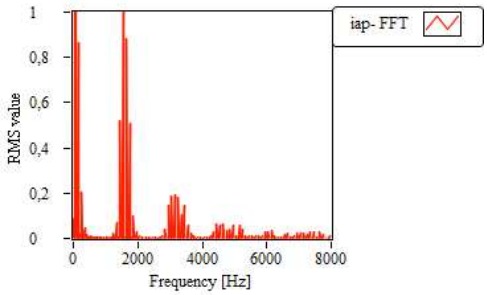

Fig. 13: Spectrum of primary current i_{ap} by only one PWM rectifier running

VI. Conclusion

This paper has presented the main issues we encountered by the development of the real-time model of 3 parallel PWM rectifiers. This model allows us to perform a substantial part of the control SW testing without existence of a real device [3]. Moreover, such tests with model can be easily rerun.

Therefore, it is planned to use this model to test the control SW in all projects in our company in the future.

Acknowledgment

The whole research was made under the research project Extension of Traction Drive Real-time Model (No. 65Z6210) in ŠKODA ELECTRIC, a.s. The publications were financially supported by the Technology Agency of Czech Republic (TACR) under the grant Competence Center of Railway Vehicles (No. TE01020038).

References

[1] J. Bauer, "Single-Phase Pulse Width Modulated Rectifier," *Acta Polytechnica*, vol. 48, no. 3/2008, Czech Technical University in Prague, pp. 84-87.

[2] *LogiCORE IP Floating-Point Operator v5.0*. XILINX, 2011.

[3] M. Kopecký, J. Švanda, and M. Vlček, "Využití real-time simulací při návrhu řízení trakčních pohonů," *XXXIII. konference o elektrických pohonech*, Pilsen, pp. 84-89, June 2013.

[4] M. Kopecký, V. Buba, "Příspěvek k řízení pulzního usměrňovače lokomotivy 109E," *XXX. konference o elektrických pohonech*, Pilsen,, June 2007.

[5] M. Kopecký, M. Bednář, "Dosavadní zkušenosti s algoritmy řízení 4Q na lokomotivě 109E a jednotce 5Ev Litva," *XXXI. konference o elektrických pohonech*, Pilsen,, June 2007.

[6] J. Javůrek, *Regulace moderních elektrických pohonů*. Grada, 2003.

[7] E. Vitásek, *Základy teorie numerických metod pro řešení diferenciálních rovnic*. Academica, 1994.

[8] J. Švanda, *Using NI PXI Express and CompactRIO to Develop a Hardware-in-the-Loop Tester for Electric Driver ECUs of Locomotive*, National Instrument, Solutions, Case Studies. [Online]. Available: http://sine.ni.com/cs/app/doc/p/id/cs-15423 [Accessed: 8 April. 2014].

[9] *High-Throughput LabVIEW FPGA Exercises*. National Instruments, 2012.