Impact of the Hardened Floating-Point Cores on HIL Technology

Alberto Sanchez[a,*], Elías Todorovich[b,c], Angel de Castro[a]

[a]*HCTLab Research Group, Universidad Autonoma de Madrid*
[b]*Universidad Nacional del Centro de la Provincia de Buenos Aires, Argentina*
[c]*Faculty of Engineering, FASTA University, Mar del Plata, Argentina*

Abstract

The Hardware-In-the-Loop (HIL) technique is increasingly used for testing power electronics. FPGAs (Field-Programmable Gate Array) are becoming usual in this kind of emulation due to their acceleration capabilities. But even using FPGAs, it has not been possible to reach real time simulations when small integration steps are necessary (around 100 ns or lower) if floating-point representation is used. Fixed-point has been the solution, but at a high design effort cost. With the release of FPGAs with HFP (Hardened Floating-Point) cores — dedicated floating-point blocks implemented in silicon —, the minimum achievable simulation step decreases significantly. This paper presents a comparison between HFP cores, floating-point in programmable logic and fixed-point for HIL models. Results show that both HFP-based and fixed-point arithmetic achieve a simulation step around 10 ns for a full-bridge converter model. A comparison regarding resolution and accuracy is also presented, because acceleration is not the only issue when decreasing the integration step. Numerical resolution also plays an important role, and 32-bit floating-point representation finds a double barrier: acceleration marked by technology, and numerical resolution. Both are explored in this paper.

Keywords: Hardware-in-the-loop, floating-point, fixed-point, real-time emulation, programmable logic

*Corresponding author
Email address: `alberto.sanchezgonzalez@uam.es` (Alberto Sanchez)

1. Introduction

Over the past two decades, there has been a significant growth in digital control of switched-mode power supplies [1, 2, 3, 4, 5], motors and other electronic devices [6, 7, 8, 9]. This has created the need of debugging mixed analog-digital systems, complicating simulations and making them longer. The reason for long simulations is usually the difference between natural times in digital and analog parts. At the same time, digital devices have grown in performance and speed, making it possible to emulate analog systems in real time, known as HIL (Hardware-In-the-Loop). Apart from the significant acceleration of the debugging process using HIL instead of mixed-signal simulations, HIL also offers the opportunity to debug the controller in its final implementation. Therefore, it is not surprising the growth of HIL market. As an example, [10] offers an extensive review of the simulation alternatives for microgrids and it evinces the consolidated use of HIL in power electronics.

It is important to keep in mind that HIL systems must run at real time. That is why the use of FPGAs has caused a revolution of HIL systems. While microprocessor-based HIL implementations achieve integration steps of about hundreds or tens of μs [11], FPGAs can reach integration steps of tens or hundreds of ns [12, 13, 14, 15, 16, 17]. Therefore, commercial tools like Typhoon HIL, OPAL-RT, dSPACE and Hypersim use FPGAs to accelerate their models.

When using FPGAs, commercial HIL systems can handle complex models, defined by the user, with an integration step of tens of μs, almost without requiring user optimization. For instance, in [18], a supervisory predictive control of a power plant and a model of a gas microturbine are implemented with the RT-LAB simulator from OPAL-RT. In [19], the Hypersim real time simulator is used to implement a model of the VSC-HVDC (Voltage-Source Converter High Voltage Direct Current) link between France and Spain achieving a 20 μs time step. In [20], OPAL-RT eMEGASIM and OMNeT++ are used to model clusters of microgrids and their communication network in order to analyze the latency of a power routing algorithm.

However, in the case of systems that require an integration step in the order of tens or hundreds of ns, like high frequency switching converters, these tools cannot improve the integration step as much as desired, so it is necessary to implement an optimized model in an FPGA by hand [21, 22, 23, 24, 25]. In those cases in which the speed of the HIL model is crucial, the way of representing the internal variables becomes critical. Two main arithmetics can be used: fixed-point and floating-point. The former allows high synthesis frequencies but the design effort is not negligible, while the latter is much easier to use but it allows notably lower frequencies. In [26] a comparison between fixed and floating-point arithmetics was accomplished, showing that the fixed-point model worked about ten times faster while using less than ten times fewer resources in terms of LUTs (Look-Up Table). However, regarding the implementation effort, it is true that optimized fixed-point models may not be viable if the model is not simple.

The main contribution of this paper comes from the comparison of fixed-point and floating-point for HIL systems, but taking into account the irruption of new floating-point possibilities in FPGAs. Recently, FPGAs with HFP (Hardened Floating Point) cores, i.e. floating-point cores implemented in silicon without using programmable logic, have begun to be commercialized. Although the term *hardened* has being mainly used to describe processors implemented in silicon inside the FPGA, it is also used for any other cores implemented in silicon like the new floating-point cores [27, 28]. Using HFP cores, the Intel Arria 10 FPGA family offers 1,500 GFLOPS of DSP (Digital Signal Processor) performance, which is the total performance of all cores running simultaneously and in the best possible conditions. Therefore, this is a theoretical limit in which they take advantage of core pipelining. However, the pipeline strategy in a power converter HIL model is not a clear advantage, since the results of each integration step are fedback for the next integration step. Anyhow, it is clear that the performance of floating-point in FPGAs will improve when using HFP cores. The goal in this paper is to quantify the improvement when using this technology for HIL models, comparing it to the performance of the two classical numerical implementations in FPGAs: fixed-point and floating-point without HFP cores. In order to make the comparison, a HIL model is implemented in three ways: using fixed-point representation, and using floating-point representation but in two versions, using HFP cores and implementing the floating-point operations in generic programmable logic (which was the only method until now).

The rest of the paper is organized as follows. Section 2 shows the limits of the current technology for HIL. Section 3 introduces the case of study, the power converter that has been used for the HIL model example. Sections 4 and 5 show the implementation details and experimental results. Finally, Section 6 provides the conclusions.

2. Technology limits and numerical resolution

As explained in the previous section, FPGAs can be used to speed up the simulations of an analog plant. Some software tools allow the designer to describe the plant model with high level language or even with schematics, and they automatically translate the input into synthesizable code. Even though they generate functional code, the results in area and speed are not optimal. For instance, authors in [26] compared the same model using System Generator and hand coded HDL, and the System Generator model ran at half the speed compared to the hand coded model and also obtained worse area results.

In addition, in [29] HIL models are presented using an automatic HDL translator. In [29], the equations for the model are extracted and codified into LabView. The software tool translates the equations and algorithm blocks into synthesizable HDL using fixed-point arithmetic. The authors obtain an integration step of 150 ns for a three-phase inverter simulation and 6.4 μs for a three-phase power distribution network simulation.

Therefore, hand coded HDL code can be used to optimize speed and area. One of the first choices that must be made is the selection of arithmetic. This choice is not a matter of designer predilection but it affects the development speed, speed and area of the model, and its resolution.

The two main eligible arithmetics for a model are fixed-point and floating-point. Fixed-point obtains the best synthesis results, in terms of area and speed. Moreover, in the past, no synthesizable floating-point libraries were available for FPGAs. The main drawback of fixed-point is the design time that it needs. To obtain optimal area and speed results, the designer should decide every signal width and the number of bits dedicated to the integer and fractional parts inside the signal. This process must be done carefully to avoid numerical overflows but also to provide enough fractional bits to achieve the desired numerical resolution as the model accuracy depends on it.

The release of synthesizable floating-point libraries removed the need to choose signal widths. By using floating-point, the designer does not need to care about the magnitude of the signals because the point location changes as needed. However, these libraries have two disadvantages: the aforementioned area and speed results, which are worse than those of fixed-point, and their constrained resolution. The characteristics of floating-point allows the designer to optimize the resolution of a signal because it uses normalized notation (i.e. only one significant integer bit). However, the number of bits for the significand is fixed so signals cannot be adapted if they need to store big magnitudes while keeping low value fractional bits. This circumstance is frequent in HIL systems because many plants are modeled using small integration steps, so it is common to calculate the voltage of a converter, which can be around hundreds of volts, while needing to add small increments of around microvolts. Both must be stored in a single variable, which can lead to resolution problems.

Authors in [30] analyzed the resolution problem and stated that the minimum number of bits needed to represent a number can be calculated as follows:

$$w = \lceil \log_2 \frac{x}{\Delta x} \rceil + n \tag{1}$$

where w is the width of the variable, x is its value and Δx is its increment. n is an additional number of bits to increase the resolution in such a way that particularly small numbers can be accumulated. Authors in [30] reached the conclusion that the value of n should be around 8 or above.

A fixed-point approach does not require every signal to keep the same size, so optimal widths can be chosen. However, floating-point libraries need normalized signals in order to be added or multiplied. For instance, in the previous example (hundreds of volts and increments of microvolts), the number of bits that the significand should have is around 27+8=35 bits, while the standard significand for single-precision floating-point signals is 24 bits (23 bits plus one implicit bit). Using fewer bits than necessary for the significand leads to accuracy loss and probably to useless simulations. Double-precision floating-point could be used so 53 bits are dedicated to the significand, but area and speed results will probably be unacceptable.

The magnitude of the incremental values depends directly on the application and the integration step. Very different values are obtained when the integration step is one nanosecond or one microsecond. Therefore, floating-point approach can be chosen if the application meets the requirements of Eq. 1 for a significand width of 24 bits. As described in the next section, the incremental value is proportional to the integration step, so single-precision floating-point can be used for applications where a small integration step is not needed.

The other drawback of floating-point that has been mentioned is speed and area of the emulation. Recently, FPGAs with floating-point DSP have been released, mitigating this drawback significantly. For instance, the Intel Arria 10 and Stratix 10 FPGA families provide numerous floating-point DSPs, reaching maximum frequencies of up to 300 MHz (3.33 ns of period). These high frequencies are reached thanks to hardened DSPs, i.e. DSPs built in silicon and not using programmable resources. Nonetheless, it must be noted that the advertised performance is calculated using pipeline. The designer can only take advantage of the pipelined architecture if the plant model is not fedback. If the previous output of the DSP is needed for the input of the present calculation, pipeline is not useful. Therefore, the introduction of floating-point DSP for modeling analog plants must be studied thoughtfully.

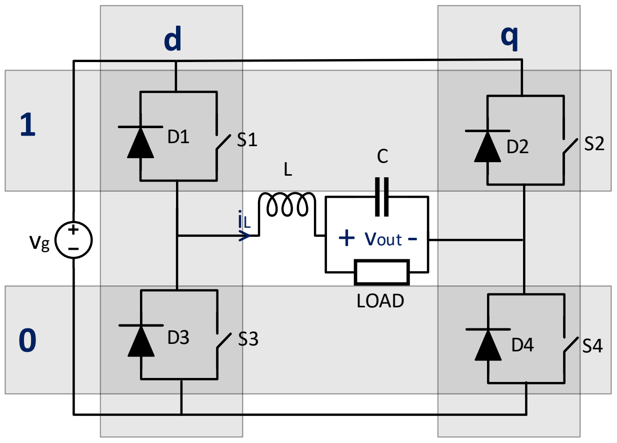

Figure 1: Topology of a full-bridge converter

In this paper, fixed-point and DPS-based floating-point approaches are qualitative and quantitatively compared, using a full-bridge converter as the application example to perform the comparison.

3. Application example

As explained in the previous section, the accuracy of the model is limited by the resolution of the chosen arithmetic. As the variables should store the last value and a small increment, the signal width is critical. While the last value depends directly on the application (e.g. the output voltage of the converter or the load current), the small increments depend also on the simulation step, as shown below in this section. Likewise, the simulation step depends on the application because high switching frequency converters need a small simulation step to improve the resolution of the converter, as the simulation step should be much smaller than the switching period to correctly reflect the duty cycle.

A set of converter parameters (inductance, capacitance, expected voltages and currents, etc.) has been chosen so single-precision floating-point can be used to implement the model. These parameters are defined in Section 5, as they do not affect the equations that model the analog plant.

Figure 1 shows the topology of a basic full-bridge converter. In order to model the converter, every different configuration must be described into equations. Two variables called d and q are declared to represent the current branch that is taken on the left and right sides of the converter, respectively. The current takes one branch if the respective switch is closed or if both switches of the same side (left o right) are open and the respective antiparallel diode is in conduction mode.

Algorithm 1 defines the values of variables d and q considering these conditions. It must be taken into account that the control signals of the four switches (S1 to S4) are the input signals of the model, which are internally translated into d and q. For instance, if S1 is closed, d will take the value 1, while q will take the value 1 when S2 is closed. These values are also taken if D1 or D2 are conducting, as reflected in algorithm 1.

Algorithm 1 Branch selection algorithm depending on the switches and diodes

```
1: procedure BRANCH SELECTION
2:     if $S_1 = ON$ or ($S_1 = OFF$ and $S_3 = OFF$ and $i_L < 0$) then
3:         $d \leftarrow 1$
4:     else
5:         $d \leftarrow 0$
6:     if $S_2 = ON$ or ($S_4 = OFF$ and $S_2 = OFF$ and $i_L > 0$) then
7:         $q \leftarrow 1$
8:     else
9:         $q \leftarrow 0$
```

Figure 2 shows all converter configurations, considering d and q variables. The model proposed in this paper is the simplest one, using fixed time step and allowing synthesizable implementations in HDL. The aim is not to calculate the high frequency transients of the electronic components, but to generate a behavioral model of the power converter for testing the regulator, using a small time step. The model needs to calculate the output voltage (v_{out}) and inductor current (i_L) every time step, taking into account the described configurations.

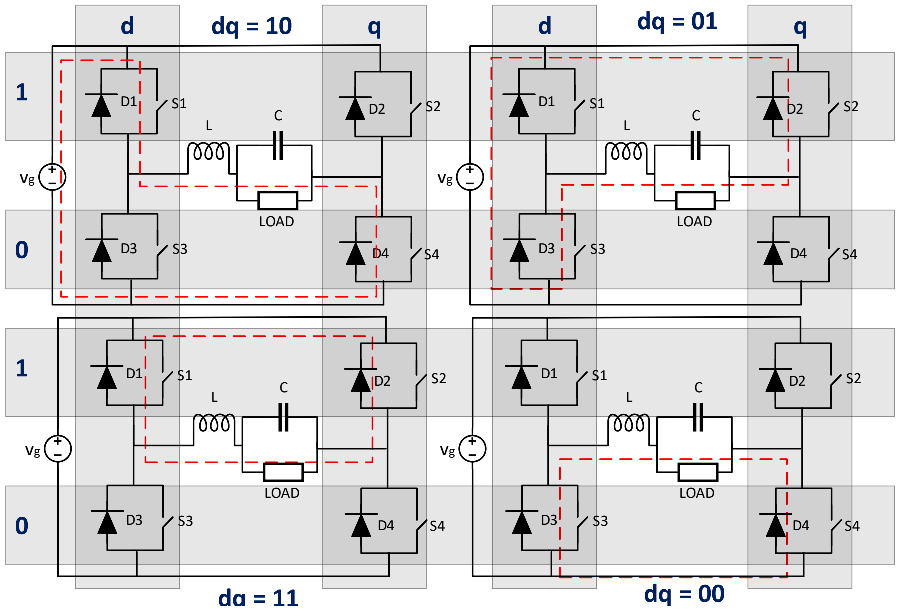

Figure 2: Different running configurations of a full-bridge converter

The capacitor current is defined in (2):

$$i_C \quad = \quad C \cdot \frac{dv_c}{dt} \tag{2}$$

where i_C is the current through the capacitor and v_c is the capacitor voltage, which is equal to the output voltage. Converting (2) into a difference equation, the output voltage for each time step k is defined in (3):

$$v_{out}(k) \quad = \quad v_{out}(k-1) + \frac{\Delta t}{C} \cdot i_C \tag{3}$$

Δt is the time step of the calculus of the state variables so, as the time step is fixed, $\frac{\Delta t}{C}$ and $\frac{\Delta t}{L}$ (see Eq. 5) are constants. The current through the capacitor, i_c, is $i_L - i_R$ regardless of the configurations, considering that i_R is the load current.

Similarly, the inductor voltage is defined in (4):

$$v_L = L \cdot \frac{di_L}{dt} \tag{4}$$

Converting (4) into a difference equation, the inductor current for each time step k is defined in (5):

$$i_L(k) = i_L(k-1) + \frac{\Delta t}{L} \cdot v_L \tag{5}$$

The inductor voltage does depend on the different configurations of the converter. When $dq = 10$, v_L is $v_g - v_{out}$, when $dq = 01$, $v_L = -v_g - v_{out}$ and, finally, when $dq = 11$ or $dq = 00$, $v_L = -v_{out}$. Taking all into account, Eq. 6 must be applied when $dq = 10$, Eq. 7 must be applied when $dq = 01$ and Eq. 8 must be applied when $dq = 11$ or $dq = 00$.

$$\begin{aligned} i_L(k) &= i_L(k-1) + \frac{\Delta t}{L} \cdot (v_g - v_{out}) \\ v_{out}(k) &= v_{out}(k-1) + \frac{\Delta t}{C} \cdot (i_L - i_R) \end{aligned} \tag{6}$$

$$\begin{aligned} i_L(k) &= i_L(k-1) + \frac{\Delta t}{L} \cdot (-v_g - v_{out}) \\ v_{out}(k) &= v_{out}(k-1) + \frac{\Delta t}{C} \cdot (i_L - i_R) \end{aligned} \tag{7}$$

$$\begin{aligned} i_L(k) &= i_L(k-1) + \frac{\Delta t}{L} \cdot (-v_{out}) \\ v_{out}(k) &= v_{out}(k-1) + \frac{\Delta t}{C} \cdot (i_L - i_R) \end{aligned} \tag{8}$$

For the sake of clarity, the previous equations do not consider any electrical losses. The addition of electrical losses obviously affects the performance of the simulation because it increases the minimum clock period. In order to continue reaching real-time, the simulation step should also be increased. Therefore, a trade-off between simulation step and accuracy should be reached.

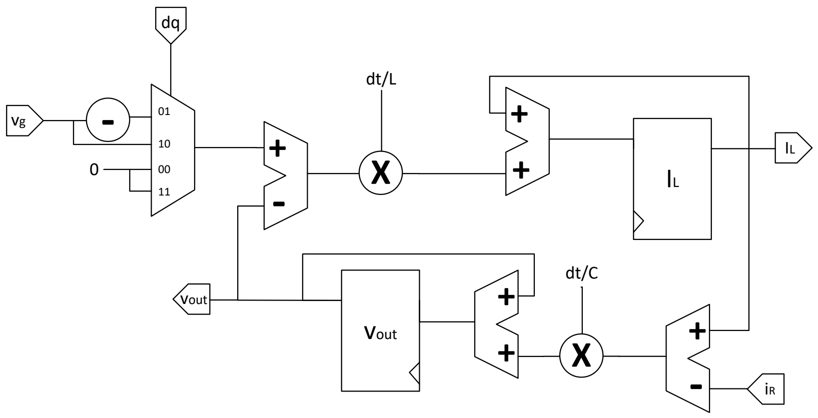

Figure 3: Schematic of a full-bridge converter model

Once the equations are defined, the model must be implemented using either fixed-point or floating-point arithmetic. Regardless of the arithmetic, the hardware that must be implemented is shown in Figure 3. The next section presents the implementation details of both implementations.

4. Hardened Floating-point DSP Blocks

Support for HFP cores has been marketed in the programmable logic world since Intel presented the 20-nm Arria 10 family in 2014. This new family includes DSP blocks for fixed-point arithmetic and IEEE 754 single-precision floating-point arithmetic. This is the 32-bit IEEE 754 format with an 8-bit exponent and 24-bit significand (23 bits and 1 implicit bit) encoding. Despite other configuration possibilities, Arria 10 DSPs can be configured in floating-point mode to execute one single-precision addition, multiplication, accumulation, one multiply-add operation, or one multiply-accumulate operation. Furthermore, up to four pipeline stages are available in floating-point mode. Each pipeline stage can be optionally bypassed, which is particularly useful in a typical closed-loop control [31, 32].

Recently, [33] has used the function log(x) to illustrate the HFP block benefits, showing significant reduction in logic resources and performance flexibility

Table 1: Full-bridge parameters used in the Results section

Parameter	C	L	V_{in}	R_{Load}	T_{sw}
Value	100 μF	900 μH	200 V	12 Ω	50 μs

compared to current methods.

HFP blocks have several benefits. Firstly, they reduce soft logic requirements. Therefore, as floating-point operations fit on embedded features, implementations reduce area consumption and increase clock frequencies. Secondly, although it is beyond the scope of this paper, power consumption is expected to be lower. Thirdly, development time can be significantly reduced as there is no need for manual translation into fixed-point implementations, which includes extensive tests to guarantee that the resulting precision is still acceptable for the application.

Intel HFP blocks can be synthesized from OpenCL and Simulink models by means of high-level synthesis tools. However, in this work Intel Megafunctions were used and instantiated into HDL models.

5. Results

5.1. Testing methodology

In this section, all the arithmetics mentioned in the previous sections are compared: fixed-point, floating-point using programmable logic, and floating-point using HFP cores. Fixed-point and floating-point using programmable logic have been implemented using the VHDL 2008 fixed-point and floating-point libraries [34]. All full-bridge models have been configured using the parameters shown in Table 1. As it can be seen, a resistive load has been chosen for the experiments. On the other hand, the switching period (T_{sw}) of the full-bridge model is 50 μs. As it will be seen, the proposed systems achieve simulation steps of around 53 ns (18.70 MHz of running frequency). Therefore, the simulation step is much lower than the switching period, achieving accurate results and simulating even the switching ripple of the state variables.

The proposed models have been tested in open loop, without using any closed loop regulators. If closed loop regulators were used, they would compensate the numerical errors of the model, so the whole system would get the desired values at steady state even if the model did not have enough numerical resolution. For that reason, the control of the full-bridge model has been implemented with a simple DPWM (Digital Pulse Width Modulation) with a switching frequency (T_{sw}) of 50 μs, and choosing the appropriate duty cycles. The model reads the switches control signals, which are the switches states, then chooses the selected branches (as shown in Algorithm 1) and finally applies the pertinent equations as it was seen in the schematic of Fig. 3. It should be mentioned that, although in this example PWM signals are used for the control, the model actually reads the instantaneous values of the switches control signals, which are the inputs of the model, so any modulation can be used, without requiring constant frequency or any other restriction.

The evaluation of the proposed systems is done by instantiating the different models, collecting the state variable values, and comparing those values with those of a reference model. The reference model in VHDL is based on variables of *real* type and an integration step of 1.25 ns. *Real* type has been chosen as reference because it is implemented with double-precision floating-point, so it has 53 bits of significand. Therefore, it has enough resolution even when the simulation step is around a nanosecond.

The accuracy of each converter model is evaluated considering the mean absolute error values of i_L and v_{out} during the simulation. It should be mentioned that the relative error cannot be applied since there are situations when the denominator values become zero and the relative error is indeterminate.

Multiple simulations have been accomplished in order to simulate different conditions: input voltage transients, changes on the duty-cycle, load transients and also simulation of steady states. All the experiments show the same behavior regarding the different arithmetics. Therefore, for the sake of clarity, only the results for one representative experiment are shown in this paper.

The simulation that has been chosen to explain the results has been accom-

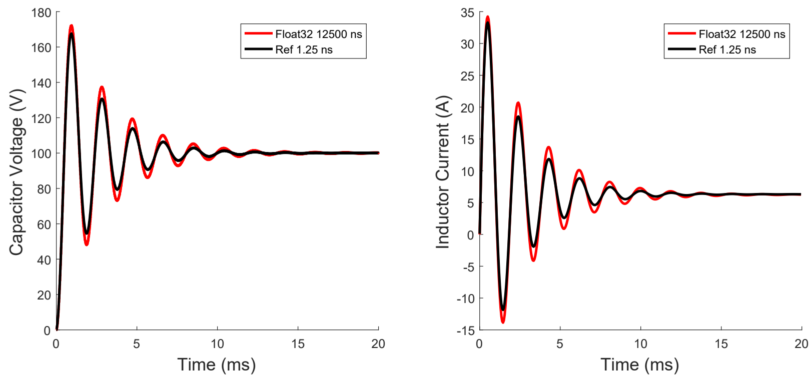

Figure 4: Experiment chosen to extract the accuracy results

plished by triggering a transient from off-state ($V_{out} = 0\ V$) to a duty cycle of 75% ($V_{out} = 100\ V$ at steady-state), as it is shown in Fig. 4. This simulation contains a start-up condition with a big transient and finally the model reaches steady state. The transient allows us to check the dynamics of the model for all the arithmetics, while the steady state is used to know if the model reaches the expected values even if the dynamics were not accurate. As it can be seen in Fig. 4, the simulation duration has been set at 20 ms. The example figure shows a comparison between the reference model and a floating-point model with a simulation step of $12,500\ ns$ so the error can be visually appreciated. In most other cases the error is so small that it can only be numerically analyzed.

5.2. Integration Step vs. Error

Figures 5 and 6 show the relation between integration step dt and absolute error in current i_L and voltage v_{out}, respectively. When the model based on *real* type is used, as the integration step decreases, so does the error all along the evaluated domain. This is the expected behavior because as the integration step is reduced, the difference equations shown in (6), (7) and (8) are more accurate. However, the rest of the models have two limits that should be considered.

First of all, the model should run at real-time so the integration step must be equal to the clock period. Hence, the integration step reduction is limited by the minimum achievable clock period for the current technology. It can be seen in Figures 5 and 6 that fixed-point and HFP-based models have their limits very close, while the limit is considerably higher for float32. With the evolution of the technology, these limits will surely descend so, from a point of view of technology, the models would be improved.

On the other hand, there is another limit imposed by the resolution of the arithmetic. If the simulation step is reduced below 50 ns, 32-bit floating-point arithmetic presents resolution issues, and the reduction of the simulation step is counterproductive. This problem is caused by the arithmetic, so it cannot be avoided unless the width of the signals are increased. The problem of using wider variables is that, if a non-standard floating-point format were used, the model would not take benefit of the HFP cores, so the simulation step would be increased.

The resolution problem is not present in the case of fixed-point representation (*Sfixed* in Figures 5 and 6), as the widths of the variables can be adapted as needed. Therefore, the fixed-point model is parameterizable and the resulting area consumption and time figures vary accordingly. This parametrization allows the resolution to increase as the integration step decreases. This situation can be seen in Figs. 5 and 6, where the accuracy of sfixed model is always proportional to the simulation step. On the other hand, the main problem with fixed-point is the time needed for the design process itself.

As it can be seen, the decision about which arithmetic should be used is not trivial. As mentioned in Section 2, the decision about using an arithmetic should be taken by considering the application, i.e., the simulation step and the expected values for the state variables. As a starting-point, the model based on $float32$ type, using programmable logic and VHDL 2008 library for floating-point, could be the first implementation option in terms of design effort. If the simulation step of the model should be lower, HFP-cores can be used while the application allows the arithmetic to be accurate enough. If resolution issues appear, fixed-point should be chosen, as it allows the designer to chose the width of every variable while optimizing the model.

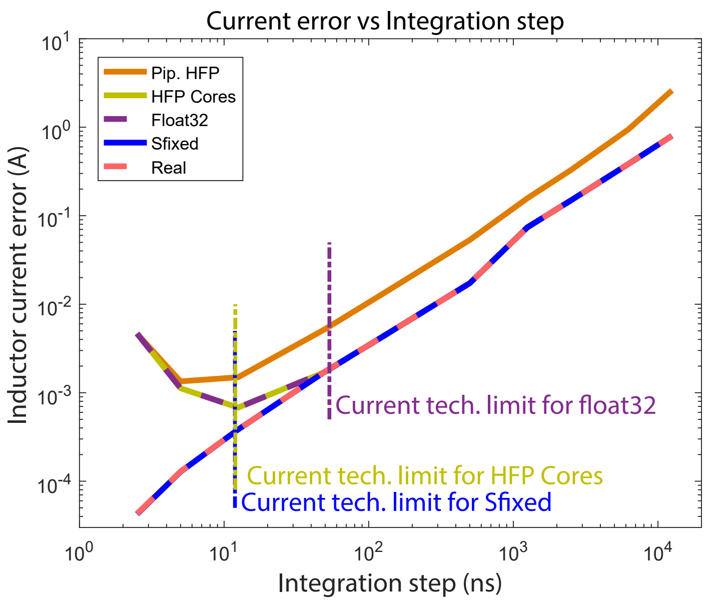

Figure 5: Absolute errors in current

5.3. Area and Time

Table 2 shows area and time figures for all the implemented full-bridge versions. These are post place and route results obtained by the Quartus Prime Standard Edition tool, version 16.0.2. The selected device is 10AX016E4F27E3LP. It belongs to the set of the smallest devices of the GX product line, with 160k logic elements and 156 variable-precision DSP blocks. The different case study versions were synthesized and implemented using the default tool options and a timing constraint of 12 ns, in order to obtain the fastest circuits. The $sfixed$ row shows the results for an integration step of 12.5 ns, which is the best synthesizable period in the selected technology. As the model based on fixed-point arithmetic is parameterizable and it depends on the integration step, detailed results are shown in Table 3. Area (ALM) saving and acceleration are compared to the numbers of the implementation based on $float32$. Note that using HFP cores, the required area is 222 times better than for the required by the $float32$ version while the acceleration is as high as that of the $sfixed$ version (taking

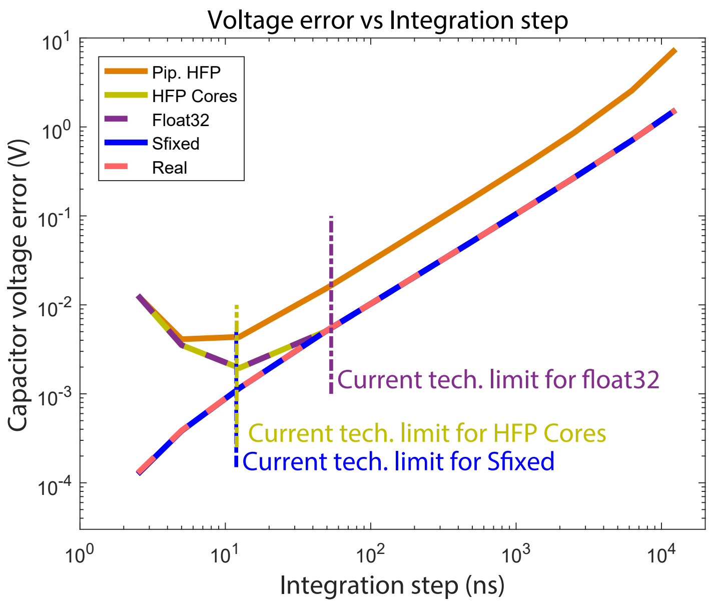

Figure 6: Absolute errors in output voltage

into account ALMs, but not registers or DSP blocks).

A pipelined model using HFP cores has also been implemented. Last stage registers of the HPF cores that implement the subtractor are enabled to implement the pipeline registers. In this way, the register number in the programmable logic is the same as that for the combinational models. As mentioned above, the proposed model cannot take advantage of pipeline because it has to calculate two state variables that depend on each other (see Fig. 3). Because of this dependence, the state variable registers are activated only every second clock cycle. For instance, the current must wait the output voltage to be ready, so there is an idle cycle between two functional cycles.

In table 2, Column F_{ef} shows the effective maximum model frequency, related to the minimum integration step allowed by the current technology. F_{ef} is equal to F_{max} in all cases but except for the HFP pipelined version using a pipeline of two stages (HFP Pip. row).

As predicted, the clock frequency in the pipelined version is higher but the

Table 2: Area and time results for Full Bridge converter

	Area				Frequency (MHz)		
Version	ALMs	Saving	Registers	DSP blocks	F_{max}	F_{ef}	Acceleration
$float32$	4670	1x	64	2	18.70	18.70	1.0x
$sfixed$	301	16x	128	2	84.12	84.12	4.5x
HFP Cores	21	222x	64	4	83.62	83.62	4.5x
HFP Pip.	20	234x	64	4	130.04	65.02	3.5x

Table 3: Area and time results for sfixed

	Area				Frequency (MHz)	
dt (ns)	ALMs	Saving	Registers	DSP blocks	F_{max}, F_{ef}	Acceleration
2.5	304	15x	133	2	86.13	4.6x
5	302	15x	131	2	85.64	4.6x
12.5	301	16x	128	2	84.12	4.5x
50	294	16x	124	2	84.61	4.5x
125	287	16x	121	2	84.04	4.5x
500	285	16x	117	2	86.49	4.6x
1250	277	17x	115	2	84.90	4.5x
2500	276	17x	113	2	88.70	4.7x
6250	317	15x	111	1	87.24	4.7x
12500	262	18x	108	1	88.47	4.7x

effective frequency is lower because every state variable is actually only updated in alternate cycles. Moreover, the effective frequency is smaller than that of the non-pipelined model because pipeline adds delays (it enables two registers, it requires register enable logic, etc.).

Finally note that, as the integration step increases in sfixed models, the required area decreases, as shown in Table 3. The exception is when $dt = 6250\ ns$ or higher; in this case, one -and not two- DSP core is used for the lowest dt values.

6. Conclusions

This paper has presented a comparison between HFP-based, floating-point with programmable logic and fixed-point representation to implement a power converter model. The conclusion is that there are three points to consider when deciding the most appropriate representation for each specific model: design effort, maximum execution frequency and numerical resolution. Regarding the design effort, the preferred choice is to use floating-point in any of its two forms in order to avoid calculating the width of every model variable. When considering maximum execution frequency, HFP cores and fixed-point achieve similar results, about x4.5 faster than floating-point with programmable logic. Although pipeline can increase the maximum clock frequency, the overall system speed is decreased, so it can be discarded for HIL models. Finally, regarding numerical resolution, it has been shown that the problem can appear for small simulation steps. In the converter used for the experimental results, numerical resolution problems start to arise in 32-bit floating-point when the simulation step approaches the minimum possible for real time in this technology (about 10 ns). However, in other topologies or with other parameters, resolution issues may appear for larger or smaller simulation steps, depending on the relation between the values of the state variables and their increments, which are proportional to the simulation step. As fixed-point variable widths are customized for the application, this problem can be completely avoided.

As a summary, there is no right decision for all cases, but the most appropriate representation must be chosen for each application. Anyhow, HFP cores have inclined the balance towards floating-point, which will be probably the best choice unless resolution issues appear in a specific application.

Acknowledgment

This work has been supported by the Spanish Ministerio de Economía y Competitividad under project TEC2013-43017-R

References

[1] B. Patella, A. Prodic, A. Zirger, D. Maksimovic, High-frequency digital PWM controller IC for DC-DC converters, Power Electronics, IEEE Transactions on 18 (1) (2003) 438–446. doi:10.1109/TPEL.2002.807121.

[2] A. Peterchev, J. Xiao, S. Sanders, Architecture and IC implementation of a digital VRM controller, Power Electronics, IEEE Transactions on 18 (1) (2003) 356–364. doi:10.1109/TPEL.2002.807099.

[3] Maksimovic, Zane, Erickson, Impact of digital control in power electronics, in: 2004 Proceedings of the 16th International Symposium on Power Semiconductor Devices and ICs, 2004, pp. 13–22. doi:10.1109/WCT.2004.240284.

[4] S. Buso, L. Malesani, P. Mattavelli, Comparison of current control techniques for active filter applications, IEEE Transactions on Industrial Electronics 45 (5) (1998) 722–729. doi:10.1109/41.720328.

[5] D. M. V. de Sype, K. D. Gusseme, A. P. M. V. den Bossche, J. A. Melkebeek, Duty-ratio feedforward for digitally controlled boost PFC converters, IEEE Transactions on Industrial Electronics 52 (1) (2005) 108–115. doi:10.1109/TIE.2004.841127.

[6] M. W. Naouar, A. A. Naassani, E. Monmasson, I. Slama-Belkhodja, Fpga-based predictive current controllerfor synchronous machine speed drive, IEEE Transactions on Power Electronics 23 (4) (2008) 2115–2126. doi:10.1109/TPEL.2008.924849.

[7] M. Salehifar, M. Moreno-Eguilaz, G. Putrus, P. Barras, Simplified fault tolerant finite control set model predictive control of a five-phase inverter supplying {BLDC} motor in electric vehicle drive, Electric Power Systems Research 132 (2016) 56 – 66. doi:http://dx.doi.org/10.1016/j.epsr.2015.10.030.

URL http://www.sciencedirect.com/science/article/pii/S0378779615003260

[8] H. Berriri, W. Naouar, I. Bahri, I. Slama-Belkhodja, E. Monmasson, Field programmable gate array-based fault-tolerant hysteresis current control for ac machine drives, IET Electric Power Applications 6 (3) (2012) 181–189. doi:10.1049/iet-epa.2011.0053.

[9] Z. Tir, O. P. Malik, A. M. Eltamaly, Fuzzy logic based speed control of indirect field oriented controlled double star induction motors connected in parallel to a single six-phase inverter supply, Electric Power Systems Research 134 (2016) 126 – 133. doi:http://dx.doi.org/10.1016/j.epsr.2016.01.013.
URL http://www.sciencedirect.com/science/article/pii/S037877961600016X

[10] A. S. Vijay, S. Doolla, M. C. Chandorkar, Real-time testing approaches for microgrids, IEEE Journal of Emerging and Selected Topics in Power Electronics 5 (3) (2017) 1356–1376. doi:10.1109/JESTPE.2017.2695486.

[11] B. Lu, X. Wu, H. Figueroa, A. Monti, A low-cost real-time hardware-in-the-loop testing approach of power electronics controls, Industrial Electronics, IEEE Transactions on 54 (2) (2007) 919–931. doi:10.1109/TIE.2007.892253.

[12] B. Jandaghi, V. Dinavahi, Hardware-in-the-loop emulation of linear induction motor drive for maglev application, IEEE Transactions on Plasma Science 44 (4) (2016) 679–686. doi:10.1109/TPS.2016.2535460.

[13] S. Karimi, P. Poure, S. Saadate, An hil-based reconfigurable platform for design, implementation, and verification of electrical system digital controllers, Industrial Electronics, IEEE Transactions on 57 (4) (2010) 1226–1236. doi:10.1109/TIE.2009.2036644.

[14] G. Parma, V. Dinavahi, Real-time digital hardware simulation of power electronics and drives, Power Delivery, IEEE Transactions on 22 (2) (2007) 1235–1246. doi:10.1109/TPWRD.2007.893620.

[15] M. Matar, R. Iravani, FPGA implementation of the power electronic converter model for real-time simulation of electromagnetic transients, Power Delivery, IEEE Transactions on 25 (2) (2010) 852–860. doi:10.1109/TPWRD.2009.2033603.

[16] A. Myaing, V. Dinavahi, FPGA-based real-time emulation of power electronic systems with detailed representation of device characteristics, Industrial Electronics, IEEE Transactions on 58 (1) (2011) 358–368. doi:10.1109/TIE.2010.2044738.

[17] Y. Chen, V. Dinavahi, Digital hardware emulation of universal machine and universal line models for real-time electromagnetic transient simulation, Industrial Electronics, IEEE Transactions on 59 (2) (2012) 1300–1309. doi:10.1109/TIE.2011.2157296.

[18] J.-Y. Dieulot, F. Colas, L. Chalal, G. Dauphin-Tanguy, Economic supervisory predictive control of a hybrid power generation plant, Electric Power Systems Research 127 (2015) 221 – 229. doi:http://dx.doi.org/10.1016/j.epsr.2015.06.006.
URL http://www.sciencedirect.com/science/article/pii/S0378779615001819

[19] S. Dennetire, H. Saad, B. Clerc, J. Mahseredjian, Setup and performances of the real-time simulation platform connected to the {INELFE} control system, Electric Power Systems Research 138 (2016) 180 – 187, special Issue: Papers from the 11th International Conference on Power Systems Transients (IPST). doi:http://dx.doi.org/10.1016/j.epsr.2016.03.008.
URL http://www.sciencedirect.com/science/article/pii/S0378779616300554

[20] K. Boroojeni, M. H. Amini, A. Nejadpak, T. Dragievi, S. S. Iyengar, F. Blaabjerg, A novel cloud-based platform for implementation of oblivious power routing for clusters of microgrids, IEEE Access 5 (2017) 607–619. `doi:10.1109/ACCESS.2016.2646418`.

[21] E. Monmasson, L. Idkhajine, M. Cirstea, I. Bahri, A. Tisan, M. Naouar, FPGAs in industrial control applications, Industrial Informatics, IEEE Transactions on 7 (2) (2011) 224–243. `doi:10.1109/TII.2011.2123908`.

[22] E. Monmasson, L. Idkhajine, M. Naouar, FPGA-based controllers, Industrial Electronics Magazine, IEEE 5 (1) (2011) 14–26. `doi:10.1109/MIE.2011.940250`.

[23] J. Rodriguez-Andina, M. Moure, M. Valdes, Features, design tools, and application domains of FPGAs, Industrial Electronics, IEEE Transactions on 54 (4) (2007) 1810–1823. `doi:10.1109/TIE.2007.898279`.

[24] J. Alvarez, O. Lopez, F. Freijedo, J. Doval-Gandoy, Digital parameterizable VHDL module for multilevel multiphase space vector PWM, Industrial Electronics, IEEE Transactions on 58 (9) (2011) 3946–3957. `doi:10.1109/TIE.2010.2100334`.

[25] F. Taeed, Z. Salam, S. Ayob, FPGA implementation of a single-input fuzzy logic controller for boost converter with the absence of an external analog-to-digital converter, Industrial Electronics, IEEE Transactions on 59 (2) (2012) 1208–1217. `doi:10.1109/TIE.2011.2161250`.

[26] A. Sanchez, A. de Castro, J. Garrido, A comparison of simulation and hardware-in-the-loop alternatives for digital control of power converters, IEEE Transactions on Industrial Informatics 8 (3) (2012) 491–500. `doi:10.1109/TII.2012.2192281`.

[27] J. Tyhach, M. Hutton, S. Atsatt, A. Rahman, B. Vest, D. Lewis, M. Langhammer, S. Shumarayev, T. Hoang, A. Chan, D. M. Choi, D. Oh, H. C.

Lee, J. Chui, K. C. Sia, E. Kok, W. Y. Koay, B. J. Ang, Arria$^{\text{TM}}$10 device architecture, in: 2015 IEEE Custom Integrated Circuits Conference (CICC), 2015, pp. 1–8. doi:10.1109/CICC.2015.7338368.

[28] M. Langhammer, B. Pasca, Single precision natural logarithm architecture for hard floating-point and dsp-enabled fpgas, in: 2016 IEEE 23nd Symposium on Computer Arithmetic (ARITH), 2016, pp. 164–171. doi:10.1109/ARITH.2016.20.

[29] R. Razzaghi, M. Mitjans, F. Rachidi, M. Paolone, An automated FPGA real-time simulator for power electronics and power systems electromagnetic transient applications, Electric Power Systems Research 141 (2016) 147 – 156. doi:http://dx.doi.org/10.1016/j.epsr.2016.07.022.
URL http://www.sciencedirect.com/science/article/pii/S0378779616302863

[30] O. Goñi, A. Sanchez, E. Todorovich, A. de Castro, Resolution analysis of switching converter models for hardware-in-the-loop, IEEE Transactions on Industrial Informatics 10 (2) (2014) 1162–1170. doi:10.1109/TII.2013.2294327.

[31] Altera Corp., Arria 10 Core Fabric and General Purpose I/Os Handbook (2016).

[32] J. Tyhach, M. Hutton, S. Atsatt, A. Rahman, B. Vest, D. Lewis, M. Langhammer, S. Shumarayev, T. Hoang, A. Chan, D. M. Choi, D. Oh, H. C. Lee, J. Chui, K. C. Sia, E. Kok, W. Y. Koay, B. J. Ang, Arria 10 device architecture, in: Custom Integrated Circuits Conference (CICC), 2015 IEEE, 2015, pp. 1–8. doi:10.1109/CICC.2015.7338368.

[33] M. Langhammer, B. Pasca, Single precision natural logarithm architecture for hard floating-point and dsp-enabled fpgas, in: 2016 IEEE 23nd Symposium on Computer Arithmetic (ARITH), 2016, pp. 164–171. doi:10.1109/ARITH.2016.20.

[34] Ieee standard vhdl language reference manual, IEEE Std 1076-2008 (Revision of IEEE Std 1076-2002) (2009) c1–626doi:10.1109/IEEESTD.2009.4772740.