Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.
Digital Object Identifier XX.XXXX/YYY.20XX.ZZZZZZZ

femtoCore: An Application Specific Processor for Vertically Integrated High Performance Real-Time Controls

Filippo Savi[1], (Student Member, IEEE), Jayakrishnan Harikumaran[3], (Student Member, IEEE), Davide Barater[2], (Senior Member, IEEE), Giampaolo Buticchi[1], (Senior Member, IEEE) Chris Gerada[1,3], (Senior Member, IEEE), Pat Wheeler[3], (Fellow, IEEE),
[1]Key Laboratory of More Electric Aircraft Technology of Zhejiang Province, University of Nottingham Ningbo China, Ningbo 315100
[2]Università degli studi di Modena and Reggio Emilia, Modena, Italy
[3]University of Nottingham, Nottingham, United Kingdom

Corresponding author: Filippo Savi (e-mail: filippo.savi@nottingham.edu.cn).

This work is supported by Ministry of Science & Technology under National Key R&D Program of China, under Grant 2021YFE0108600.
This work is supported by Ningbo Science and Technology Bureau under S&T Innovation 2025 Major Special Programme with grant No. of 2019B10071.
This work is supported by the Key International Cooperation of National Natural Science Foundation of China under Grant 51920105011

ABSTRACT In applications that require a high availability and high performance (for example aerospace),modular power electronics and multi-phase machines represent an advantageous choice. In this framework, a control system able to handle a high number of PWM signals and communication interfaces as well as featuring a high computational power is required. This paper proposes a novel HDL plus soft-core approach to be implemented on System-on-Chip hardware which allows for the efficient and modular implementation of the modern control techniques with strong guarantees in terms of determinism. The proposal lies in the adoption of a very simplified and optimized floating-point soft-core, the femtocore (fCore) and its tool-chain, which allows C-like implementation of complex algorithms in a HDL-design power electronics control framework. Several fCore units can be arranged for parallel processing to handle the time requirements of a complex modular system even with low sampling time (100 kHz or more). The proposed architecture is experimentally validated in a proof-of-concept, six-phase electric machine including a comparison against a traditional method.

INDEX TERMS Digital signal processors, Control system implementation, Current control, Machine drive

I. INTRODUCTION

ADVANCES in the field of static power conversion and machine drives have started to put a lot of strain on the traditional architectures used for their control systems. The first of several factors contributing to this issue is the advent of wide-bandgap devices, Silicon Carbide (SiC) and Gallium Nitride (GaN) transistors, which can be operated at very high switching frequencies in the range of hundreds of kilohertz to megahertz range [1], [2], without significant increases in switching losses, reducing the need for bulky and expensive filtering components. Another trend exacerbating this issue is the ever growing adoption of modern high performance control techniques, such as Model Predictive control (MPC) [3], [4] where a mathematical model is used to predict the effect that the controller actions will have on the physical system, one or more cycles in the future. This allows not only to achieve the desired action from the system, but also minimize (or maximize) other performance metrics, that are typically not actively controlled in simpler control systems, through their inclusion in the cost function. The main drawback of these methods, is the large computational complexity, as a mathematical model needs to be evaluated multiple times per period. Last but not least, there is a heavier focus on fault tolerance due to a strong push toward transport electrification. In this field, failure of fundamental systems, can have extremely severe consequences, that are not typically encountered in an industrial automation setting. In fields like marine and offshore drilling [5] or aerospace [6]–[8], a failure in one of the critical mission components can easily have catastrophic consequences, like the loss of a vehicle, large oil spills and potentially even loss of lives. For these reasons, both reliability and fault tolerance are key design specifications in such mission critical applications. To address this need, machine and converter architectures [9], [10] that can tolerate one or more faults, are being studied. The introduction of redundancy at multiple levels, has the unwanted side effect of further increasing the computational needs of control systems. The net effect of all these trends, is a growing inadequacy of the traditional control system architecture [11], where a monolithic microcontroller (MCU) is directly in charge of the whole system. Vertical and horizontal scaling has been proposed as a solution, by silicon manufacturers, which propose both high frequency and multi-core microcontroller designs. These techniques, although effective in increasing computational power of these platforms, have the drawback of decreasing the overall system determinism and execution time consistency, as jitter increases. As high frequency designs need to employ caches and deeper pipelines in order to hit the required frequencies, and multi-core processors, while utilizing unaltered cores, need to deal with the problems brought about by inter process communication (IPC) and shared resource access contention,with similar effects [12]–[14]. While these trade-offs might be acceptable in most applications, they are not acceptable for mission critical hard real-time. In such a setting a significant amount of execution time jitter increase cost; as computational resources need to be over-provisioned to a degree where deadline overrun are certifiably avoided in the worst case scenario. Field Programmable Gate Arrays (FPGA) have also been proposed for the full implementation of control systems, as in [15]–[17]. While these type of devices and architectures can scale to massive systems with very high computational capabilities, they are much more complex to realise due to the difficulty of HDL (Hardware definition Language) development as opposed to software development. Finally, commercial Hardware-in-the-Loop (HIL) systems can be used to implement the whole control system, as in [18], [19] directly from simulation models, in a black box manner, typically using conventional processors, with some FPGA assistance. While this last option massively simplifies implementation, but is also fairly limited for high speed application, where the latency of these platform becomes a bottleneck, in addition to their high cost.

In this paper a novel control system implementation architecture is presented, based on a FPGA/SoC platform containing programmable logic and a dual core processor in the same package (Xilinx Zynq or equivalent), it includes a custom processor core, specifically developed to assist in the implementation of digital control systems. It features:

- Deterministic and constant execution time.
- Purely software based control system implementation.
- Intermediate FPGA architecture between soft-core and full HDL.
- Cost-effective industrial deployment capability.

In Section II the overall system architecture is presented, in Section III the custom processor core is detailed, in Section IV the proposed core and architecture are compared with a traditional implementation. In Section V a real world system is presented to experimentally validate the architecture performance and then conclusions are drawn in Section VI.

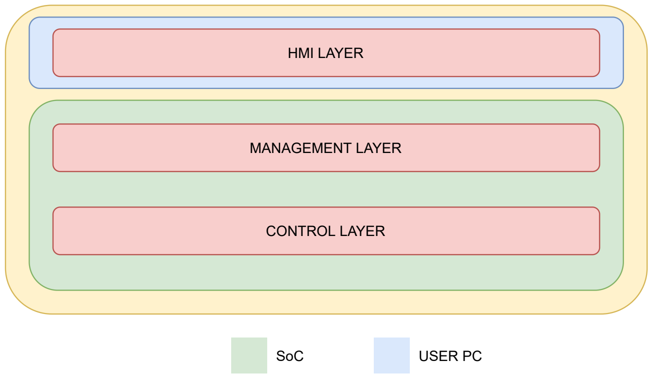


FIGURE 1. System Layers diagram.

II. SYSTEM ARCHITECTURE

The proposed system architecture is constituted by three separate layers, as shown in Fig. 1, each one responsible for a subset of functionalities, starting from the top we find the HMI (Human Machine Interface) layer, that translates the users commands to a form that can be easily acted upon by the underlying layers. The management layer offloads the most common tasks like configuring the FPGA with the correct bitstream and programming the processing core, along with being in charge of storing all user and application specific information. The lowest layer, the control layer, is the one where all the hard real-time control functions are implemented.

The HMI is running directly on the user PC, leveraging modern web standards, through a web application, allowing easy provisioning of the whole system, with no additional software installation required. All other layers, that form the backend server to the application, are implemented in a System on Chip (SoC) that is comprised by a programmable logic (PL) part, which contains a FPGA fabric, and a dual core ARM cortex A9 processor system, with relative peripherals. Client and server can thus be connected through a standard wired or wireless network connection capable of supporting TCP/IP standards.

A. CONTROL LAYER

The control layer, shown in Fig. 2 is responsible for the hard real-time aspects of the whole system. It interfaces with external sensors, calculates the required control actions and then passes them along to the rest of the system. The first step is fulfilled through several standard interfaces (SPI, I2C, etc.) that connect the main SoC with the external ADCs and sensors. The raw data is then processed with ad-hoc logic and DSP blocks to perform calibration and filtering. Then the treated samples are inserted through Direct Memory Access (DMA) in the register space of the processing core, that calculates the required control actions. Once this is complete, another DMA engine will extract the result and push them out to the communication units, so that the information can

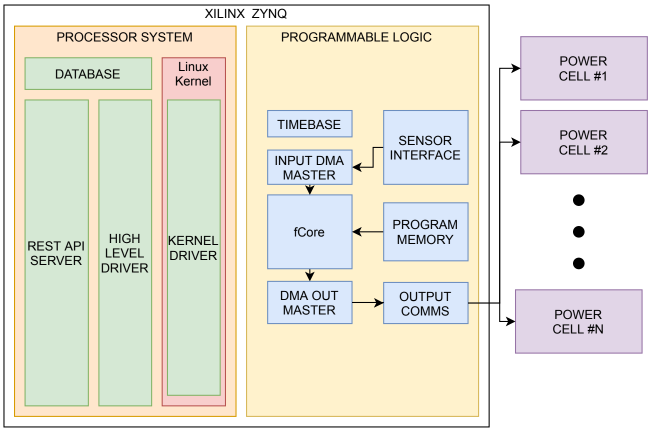


FIGURE 2. SoC architecture diagram.

be pushed out to to each power cell for actuation. A highly configurable timebase module is used to synchronise all the logic in the system, enabling arbitrary variation in frequency and timing of the various phases.

B. MANAGEMENT LAYER

The main purpose of this layer is to abstract away the complexity of the hardware platform, providing a stable interface that the HMI layer can build upon. To this effect, the layer performs multiple tasks. Being a pure software component, without hard real-time constraints, several implementation options are possible. On one hand, a certified real-time OS (RTOS) can be used, if needed for safety related regulatory reasons. On the other hand, if permitted, a standard Linux system can deliver better security, thanks to a much more audited code base, ease of use, with a standard compliant, secure and fully featured networking stack and access to a range of other technologies. To ease deployment and increase manageability, the whole layer makes heavy use of containerization. A side benefit of this approach is the establishment of a clear and distinct interface between the HMI and other user facing components, that could potentially be connected to the internet, and the rest of the system. This separation allows easy authentication and enhance system security, through the use of firewalls, and the application of a defense in depth technique, where multiple independent layers of protection need to be breached to fully gain control of the system.

The first and only user facing component of this layer is a REST server that acts as the back-end for the HMI application and functions as a gateway between it and the rest of the system. All application specific and configuration data is stored in a separate database that allows the whole system to be self contained, and completely independent from the client machine. Finally, the driver component communicates with the server and upon its commands acts on the control layer accordingly. This is comprised of two separate parts: the first is a lower level loadable kernel module (LKM) which abstracts away all the platform dependent functionalities, such as bus and DMA access. The second is a high level C++ driver, implemented as a regular user mode application that does all the low level data processing, such as sorting, filtering, and so on.

C. HUMAN MACHINE INTERFACE

This component, implemented as a standard client side web application, allows the user to interact with the control system in the most natural way possible, allowing on the fly change of parameter, triggering of actions and facilitating data visualization, with a real-time display of selected parameter gathered in the control layer, such as sensor readings, error variables, desired control actions, etc.

In order to allow the user to act on meaningful control variables, rather than having to work directly on the register values exposed by the control layer, whose values need to be parsed and pre-processed through a set of simple operations, like re-scaling, addition of offset, conversion to other formats etc. A simple scripting system has been introduced, that allows the user to program all this required steps. Upon a user triggering of a pre-defined and configurable event, this code will be run, with the results being sent for action to the lower layers. This strikes a balance between complexity of the layer and usability, allowing the user to work with familiar values and parameters, while not requiring a complete redesign of the HMI component structure for each different application.

III. FEMTOCORE PROCESSOR

The implementation of control systems on FPGA can take two routes: the fully custom logic, which can attain much higher operating frequencies, at the cost of a long and complex implementation and verification process, or the serial, through some kind of processor or finite automata. This second choice allows the use of a much quicker software development model, as opposed to the HDL one, with the main drawbacks being longer cycle times and potentially a lack of determinism. The frequency trade off does not impact the considered application as modern FPGA working frequencies are high enough to allow the calculations to take up to few thousand cycles while still respecting the required deadlines. The issue of determinism is much more problematic for mission critical applications, as it brings about all the problems inherent in real-time low jitter safety critical software development, and the related certifications.

Upon close examination of the desired use case, it is apparent that the implementation of most control systems can be reduced to the solution of one or more equations, and provided that all required data is made available by the rest of the system, the processing core's only responsibility is to perform a series of arithmetic and logic operations in order to obtain the required control action. To take advantage of this characteristic of the application, a custom instruction set (ISA) and processor core has been specifically designed in order to allow the implementation of the control system calculations as software, while retaining a fully deterministic execution.

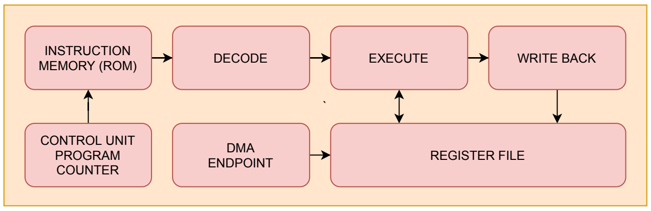


FIGURE 3. High level fCore architecture diagram.

Arithmetic	Logic	Conversion	Saturation	miscellaneous
addition	and	From integer	saturate max.	greater than
subtraction	or	To integer	saturate min.	load constant
product	not			
reciprocal				

TABLE 1. Available operations

A. PROCESSOR ARCHITECTURE

The high level architecture of the designed processor, shown in Fig. 3. The design is fairly traditional, loosely following a harvard architecture, yet with few unconventional characteristics: the base is a simple three stage pipeline with a decoder, an execution unit and finally result write-back. It is derived from the classic five stage RISC pipeline, eliminating the instruction fetch stage, as all instructions have a fixed size, and the memory access, not needed in this architecture. The core utilizes a single pool of memory, denominated register file, to hold all intermediate values, the size of which is limited by the width of the operand and destination register specified in the ISA. In the currently proposed implementation this consists of thirty-two 32-bit general purpose registers (r1 to r31) and a single zero register (r0), that holds the constant value of zero and is used internally to implement several virtual operations, such as register to register data movement and the no operation (also known as nop). The current structure of the ISA, also allows extension of the memory pool to 512 words while keeping a 32 bit instruction word, through 9 bit register addresses. A control unit starts the linear flow of execution upon reception of an external trigger signal and halts it when reaching either the stop instruction or the end of the program memory. This unit also contains the program counter that is advanced once for each execution cycle. A DMA endpoint is also present that allows the external logic to load the inputs directly in the register file, at the appropriate location, and to extract the results once the done signal is issued. Each one of the available operations, listed in Tab. 1, can be encoded in a single 32-bit wide instruction, except for the constant load which takes two. The most important feature of this core is the lack of any control flow instruction, such as conditional or unconditional jumps. This guarantees that the duration of each execution of a single program to be exactly identical completely eliminating jitter by design. This allows also to easily evaluate ahead of time, during compilation, how long a specific program will take to complete execution, and of consequence its maximum operating frequency.

In a trade off between functionality and resource consumption, the implementation of full floating point division is excluded from the ISA. In its place a reciprocal operation is included allowing the calculation of the $1/x$ fraction, to be multiplied by the desired divided. It should be noted that these two operations are not strictly equivalent, as rounding can lead to differing results between the two. As shown in [20], the error is at most 1.5 Units of Least Precision (ULP), a value small enough to be overshadowed by the much larger uncertainties introduced in feedback control system by noise and other sources of error in sensors or during the analog to digital processing.

In order to reach a reasonable clock frequency, the execution of a single floating point operation is implemented as a five stage pipeline, that once filled, can process an instruction every clock cycle. It should be noted that in traditional design a long pipeline is undesirable since it might need to be flushed upon a jump or branch instruction, leading to uncertainty in the run time and decreasing throughput. In the femtoCore, the execution time is largely independent from the pipeline length, that, once full, will never be flushed, leading to a fully deterministic execution time.

The only minor downside from the multi cycle operation of the execution unit is the need for delay slots in the program to avoid data hazards when the next adjacent instructions are co-dependent. This addition can be performed either by the compiler/assembler, through a simple static analysis pass, or in hardware, with a small increase in front-end complexity.

When dealing with uncoupled multi-channel systems, as in the intended application, the same program will need to be executed multiple times, once for each channel. To benefit this use case, the core has support for automated Single Instruction Multiple Data (SIMD) execution. When this mode is enabled, the execution is interleaved, executing each instruction on every channel before advancing the program counter, the Register file is also expanded to have a full complement of thirty-two registers for each channel. A channel counter helps to select the active partition of the full register file. When operating in this mode since no data dependency is present between channels, less delay slots are needed. When more than six channels are present the pipeline latency is completely masked requiring no additional delay slots.

B. ISA

The principal distinctive feature of this architecture is the restricted Instruction Set Architecture (ISA), which plays a fundamental role, along with processor architecture and implementation in achieving the goal of a completely deterministic and consistent execution time. A careful design of the allowed operations, can ensure bounded execution time, while not impeding software development for the targeted application. The main manifestation of this philosophy is the lack of ISA support for branching or procedure call, ensuring a linear, predictable and constant flow of execution. Another side benefit of this limitation is the enablement of pure instruction counting as an execution time measurement technique, without having to make any assumptions on inputs or state of memory. Allowing the toolchain, once clock frequencies and deadlines have been evaluated, to detect and report potential overruns at compile time. The unified memory structure also eliminate the need for most data handling operations, as the whole memory pool can be directly accessed by the execution units, leaving the load constant as the only operation in this class.

From a physical perspective all instructions have a very similar structure with a 5-bit opcode followed by a series of optional 6-bit arguments representing the addresses of operands and destination. In particular four different structures, shown in Fig. 4, are used.

- **Independent instructions**: This structure is used for a varied class of instructions mainly needed to control the execution flow of the program. The instructions are composed by the opcode only, with the remaining bits zeroed.
- **Load Constant**: This structure, used for the load constant instruction only, here the opcode is followed by a destination address, with all other bits zeroed. The constant to load needs to be placed as a complete 32-bit word after this instruction, interleaving it with the instruction stream.
- **Unary instructions**: This structure is used for instruction that act on a single operand, and is used for conversion between float and integer number formats. For these the opcode is followed by operand and result destination addresses.
- **Binary instructions**: This structure is used for arithmetic, logic and comparison instructions that act on two operands and return a result. Here the opcode is followed by the two operand and destination addresses.

C. TOOLCHAIN

Complementing the femtoCore processor, a software/-firmware development tool-chain has been developed, which greatly simplifies the task of translating the desired control techniques to executable machine code. The first of its components is an Assembler, which can be used to extract the most performance out of the architecture, through hand written assembly code. Few high level features have been implemented simplify the development, such as named variable, to avoid having to directly allocate registers or automatic floating point constant conversion. The second component of the tool-chain is a higher level compiler, aimed at a more general developer audience, that can be used with a strict subset of the C language, supporting all possible features given the limitations of the hardware. While a debugger is also usually provided, with all general purpose compilers, to allow single stepping through the code in order to verify its behaviour, its usefulness for the application targeted by this paper, as halting execution of a hard real-time control system will lead to hardware faults or even damage. In its place, an emulator developed in MATLAB is supplied instead, that reads and executes the same machine instructions used in the real hardware, allowing single step debugging through the code. Last but not least HDL simulation can be used, to verify the cycle accurate behaviour of the core when integrated in a larger system. To aid in this process the tool-chain can directly produce verilog memory initialization files to pre-populate the instruction memory, avoiding the need to use a dedicated AXI Bus Functional Model (BFM) to emulate the ARM processor to femtocore interface

IV. COMPARATIVE ANALYSIS

The first, in Fig. 6a, is the use of separate FPGA and processor, on the same circuit board connected together through the processor external bus interface. This solution is typically the least flexible and slowest of the four, the external interface severely limits the maximum frequency of the communication, the synchronisation of the two elements can also be challenging when bidirectional information flow is required. Due to these limitations, when this solution is adopted, the programmable logic device is only used to implement the final modulation. Consequently the control tasks are performed completely by the processor. As such the advantages and disadvantages for this solution are the same as for the traditional single MCU implementation. The second solution, in Fig. 6c, consists of integrating the processor directly inside the FPGA using a softcore IP, these are available both from FPGA vendors (Xilinx Microblaze and Intel Nios II) or third parties (ARM cortex M1 and M3). The integration of both components on the same die allows much higher bandwidth between the two components and makes synchronisation of the different parts of the design much simpler, allowing some degree of acceleration of computationally intensive tasks by the FPGA. The third architecture, in Fig. 6b, replaces the softcore processor with a hard macro processor, realised in silicon on the same die of the FPGA, and connects them with a communication bus (typically AMBA AXI). This solution allows the use of a much more performing processor core, like the Cortex A9 running at several hundred megahertz on the Zynq. This change comes however with some stark disadvantages. While the softcore processor can be configured to minimize the amount of execution time jitter, by removing caches, disabling branch prediction, where present and so on; the hard processor system on the SoCs is optimized for raw throughput, as opposed to latency or low jitter, forcing the adoption of large timing margins to compensate, negatively affecting the achievable performance. The last Architecture, the one proposed in this paper, shown in Fig. 6d is closely related to the second one, where the general purpose softcore processor is replaced by the femtoCore processor. The main advantage of this arrangement is easy synchronization, since the core execution can be started with an external signal and the run-time is constant.

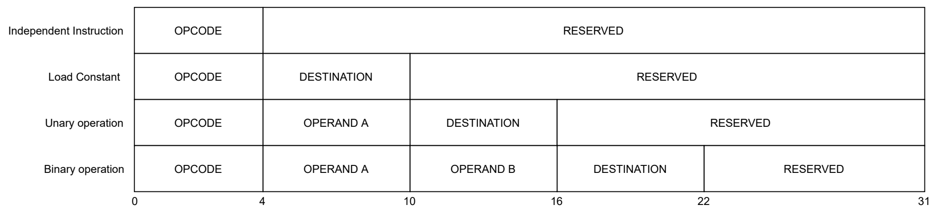


FIGURE 4. Physical instruction structures.

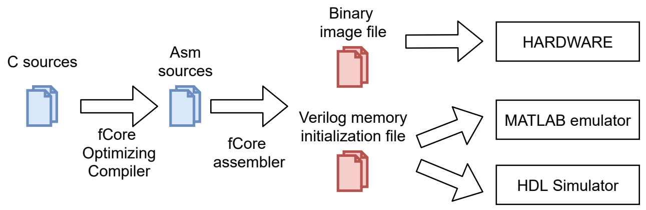


FIGURE 5. Structure of the femtocore tool-chain

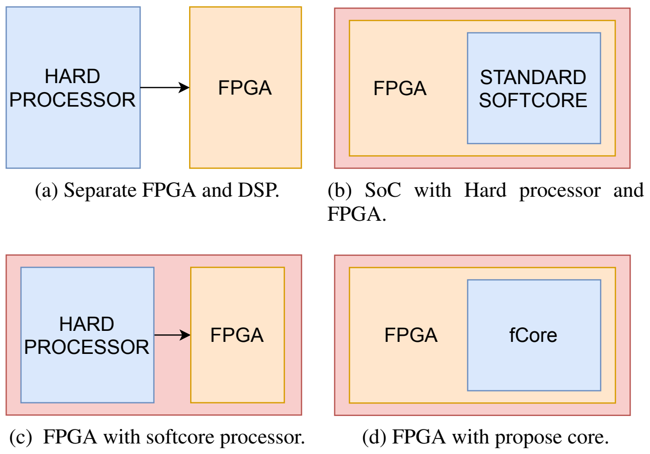

HARD PROCESSOR
FPGA
(a) Separate FPGA and DSP.
FPGA
STANDARD SOFTCORE
(b) SoC with Hard processor and FPGA.
HARD PROCESSOR
FPGA
(c) FPGA with softcore processor.
FPGA
fCore
(d) FPGA with propose core.

FIGURE 6. Architecture considered in the comparison.

In table 2 it is shown a more detailed qualitative comparison of few key features and metrics between the four solutions. The + symbol denotes a strength of the architecture, a − signifies a weakness and an = shows where the solution is average. In the clock frequency category the hard processor shows its definite advantage. Both softcore and the proposed solution are average and the separate IC topology is considered weak, as in this last architecture, the separate components might be able separately to run at high frequencies, however the communication bottleneck between them limits the overall system effectiveness. As shown in the previous section, only the proposed solution can claim a truly deterministic execution time, for the control algorithm that can be known in advance. In the category of programming complexity solution, (C) is weak, as the processors found in these type of systems are designed with compatibility with modern operating systems in mind, making them a lot more complex to use with respect to all other architectures that use microcontroller architectures specifically designed to be used in a bare metal context. The proposed solution on the other hand can be easily developed for since, for the targeted application, only a relatively short sequence of mathematical operations are required, as all other I/O and timing tasks are carried out by the custom logic outside of the core. When looking at hardware complexity, we see solution (A) and (B) having a clear advantage with respect to the remaining two as solution (C) requires dealing with the data exchange between hard core and FPGA portion, while the proposed solution requires external logic to handle sensor sampling and data movement.

	A	B	C	D
Clock frequency	−	=	+	=
Determinism	−	−	−	+
Programming complexity	=	=	−	+
Hardware complexity	+	+	−	−

TABLE 2. Metrics Comparison between the solutions

V. CASE STUDY

A. SYSTEM DESCRIPTION

In order to validate the claims, a case study has been set up, in order to compare the proposed architecture with a more traditional one. The target application is a multi-phase drive, the target motor, a permanent magnet synchronous machine, is composed by two set of threephase winding, radially offset by a 30° electrical angle. The neutral points of the two stars can be connected together or kept separate, depending on whether galvanic isolation or better performance under fault are prioritised.

With regards to power electronics and control, two different systems have been used, the first, implemented following the proposed architecture, uses a six phase distributed design, where single phase cells, composed of a half bridge inverter sub-section, control each winding, with a digital communication network connecting them to the central controller, as shown in Fig. 7. The second system used in this comparison follows a more traditional architecture where the two three-phase sets of the machine are kept completely separate, neutral connection included, and driven by two identical monolithic three-phase diode clamped NPC inverters. The control system is implemented with a single Texas Instruments (TMS320F377D) Microcontroller/DSP that takes care off everything, from the analog to digital conversion of the

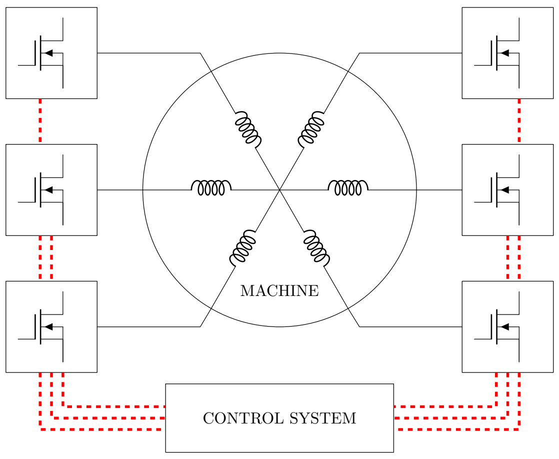


FIGURE 7. Diagram of the hardware and control system configuration used in the case study with the proposed architecture.

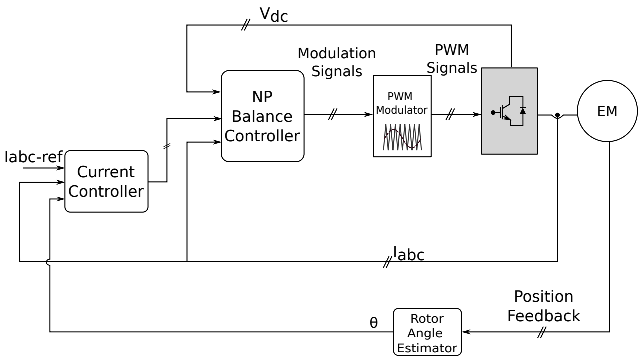


FIGURE 8. Diagram of the hardware and control system configuration used in the case study with the traditional architecture.

sensor signals to the PWM output generation. The use of two different power electronics platforms was necessary, given the incompatibility of the interfaces between them and the relative control logic boards. To make the comparison as valid as possible, the same current control technique has been implemented, following the exact same structure using floating point math, on both the traditional and the proposed architecture. From an hardware control perspective, the only difference between the two systems is the presence of a neutral point controller in the NPC inverter control system. The execution time used in this task was not included in the measurements. All other differences, including Switching frequency, power devices topology, etc. are inconsequential to the result of the comparison, as a functionally equivalent portion of code executed by the two processors (femtoCore and TI DSP), and only a single switching cycle is considered.

The main parameters for both systems are shown in table 3. The overall goal of the system is to have a completely fault tolerant actuation solution, where upon one or more phase faults, the system can react, isolating the damaged part and reconfigure itself, either statically or at run-time, to run as a lower phase count system. This will allow, in mission critical applications, to keep the potentially damaged actuator running, once appropriate de-rating factors are applied, for long enough to reach a safe global system condition, where the actuator can be safely excluded from operation, until serviced.

Parameter	Value
Switching/sampling frequency (proposed architecture)	60 kHz
Switching/sampling frequency (Traditional architecture)	15 kHz
DC-Link voltage	270 V
Peak output current	20 A
machine pole pairs	4
machine speed	1000 rpm

TABLE 3. Main system parameters

While the power electronics hardware for the two systems is different, the exact same control strategy is used in both cases, making the comparison of the two implementations fair. For fault tolerance reasons it has been decided against the use of a more traditional field oriented control, using Vector Space Decomposition (VSD) and rotating reference techniques, due to potential instability in the fault mode transition.

Instead, a Proportional Integral Resonant (PIR) approach is used to directly control each phase current in a static reference frame, where the controllers, can track sinusoidal signals, thanks to the resonant elements, that gives the closed loop system, potentially infinite, gain at the operating frequency.

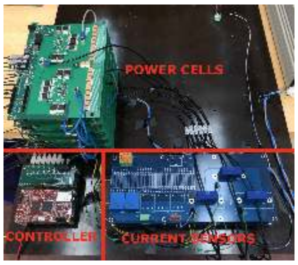
To test the performance of the system current control, a standard testing rig, shown in figure 9 has been used, with the controlled motor mechanically coupled to a second machine that is used to load the first one. During the test the load was controlled by a commercial drive to keep a constant rotational speed, while the test machine, generated an constant torque opposing the rotation.

An example of output current quality resulting from the aforementioned control can be seen in figure 10. The currents, after being sensed by six LEM LA55P-SP1 closed loop hall effect transducers, have been simultaneously sampled by six independent 14-bit analog to digital converters (LTC2313-14), each one running at 240 kSps. The resulting data as captured has been saved and plotted without any further post-processing step.

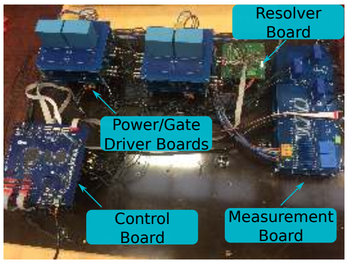
A quantitative comparison between the two solutions has been performed in order to show the relative performance gains attainable with the proposed architectures.

B. PROPOSED ARCHITECTURE TIMING ANALYSIS

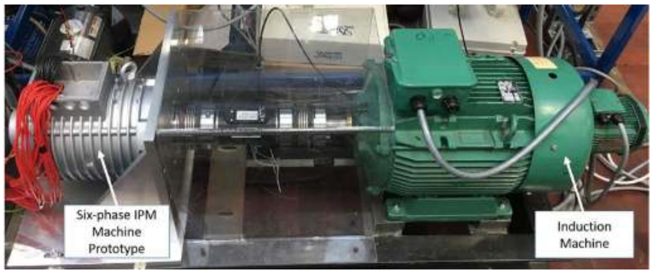
Since the proposed architecture behaviour is fully deterministic, its performance can be evaluated through a cycle accurate simulation of the system, including the control layer, communications and the power cell control logic. This route



(a) Proposed architecture experimental setup.



(b) Traditional architecture experimental setup.



(c) Controlled machine.

FIGURE 9. Pictures of the experimental setups.

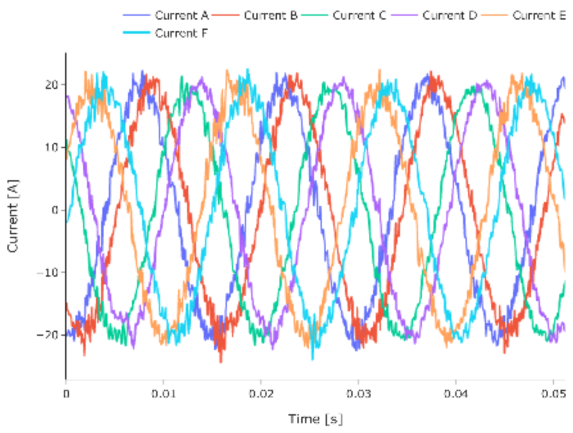


FIGURE 10. Experimental Current Waveforms.

has been chosen as opposed to runtime measurement since it can provide visibility of the whole system state without any additional performance penalty, as opposed to the introduction of instrumentation points for dynamic analysis. To perform the simulation, the FPGA vendor toolchain (Xilinx in this instance), has been used, hooking the top level module of the design with a set of Functional models that emulate in cycle accurate fashion the external sensing components, and run at a clock frequency of 100 MHz.

The latency from ADC sampling to PWM modulators control register update is 5.81 µs,supporting, on suitable hardware, a switching frequency of 172 kHz, of which, 710 ns constitute the pre-calculation time, from when the sampling command is issued to when the femtoCore starts running, the complete execution time is 3.31 µs, 520 ns per channel.

Delay type		Traditional	Proposed
pre-execution		128	79
execution	total	608	331
	per phase	150	55

TABLE 4. Comparison of the number of clock cycles needed for the execution of the current control with the traditional and proposed architecture.

Lastly 1.79 µs is the post execution time, that covers from the conclusion of the execution to the update of the modulators, whose most important factor is the communication transmission latency.

C. TRADITIONAL ARCHITECTURE TIMING ANALYSIS

The evaluation of the performance of the proposed traditional architecture, unlike the proposed one must be carried out on hardware at runtime, as neither simulation nor instruction counting can fully capture the complex dynamics of a modern processor system. To capture timing information with the lowest impact possible several runs are made each one measuring the duration of a different portion of interest in the code. In particular a single GPIO pin is toggled at the start of a task and then again upon completion. An oscilloscope is used to measure the duration of the resulting pulse.

For a core clock frequency of 200 MHz, the total end-to-end delay in this architecture is 3.64 µs, of which 640 ns are of pre-execution time, comprising the ADC measurement; while the remaining portion comprises the execution of two controllers for each three phase winding, as the third phase reference is derived as the algebraic sum of the other two.

D. TIMING COMPARISON

For the comparison between the two architectures to be completely fair, few factors need to be accounted for, first and foremost is the difference in the clock frequency the two systems are run at. In order to compensate for this, the duration of the various phases is shown in amount of clock cycles instead of absolute time. Also the traditional architecture controls only two phases for each set, four in total, with the third reference being generated as the algebraic sum of the other two in order to obtain a balanced set, while on the proposed architecture all six controllers are run. To compensate for this difference the execution time is shown in number of clock cycles per phase, as well as globally. The results of the comparison are shown in Tab. V-D, where it is clearly shown that the proposed architecture is able to execute an equivalent control task in roughly a third of the time with respect to a regular controller, while retaining the benefits of a programmed system. When comparing the pre-execution times, the gain of moving to a completely custom logic solution are more limited, as its lower bound is given by the ADC used.

VI. CONCLUSION

In this paper an implementation architecture for high performance real-time control systems is proposed, constituting an intermediate step between both full HDL and software based implementations, retaining the determinism and execution time consistency of the first, while adopting the faster and more accessible software development paradigm. A novel custom designed floating point processor is proposed. The completely deterministic execution time, and specifically designed Instruction Set Architecture allows an effortless translation of even complex algorithms with minimal loss of precision. Automatic parallelization through SIMD execution further simplifies software development for multi-phase systems. The proposed system has also been experimentally compared to a traditional MCU based one, showing a much better efficiency, implementing the same control technique in just a third of the number of cycles.

REFERENCES

[1] H. Schefer, L. Fauth, T. H. Kopp, R. Mallwitz, J. Friebe, and M. Kurrat, “Discussion on electric power supply systems for all electric aircraft,” IEEE Access, vol. 8, pp. 84 188–84 216, 2020.

[2] A. K. Morya, M. C. Gardner, B. Anvari, L. Liu, A. G. Yepes, J. Doval-Gandoy, and H. A. Toliyat, “Wide bandgap devices in ac electric drives: Opportunities and challenges,” IEEE Transactions on Transportation Electrification, vol. 5, no. 1, pp. 3–20, 2019.

[3] S. Vazquez, J. I. Leon, L. G. Franquelo, J. Rodriguez, H. A. Young, A. Marquez, and P. Zanchetta, “Model predictive control: A review of its applications in power electronics,” IEEE Industrial Electronics Magazine, vol. 8, no. 1, pp. 16–31, 2014.

[4] S. Vazquez, J. Rodriguez, M. Rivera, L. G. Franquelo, and M. Norambuena, “Model predictive control for power converters and drives: Advances and trends,” IEEE Transactions on Industrial Electronics, vol. 64, no. 2, pp. 935–947, 2017.

[5] N. Vedachalam and G. A. Ramadass, “Reliability assessment of multi-megawatt capacity offshore dynamic positioning systems,” Applied Ocean Research, vol. 63, pp. 251–261, 2017.

[6] J. W. Bennett, G. J. Atkinson, B. C. Mecrow, and D. J. Atkinson, “Fault-tolerant design considerations and control strategies for aerospace drives,” IEEE Transactions on Industrial Electronics, vol. 59, no. 5, pp. 2049–2058, May 2012.

[7] F. Savi, D. Barater, M. D. Nardo, M. Degano, C. Gerada, P. Wheeler, and G. Buticchi, “High-speed electric drives: A step towards system design,” IEEE Open Journal of the Industrial Electronics Society, vol. 1, pp. 10–21, 2020.

[8] J. Harikumaran, G. Buticchi, G. Migliazza, V. Madonna, P. Giangrande, A. Costabeber, P. Wheeler, and M. Galea, “Failure modes and reliability oriented system design for aerospace power electronic converters,” IEEE Open Journal of the Industrial Electronics Society, vol. 2, pp. 53–64, 2021.

[9] B. A. Welchko, T. A. Lipo, T. M. Jahns, and S. E. Schulz, “Fault tolerant three-phase ac motor drive topologies: a comparison of features, cost, and limitations,” IEEE Transactions on Power Electronics, vol. 19, no. 4, pp. 1108–1116, 2004.

[10] L. Costa, G. Buticchi, and M. Liserre, “A fault-tolerant series-resonant dc–dc converter,” IEEE Transactions on Power Electronics, vol. 32, no. 2, pp. 900–905, 2017.

[11] G. Migliazza, G. Buticchi, E. Carfagna, E. Lorenzani, V. Madonna, P. Giangrande, and M. Galea, “Dc current control for a single-stage current source inverter in motor drive application,” IEEE Transactions on Power Electronics, vol. 36, no. 3, pp. 3367–3376, 2021.

[12] M. Westmijze, M. J. G. Bekooij, G. J. M. Smit, and M. Schrijver, “Evaluation of scheduling heuristics for jitter reduction of real-time streaming applications on multi-core general purpose hardware,” in 2011 9th IEEE Symposium on Embedded Systems for Real-Time Multimedia, 2011, pp. 140–146.

[13] H. Kim, A. Kandhalu, and R. Rajkumar, “A coordinated approach for practical os-level cache management in multi-core real-time systems,” in 2013 25th Euromicro Conference on Real-Time Systems, 2013, pp. 80–89.

[14] P. De, V. Mann, and U. Mittaly, “Handling os jitter on multicore multithreaded systems,” in 2009 IEEE International Symposium on Parallel Distributed Processing, 2009, pp. 1–12.

[15] F. Sobrino-Manzanares and A. Garrigos, “Bidirectional, interleaved, multiphase, multidevice, soft-switching, fpga-controlled, buck–boost converter with pwm real-time reconfiguration,” IEEE Transactions on Power Electronics, vol. 33, no. 11, pp. 9710–9721, 2018.

[16] M. Sinha, J. Poon, B. B. Johnson, M. Rodriguez, and S. V. Dhople, “Decentralized interleaving of parallel-connected buck converters,” IEEE Transactions on Power Electronics, vol. 34, no. 5, pp. 4993–5006, 2019.

[17] J. Zhou, Y. Xu, H. Sun, Y. Li, and M. Chow, “Distributed power management for networked ac–dc microgrids with unbalanced microgrids,” IEEE Transactions on Industrial Informatics, vol. 16, no. 3, pp. 1655–1667, 2020.

[18] I. Hussain, R. K. Agarwal, and B. Singh, “Mlp control algorithm for adaptable dual-mode single-stage solar pv system tied to three-phase voltage-weak distribution grid,” IEEE Transactions on Industrial Informatics, vol. 14, no. 6, pp. 2530–2538, 2018.

[19] M. Rivera, J. Rodriguez, J. R. Espinoza, and H. Abu-Rub, “Instantaneous reactive power minimization and current control for an indirect matrix converter under a distorted ac supply,” IEEE Transactions on Industrial Informatics, vol. 8, no. 3, pp. 482–490, 2012.

[20] J.-M. Muller, N. Brisebarre, and S. Raina, “Accelerating Correctly Rounded Floating-PointDivision when the Divisor is Known in Advance,” IEEE Transactions on Computers, vol. 53, no. 8, pp. 1069– 1072, 2004. [Online]. Available: https://hal-ens-lyon.archives-ouvertes.fr/ensl-00087465

APPENDIX. RESONANT CONTROLLER C CODE

```
const float damping = 0.005;
const float Kr = 800;
const float Ki = 0.5;
const float Kp = 0.8;
const float Ts = 1/60e3;
const float sat_max = 135.0;
const float sat_min = -135.0;

float Ki_out = 0;
float fwd_integ = 0;
float back_integ = 0;
float fwd_in = 0;

float sat(float in, float ub, float lb){
  //N.B. The femtocore compiler will
  //recognise this pattern and
  //emit saturation instructions.
  if(in > ub)
    return ub;
  else if( in < lb)
    return lb;
  else
    return in;
}

float pir(float error, float omega){

  /* calculate proportional action*/
  float Kp_out = Kp*error;
  float pir_out = Kp_out;

  /* calculate integral action*/
  Ki_out += Ts*Ki*error;
  sat(Ki_out, sat_max, sat_min);
  pir_out += Ki_out;

  /* calculate resonant action */
  fwd_integ += Ts*fwd_in*omega;
  sat(fwd_integ, sat_max, sat_min);
  back_integ += Ts*fwd_integ*omega;
  sat(back_integ, sat_max, sat_min);
  float s_error = (Kr*error)-fwd_integ;
  fwd_in = (s_error*damping)-back_integ;
  pir_out += fwd_integ;
  sat(pir_out, sat_max, sat_min);
```

```
    return pir_out;
}
```

FILIPPO SAVI received the M.Sc. degree in Electronic Engineering from the University of Parma, in 2017, he is currently pursuing the Ph.D. degree at University of Nottingham.

His research interests include High speed drives, modular and fault tolerant power electronics for transportation electrification. Digital communications for high performance, low latency and jitter for high bandwidth feedback control.

JAYAKRISHNAN HARIKUMARAN received the B.Tech degree in electronics and communication engineering from the National Institute of Technology Calicut, Kerala, India in 2008 and M.S. degree in sustainable energy technology from Delft University of Technology, Delft, Netherlands in 2012.

He has worked in Texas Instruments ('08-'10), Tvilight B.V ('12-'13) and Shell International B.V ('13-'18) on various roles - semiconductor design, embedded systems engineering, industrial control systems and electrical engineering. Since 2018, he is a Marie-Curie doctoral researcher at the Institute for Aerospace Technology, University of Nottingham, Nottingham, United Kingdom. His research interests include design for reliability of power converters, fault tolerant drive systems and digital controller implementation for power converters in DSP and FPGA.

DAVIDE BARATER (S'11-M'14) received the Master's degree in Electronic Engineering in 2009 and the Ph.D. degree in Information Technology in 2014 from the University of Parma Italy. He was an honorary scholar at the University of Nottingham, U.K., during 2012, and a visiting researcher at the University of Kiel, DE in 2015. He is currently Associate Professor at Department of Engineering "Enzo Ferrari", University of Modena and Reggio Emilia, Italy. His research area is focused on power electronics for e-mobility and motor drives. He is the Coordinator of two European Project: RAISE, to evaluate the impact of the high voltage gradients, introduced by the fast commutations of new wide bandgap power devices (SiC, GaN), on the life time of electrical motor insulation systems. AUTO-MEA that aims to develop electrical motors and drives for next generation of electrical mobility. In particular, novel solutions for windings structures and cooling systems for improved power density, efficiency and increased frequency operation. He is Associate Editor of IEEE Transactions on Industry Applications and author or co-author of more than 60 international papers.

GIAMPAOLO BUTICCHI (S'10-M'13-SM'17) received the Master degree in Electronic Engineering in 2009 and the Ph.D degree in Information Technologies in 2013 from the University of Parma, Italy. In 2012 he was visiting researcher at The University of Nottingham, UK. Between 2014 and 2017, he was a post-doctoral researcher and Von Humboldt Post-doctoral Fellow at the University of Kiel, Germany.

He is now Professor in Electrical Engineering at The University of Nottingham Ningbo China and the Head of Power Electronics of the Nottingham Electrification Center. His research focuses on power electronics for renewable energy systems, smart transformer fed micro-grids and dc grids for the More Electric Aircraft. He is author/co-author of more than 200 scientific papers and an Associate Editor of the IEEE Transactions on Industrial Electronics and of the IEEE Transactions on Transportation Electrification.

He is the Chair of the IEEE Industrial Electronics Society Technical Committee on Renewable Energy Systems.

CHRIS GERADA received a PhD degree in numerical modelling of electrical machines from the University of Nottingham, Nottingham, UK, in 2005. He subsequently worked as a researcher at the University of Nottingham on high-performance electrical drives and on the design and modelling of electromagnetic actuators for aerospace applications. He was appointed Lecturer in electrical machines in 2008, Associate Professor in 2011, and Professor in 2013. His core research interests include the design and modelling of high-performance electric drives and machines. Prof. Gerada is an Associate Editor of the IEEE Transaction on Industry Applications. He has secured major industrial, European and UK grants, authored more than 200 papers and has been awarded a Royal Academy of Engineering Research Chair to consolidate research in the field.

PAT WHEELER received his BEng [Hons] degree in 1990 from the University of Bristol, UK. He received his PhD degree in Electrical Engineering for his work on Matrix Converters from the University of Bristol, UK in 1994. In 1993 he moved to the University of Nottingham and worked as a research assistant in the Department of Electrical and Electronic Engineering. In 1996 he became a Lecturer in the Power Electronics, Machines and Control Group at the University of Nottingham, UK. Since January 2008 he has been a Full Professor in the same research group. He was Head of the Department of Electrical and Electronic Engineering at the University of Nottingham from 2015 to 2018. He is currently the Head of the Power Electronics, Machines and Control Research Group and is the Li Dak Sum Chair Professor in Electrical and Aerospace Engineering at the University of Nottingham, China. He is a member of the IEEE PELs AdCom and was an IEEE PELs Distinguished Lecturer from 2013 to 2017. He has published 500 academic publications in leading international conferences and journals.