School of Information Technology and Engineering at the ADA University

School of Engineering and Applied Science at the George Washington University

DESIGN AND ANALYSIS OF A FPGA-BASED CONTROLLER FOR GAN-BASED THREE-PHASE INVERTERS: EFFICIENCY, PERFORMANCE, AND RELIABILITY OPTIMIZATION

A Thesis
Presented to the Graduate Program of Electrical and Power Engineering
of the School of Information Technology and Engineering
ADA University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in Electrical and Power Engineering
ADA University

By
Mujgan Huseynli

December, 2024

THESIS ACCEPTANCE

This Thesis by: Mujgan Huseynli
Entitled: *Design and Analysis of a FPGA-Based Controller for GaN-Based Three-Phase Inverters: Efficiency, Performance, and Reliability Optimization*

has been approved as meeting the requirement for the Degree of Master of Science in Electrical and Power Engineering of the School of Information Technology and Engineering, ADA University.

Approved:

(Adviser)	(Date)
(Program Director)	(Date)
(Dean)	(Date)

ABSTRACT

The main objectives of the research thesis are the design, simulation, and analysis of a FPGA controlled GaN based 3-phase grid-tied inverters in terms of efficiency and performance. High frequency switching applications provide significant challenges for conventional inverters based on MCUs due to their slow processing speeds, sequential execution, and increased sensitivity to EMI. Despite their small size, efficiency, and need for precise control, inverters based on GaN are fast. FPGA is proposed in this work as they can execute computations quickly, adapt to unique hardware needs, and handle operations in parallel.

In order to analyze functionality of both FPGA and MCU, we employ FIL and PIL simulations. Traditional MCUs have difficulties with high-frequency switching in PIL simulations because of their high resource usage and long execution durations. On the other hand, based on FIL simulations, FPGA is one of the best options for real-time control. System performance can be improved by smart resource allocation. The study recommends sampling at rates higher than the switching frequency to avoid distorted waveforms, erroneous control signals, and inferior performance. FPGA-based waveform quality and PWM signal precision improve as switching frequency increases. Finally, to enhance waveform quality and system efficiency, research could explore advanced modulation techniques such as Space Vector Modulation (SVM) or Model Predictive Control (MPC).

The subsequent phase of this research may involve transitioning from simulation to empirical testing; this would need the assembly of the circuit using genuine GaN devices, drivers, and passive components, followed by an assessment of the system's performance in a practical environment. Integrating AI & ML methods for predictive control and real-time optimization of inverter performance might also be a good option, and it might enhance not only system efficiency but also stability. This research will help to demonstrate the potential of FPGA for high-frequency GaN inverters, by proving reliable and efficient aspects of power electronics and industrial applications.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviation	Explanation
ADC	Analog/Digital Converter
BC	Boost Converter
CPU	Central Processing Unit
DSP	Digital Signal Processing
EMI	Electromagnet Interference
FIL	FPGA in the Loop
FPGA	Field Programmable Gate Array
GaN	Gallium Nitride
IGBT	Insulated Gate Bipolar Transistor
IMMD	Integrated Modular Motor Drive
IPV	Photovoltaic Array Current
LCL	Inductor–Capacitor–Inductor (Filter Topology)
MCU	Micro Controller Unit
MPPT	Maximum Power Point Tracking
MPC	Model Predictive Control
MOSFET	Metal Oxide Semi-conductor Field Effect Transistor
PIL	Processor in the Loop
PLL	Phase Locked Loop
PMU	Power Management Unit (if used in your text)
P&O	Perturb and Observe (MPPT Algorithm)
PV	Photovoltaic
PWM	Pulse-Width-Modulation
RMS	Root-Mean Square
SiC	Silicon-Carbide

SVM	Space Vector Modulation
SVPWM	Space Vector Pulse Width Modulation
THD	Total-Harmonic Distortion
VSI	Voltage-Source Inverter

CHAPTER ONE
INTRODUCTION

Problem Statement

The main thesis goals are enhancing not only the efficiency but also the performance of GaN-based 3-phase inverters controlled by FPGAs. GaN technology is advantageous because of its ability of reducing power losses, accelerating switching speeds, and making compact/small designs. Current MCU-controlled systems are inadequately equipped to fully utilize GaN devices because of their restricted computing capabilities and vulnerability to EMI. Mitigating these restrictions is essential due to the increasing demand for achieving high-performance power conversion systems in applications such as renewable integration, EVs, and industrial motors. This study seeks to enhance the efficiency and reliability of power electronics systems by utilizing the parallel processing and real-time features of FPGAs.

The advancement of power electronics has transformed electrical systems, particularly renewable-powered sustainable energy integration, electric battery-powered vehicles, and industrial-grade motor drives. A key component of improved and efficient power distribution under different loads is the three-phase inverter for DC-AC conversion. Traditional inverters have limited use in power conversion systems due to their unreliability, low performance, and inefficiency [1].

Traditional inverters have limitations due to their reliance on Si-based semiconductors such as IGBTs and MOSFETs. The switching and conduction losses in these devices lead to decreased efficiency and elevated thermal stress [2-3]. High switching losses cause thermal management systems to be inefficient and expensive. Higher frequencies are required for greater power density and fewer filter components, but silicon devices can only function at moderate switching speeds [4].

It is suggested that enhanced parasitic capacitances, switching ratios, resistances, and breakdown voltages in GaN-based semiconductors could address these challenges. They may exhibit enhanced efficiency, less thermal losses, and accelerated switching speeds [5]. Reduced passive electronics size and increased frequencies lead to more compact and lightweight inverter designs. Using this type of device is an outstanding option for applications that need a lot of efficacy and powder density, according to recent studies [6].

While there are numerous benefits to GaN-based inverters, they are often limited by traditional control hardware, including MCUs. While MCU-systems are ideal for controlling inverters due to their low cost and ease of use, they aren't capable of handling high-frequency applications. This is because high-frequency applications necessitate precise control, which is not possible with MCUs due to their sequential processing architecture. High-speed/frequency GaN devices also have latency problems, which make them incompatible in different situations. On the other hand, they do not have enough processing power to apply modern/cutting-edge control schemes and high-resolution PWM signals needed to advance their performance. Furthermore, this type of inverters struggles with insufficient processing capability and EMI sensitivity, making it challenging for them to maintain stability, react quickly enough to commands, and fully utilize GaN technology potential. Therefore, they cannot utilize the high performance and efficiency that GaN technology offers.

One effective alternative to MCUs for managing GaN-based inverters is FPGAs. They provide several benefits, including faster processing times, more flexible hardware, and the capacity to

work in parallel. Due to their parallel design, FPGAs are able to execute many control loops simultaneously, allowing for the on-time execution of complicated algorithms in real-time. Consequently, FPGAs are especially efficient for high-frequency switching applications because to their necessity for accurate timing. In addition to its computational speed, FPGAs provide versatility in creating tailored hardware configurations for certain control requirements. This enables developers to create efficient controllers that fully exploit the high-speed switching capabilities of devices utilizing GaN technology. Because of the reduced susceptibility of its architecture to EMI, FPGAs deliver reliable performance even under challenging electrical settings [11, 12]. Consequently, they may harness the potential of GaN technology by advancing the efficiency, dynamic responsiveness, and reliability of inverter systems.

With all these advantages listed above, it is clear that FPGA optimized GaN-based 3-phase inverters should be developed and studied. With these controllers, power conversion systems may fully use GaN devices while addressing the issues with MCUs.

Definition of Terms

- **Three-Phase Inverter** – A device converts the direct current (DC) into three phase alternating current (AC).

- **Gallium Nitride** - A wide bandgap technology known for high switching velocity, high efficiency, and lower conductive losses.

- **Processor-in-the-Loop (PIL)** - A real-time testing methodology in which the algorithm for the control runs on a physical microcontroller or digital signal processing unit while the plant (inverter, motor, or power system) is simulated in software. It enables the hardware-level verification of the algorithm under non-ideal operating conditions, without the danger of causing damage to physical power components.

- **FPGA-in-the-Loop (FIL)** - A real-time testing methodology in which the control algorithm or specific logic blocks are deployed and executed on an actual FPGA board. This is especially beneficial for high-frequency applications or complicated control tasks, as FPGA has the ability of parallelism and faster speed.

Significance of the Study

To overcome the limitations of MCU-controlled systems for GaN-based inverters, this study evaluates FPGA possibilities. We use the Xilinx Artix-7 XC7A100T FPGA to test its parallel-processing, high-speed computing, and hardware flexibility. PIL and FIL simulations with the TI TMS320F28379D and Artix-7 FPGA were done in MATLAB/Simulink. These methods allow THD, waveform integrity, PWM signal correctness, resource efficiency, and execution duration to be examined.

This is crucial for enhancing high-performance power conversion systems essential for renewable integration, EVs, and industrial motors. We addressed difficulties including attaining accurate control at faster switching frequencies and optimizing hardware efficiency, hence contributing to the advancement of scalable, next-generation power electronics solutions. The results seek to guide the design and execution of more efficient and compact power systems that correspond with the growing need for sustainable and high-performance energy solutions.

Integrating FPGAs with GaN technology offers practical improvements in power electronics and supports various applications. This type of inverters can operate at fast switching frequencies with low conductive losses. Therefore, it leads to power conversion efficiency enhancement. Furthermore, FPGAs also make it possible to control switching operations and further increase efficiency precisely. For example, a study showed that a 3-phase inverter with GaN technology reaches 98.5% efficiency at 400 V AC with a 60% load [13].

Lighter and more compact systems are possible with GaN devices because of their higher frequency operation, which enables the fabrication of smaller passive components. FPGAs handle complex control algorithms to support these high-frequency operations. When these two factors work together, inverter designs become more compact and lighter, which is ideal for applications where these two metrics are critical. Reduced heat emission is a result of GaN devices reduced conduction and switching losses. The inverter can be kept at a stable/safe temperature and thermal parameters can be modified in real time with the help of FPGAs, which contribute to thermal management. As a result, the inverter system will be more robust and will have longer lifespan. It is also tested that FPGAs offer unparalleled versatility in complicated control techniques like as MPC and FOC[14].
Their ability to make computation in parallel makes the inverter better at responding to changing grid circumstances in real time. A 3-phase T-type NPC grid-tied inverter achieved an efficiency of 98.49% using one FPGA-based predictive control method. The load range was from 50% to 75% [15].

FPGAs combined with GaN technology offers grid support systems including reactive-power regulation, harmonic elimination, and fault mitigation. These goals become increasingly important as the use of renewable energy sources increases and helps to maintain the dependability and quality of the power system. GaN-based devices controlled by FPGAs [16] help to easily include renewable energy sources into the power grid. This achievement confirms the worldwide endeavor to progress renewable energy sources and lower greenhouse gas emissions.
Ultimately, 3-phase inverters built on GaN technology and running FPGA controllers have many advantages including a smaller installation size, improved thermal optimization, more control adaptability, and grid support. These technologies create the basis for strong and long-lasting electronics systems, so offering a range of further advantages.

Limitations of the Study

This project has some issues that needed to be considered to fully understand its scope and boundaries.

One issue was that the project lacked the parts needed to develop the GaN-based three-phase inverter system. Due to the inaccessibility of GaN transistors, drivers, and passive parts, only the FPGA and MCU could be used for hardware testing. Therefore, PIL and FIL simulations were employed to validate experiments. Controlled environments are good for testing control strategies and algorithms, but simulations cannot match system complexity and movement.

This project was also constrained by its slow FPGA and MCU. While not the most powerful hardware platforms on the market, these devices were sufficient for the evaluation and implementation of the suggested control techniques. In high-frequency switching and sophisticated control applications, improved field-programmable gate arrays (FPGAs) with faster working times, greater memory, and more input/output possibilities can improve results.

Lastly, we focused on one component of the project to make it more usable in real life and valid for future work. Temperature variations, mechanical vibrations, and long-term operational stress could impact the system's performance and trustworthiness, however the testing was done in a computer.

CHAPTER TWO
REVIEW OF THE LITERATURE

Inverters

An inverter, alternatively called a DC/AC converter, takes a direct current as an input and produces output in the form of an alternating current. As crucial elements in today's power electronics, inverters underpin a wide array of applications, from renewable powered sustainable energy systems to electrical battery powered vehicles and industrial grade motor drives. We can classify them into two broad categories which are either voltage-source or current-source inverters. They will be abbreviated as VSIs and CSIs, respectively. VSIs are usually connected to a source of DC voltage, and they are stabilized by using a large enough capacitor plugged on the bus of DC. Conversely, CSIs are plugged source or DC current source and accompanied with connecting serially a large inductor.

To manage voltage and frequency of output, inverters employ Pulse Width Modulation (PWM) tools. This approach involves switching the inverter's power devices in a manner that transforms a potentially variable amplitude signal into pulses of varying width, thereby achieving the desired AC characteristics. Widely used PWM strategies include sinusoidal and space vector PWM, abbreviated as SPWM and SVPWM, respectively. Both are famous for cutting down harmonic-distortion and increasing efficiency for the overall system [17-25].

According to [26], although the control strategies for VSIs and CSIs differ, both are indispensable for DC-AC conversion across multiple applications. CSIs are generally suited for inductive loads such as motors and transformers, delivering high quality AC. They can also adjust voltage and frequency using the current modulation. Their common applications are inverter-based HVAC, motor drives, and control based on electromagnet. In contrast, VSIs are usually employed for loads with capacitive nature —like electronic devices and systems for lighting —providing output voltage with good stability and allowing regulation of both voltage and frequency. These aspects make VSIs better suited for use cases like uninterruptible supplies of power (also shortened as UPS), inverters based on solar technology, and appliances for daily household.

In renewable source based sustainable energy systems, panels of solar and turbines of wind generate power as DC. It is a common practice to apply inverters to convert that into AC power to integrate with the power grid as well as consuming locally. Advanced tech for semiconductors, like SiC or GaN, have further contributed to the improvement of inverter efficiency and performance. For instance, [27] presents a 2-stage, one-phase microinverter design based on GaN connected to the grid that demonstrates effective MPPT and robust power continuity, even under converter faults. Similarly, [28] underscores that incorporating SiC and GaN devices into inverter designs yields higher efficiency while reducing both cooling and filtering needs. In particular, SiC devices exhibit consistent performance across varying operating conditions, whereas GaN devices excel at switching with higher frequencies. These improvements highlight the critical position of inverters in better reliability and optimized production of energy within renewable source based sustainable energy contexts.

In electric battery powered vehicles (EVs), three-phase inverters manage power transfer among motors and batteries. Multiple design approaches have emerged to boost efficiency, shrink system dimensions, and enhance thermal behavior. For example, research on SiC MOSFET inverters demonstrates the potential for better efficiency and switching at higher frequencies,

thereby increasing performance for overall EV traction [28]. In a related study, a hybrid inverter with GaN-Si T-Type three-level configuration successfully lowers losses related to switching and increases density for power —both essential attributes for systems within EVs [29]. Additionally, the application of a power inverter based on 9 switch Z-source with MCBC for EVs indicates a promising way to cut down count of components as well as losses associated with switching, which in turn leads to reduces costs while making it possible to control two motors independently [30]. Collectively, all of the above-mentioned technological strides are leading to more efficient as well as compact inverters with reduced cost to apply in the context of electric vehicles.

In industrial motor drives, inverters are essential, especially in applications that demand accurate speed and torque regulation. The literature examines several inverter topologies for their use in IMMD systems, emphasizing the significance of efficiency, fault tolerance, and size minimization. The paper [31] offers a thorough comparison of inverter topologies, including 2L-VSI and 3L-VSI, highlighting their enhanced efficiency when employing GaN power semiconductors in contrast to traditional IGBT motor drives. Reference [32] examines sources - CSI, highlighting their advantages, such as motor-compatible waveforms and power reversing ability, which are advantageous for medium-voltage drives. Their suggested improved IVC method for CSI drives guarantees enhanced isolation and field orientation, overcoming the shortcomings of traditional IVC systems. These studies highlight the advancing nature of the technology behind inverters in improving motor drive efficiency as well as reliability.

In conclusion, inverters are crucial for the efficient functioning of renewable energy systems. They facilitate the transition of direct current (DC) power from renewable sources into steady alternating current (AC) power for grid integration or localized use. Progress in inverter technology, especially the incorporation of wide-bandgap semiconductors such as SiC and GaN, is enhancing efficiency, performance, and functionality in multiple applications.

GaN based devices

Why GaN

Breakthroughs in material science often led to further advances in applied sciences. Following the introduction of transistors at Bell Labs around 1948, these devices quickly advanced along with the first-generation of semiconductors such as silicon and germanium elements. Technology for silicon, in particular, drove the industrial revolution 3.0 and remains as the top extensively consumed semiconductor today. The second generation for semiconductors introduced compound materials such as gallium arsenide (GaAs) for transistors where high speed and power are required. Nowadays, power electronics research mainly focuses on third-generations semiconductor-based transistors made from GaN & SiC, usually abbreviated as GaN & SiC, respectively [33, 34]. The mentioned wide-bandgap semiconductors transcend the physical constraints of silicon, allowing for more efficient switching. Leveraging wide-bandgap materials is thus viewed as a key pathway to enhancing power converter efficiency and power density [35-39]. Among these materials, GaN transistors offer an especially high figure of merit, positioning them well-suited for higher-frequency operations. Fabrication process for them demands advanced packaging solutions to minimize parasitic inductances, exemplified by the BGA - ball grid array from EPC, the package with quad flat no-lead from Navitas, and the package with GaNPX from GaN —all of which are packages for chip-scale surface mount optimized for applications with high frequency.

Furthermore, power devices built using GaN technology have been widely investigated as well as adopted in converters of power thanks to small Rdson - specific resistance for on state, capabilities for fast switching, and capacitance of junction with reduced value (Coss), and absence of reverse recovery [40, 41]. Above-mentioned properties enable frequencies for high switching where they have losses with minimal values, leading to lighter, more compact converters and greater overall efficiency. Prior research demonstrates the advantages of using power converters with GaN technology. Compared with the silicon devices, GaN shows conduction, driving losses and switching with reduced values as well as lower winding losses for transformers as they require lower total device charge to support soft switching transients [42]. They also enhance the performance of LLC as well as buck converters thanks to recovery with zero reverse and capacitance with low junction [43]. In addition, inverters based on GaN technology with single-phase T-type can significantly cut loss for power and size of heatsink relative to silicon (Si) as well as silicon carbide (SiC) equivalents [44].

Advantages and disadvantages of GaN technology

Devices using Gallium Nitride (GaN) offer a wider array of benefits as well as drawbacks for applications used in power electronics.

Pros

1. **Better performance and increased efficiency:**

- Devices based on GaN technology exhibit notably reduced losses and better efficacy values, especially at very high switching frequencies. Comparative assessments indicate that GaN is the go-to choice in use cases demanding better efficacy and increased density of power, as its higher movement of electrons, that can shorten times for switching and reduce capacitance for the output [45-47]. Although devices based on Silicon Carbide (SiC) are also highly popular—owing to excellent conductivity with thermal nature, robust handling of electric field as well as wider energy bandgap—they are most advantageous in scenarios requiring high-temperature operation, higher frequencies, increased ratings for power, and substantial voltages for drain-source [45, 46, 48].
- According to [49], better on-state resistance of GaN technology as well as fast capabilities for switching causes 17% decrease in total losses for power when comparing and contrasting with SiC-MOSFETs. They also observe a reduction value of 61% relative to Si-IGBTs.
- Additionally, integrating GaN HEMTs in inverter designs can cut semiconductor losses by up to 60% versus Si MOSFETs, boosting total system efficiency by about 3%. This efficiency gain also accommodates a 60% increase in power rating without further losses, lowering heatsink volume and total expenses for the system.

1. **Management for Thermal Characteristics**

- Although GaN technology features a thermal conductivity with lower performance than SiC, its higher efficiency and lower power losses mitigate thermal management requirements. This is particularly important for high-density power electronics systems with compact design that demands managing heat as a crucial as well as key consideration for the design.
- A heatsink volume reduction of over 30%—enabled by minimized power losses—translates into decreased system size as well as economical cost, thus making solutions

relying on GaN technology both well practical for frequency of high values as well as implementations for powerful systems [50].

3. Higher Switching Frequency:

- Devices based on GaN technology can work at substantially faster switching frequency than conventional silicon technologies. In some cases, GaN HEMTs can achieve switching frequencies up to 10 times greater than Si MOSFETs, allowing significant size reductions and passive part weights and, consequently, lowering both cost and overall system volume.
- In many applications of power electronics, for example inverters for solar panels as well as motor drives, higher switching frequencies contribute to better efficient conversion of power as well as performance with improved metrics.

As noted by [50], Figure 2.1 illustrates, total losses can be reduced by over 60% at 50 kHz and by even more at 200 kHz, while the use of GaN HEMTs can raise efficiency by more than 2% at 50 kHz and 3% at 200 kHz. Furthermore, as shown in Figure 3 of [50], output power can be increased from 2500W to 3750W where Si MOSFETs and GaN HEMTs are used, respectively. We also need to note that an equivalent power loss level and a 50 kHz switching frequency is achieved that yields higher than a 60% unexpected rise in power-rating values, not needing cooling in addition.

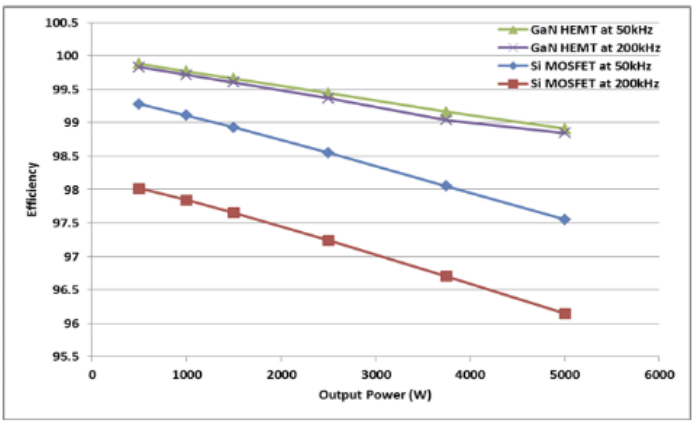

Figure 2.2 Efficiency comparison [50]

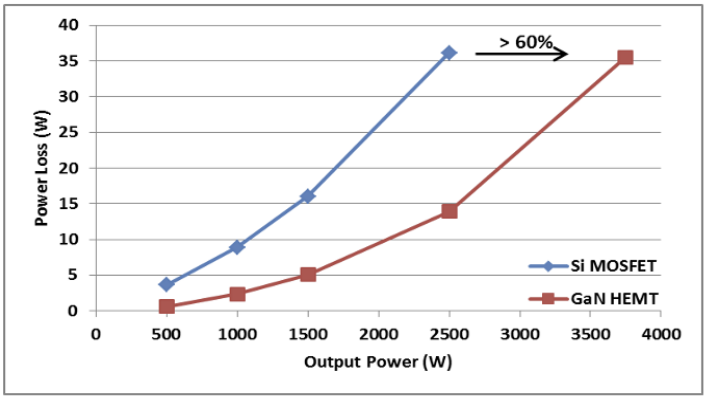

Figure 2.2. Improvement of Power rating [50]

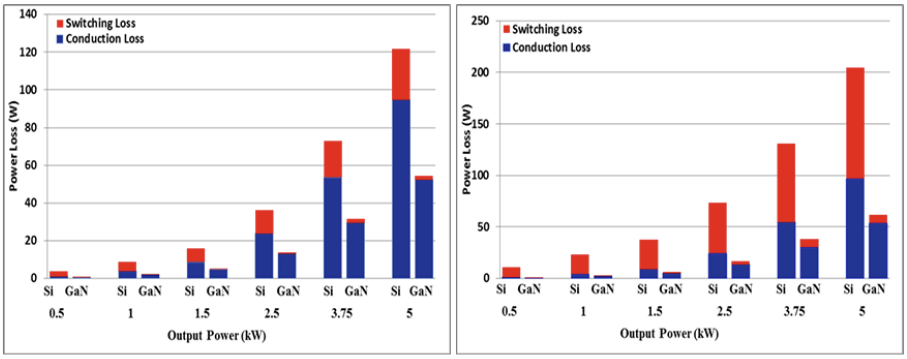

Figure 2.3 Comparison of Power losses in Si MOSFET and GaN HEMT: (a) 50 kHz (b) 200 kHz [50]

Cons

1. Poor Conductivity with Thermal Characteristics:

- Even though GaN devices offer higher efficiency, their thermal conductivity is less than that of Sic This limitation can become problematic in high power or temperature applications, as reported in [46, 47, 51, 52]. Operating under elevated temperatures and power levels can pose challenges when using GaN due to its reduced ability to dissipate heat.

2. Cost:

- GaN devices are initially more expensive than conventional Si-based components. Although their high efficiency and reduced passive and cooling requirements can cut total system costs by around 10%, the relatively high upfront cost still hinders widespread adoption [53].
- In the long run, energy savings and decreased cooling expenses may offset these initial investments, but the higher purchase price can discourage some use cases not to transition to solutions relying on GaN technology.

3. Implementation with Complexity:

- Incorporating transistors powered by GaN technology into current systems might be intricate thanks to their distinct nature and the requirements for specialized circuitry of gate drive. Their faster speed for switching requires meticulous design to mitigate electromagnetic interference, also shortly known as EMI, and completely exploits the advantages associated with this technology.
- A key concern in inverters powered by GaN technology is to manage the sampling to switching ratio of the frequencies. The mentioned ratio makes it complicated to design strategies for efficient control and can constrain the system performance in scenarios requiring frequency with high values. Controlling the systems with frequencies of higher values effectively becomes especially important when operating at MHz-level frequencies in soft-switching systems; engineers must quickly detect soft-switching conditions and deliver accurate PWM signals.

1. Challenges Associated with the Layout of the PCB:

- The layout for the Printed Circuit Board - PCB is also another critical concern to design featuring high rates of the slew and switching frequencies of high values. While

transistors made by Si and SiC frequently use packages with through-hole (where shared source inductance is a primary issue), GaN transistors mainly contend with inductance of power loop, due to the parasitic inductance coming from PCB traces made by copper. Component placement and orientation on the PCB can heavily influence this inductance. Additionally, GaN transistors have a low gate threshold for the voltage as well as a limited blocking value for the voltage, and no breakdown for the avalanche. Thus, it makes imperative for implementing rigorous layouts for PCB practices to avoid parasitic inductance problems [54-56].

FPGA controlled GaN-based Inverters

Recent developments for technology related to the business of semiconductors technology use Gallium Nitride materials and enable switching frequencies between 50 MHz and higher, a significant leap compared to the typical range for transistors relying on IGBT where this range starts from 2 and goes till 20 kHz. Devices using GaN technology provide several pros, such as lower losses for switching and the capability for functioning effectively despite higher temperatures. We showed in earlier sections, various studies have demonstrated the practical success of these devices in power electronic converters. Despite their advantages, GaN-based inverters present challenges, including thermal management, interference associated with EMI, and control complexity at frequencies for high values. A key issue in inverters using GaN technology designs is the frequency ratio of sampling by switching. Many existing platforms for outputting signals for PWM mode, including dedicated controllers for PWM as well as microcontrollers, constraint themselves to frequencies at the scale of tens (10s) of kHZ. All traditional analog as well as digital designs are therefore inadequate for GaN applications requiring operation in the range of MHz values for the frequency.

To overcome the above-mentioned limitation, FPGAs provide a promising alternative. With associated advanced architecture, FPGAs might produce pulses with high frequency values, allowing to accurately control critical values even at tens of MHz. For example, we can list duty cycle as well as frequency and phase difference of signals of PWM to be such values, Moreover, manufacturers producing these devices now provide system-on-chip (SoC) solutions, further expanding their application to not only power electronics but also systems for electrical drives [57-64].

How FPGA works

Thanks to the advances associated with designing and manufacturing better integrated circuits, nowadays FPGAs are increasingly powerful and accessible. This drives their widespread adoption in power electronics. Their ability to perform parallel processing and handle complex computations gives them an edge over conventional microcontrollers and custom units such as DSPs, digital signal processors for certain applications. [49] compares and contrasts the microcontrollers as well as DSPs against FPGAs in a greater detail than this text and we recommend an interested reader to check that. Notably, computationally intensive control use cases use FPGAs. One good example is Model-Predictive-Control that is described in literature with details [66-70]. Their abundant pins for Input/Output and resources for logical processing position them highly suitable for use cases involving sophisticated strategies for modulation as shown in the two recent works of [71] and [72].

Programmable architecture powered by silicon makes FPGAs stand out and allow to create the custom functions for digital purposes. This flexibility sets them apart from conventional

processors, which fixed architectures limit, finite cores, and instruction execution serially [73]. FPGAs excel in real-time operations and true parallel processing, enabling multiple processes to run concurrently without competing for shared resources. Each task is assigned to a specific chip section and operates on its own and without interfering to and from concurrent tasks. This design ensures consistent performance even as additional functions are integrated. Additionally, FPGAs offer accurate operations relying on timing based on hardware as well as exceptional reliability as shown in the [74].

We present a common list of components of the FPGA as well as its Input/Output blocks laid out in matrix form with the help of Figure 2.4. All constituent components talk to each other through programmable hardware-based interconnections [75].

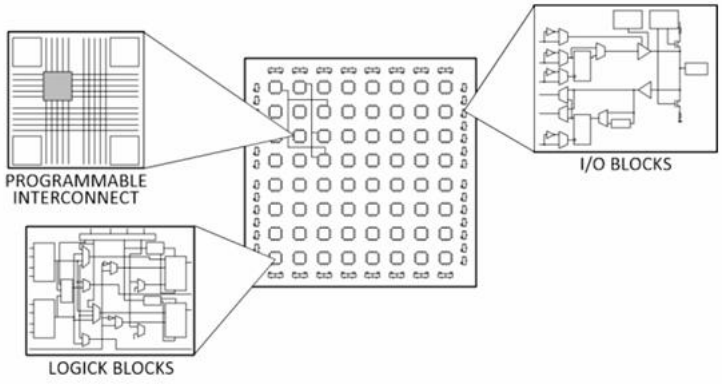

Figure 2.4 Structure of FPGA [74]

To enhance design efficiency and implementation, FPGA programming utilizes various development platforms and programming languages. Main hardware description languages, abbreviated as HDLs, include Verilog as well as its flavor SystemVerilog and VHDL. These languages empower designers to define behavior as well as structure for their digital circuits across various abstraction levels. Practitioners have been using Verilog extensively for the use case such as to create signals for PWM and implement reversible logic gate-based ALUs as shown in the [76].

There is a wide array of development tools to complement HDL. Some of them simplify the design, while the others help with simulation as well as synthesis. It is also possible to fully deploy systems powered by FPGAs using these. Some examples are ISEDesignSuite by Xilinx, Lab VIEW for FPGA, MATLAB&Simulink, and QuartusPrime by Intel. Mentioned tools maximize the power of FPGAs in fields such as communication (DSP) as well as embedded systems [77, 78].

Impacts of FPGA on Inverters

FPGAs have revolutionized inverter design and operation, particularly for use cases requiring high performance as well as efficiency. Reconfigurability and parallel processing support them position FPGAs as ideal to implement advanced strategies of control as well as manage the frequencies at high switching values required by contemporary inverters.

Being able to operate at very high frequency values makes FPGAs stand out from other types of devices for the control platform. This capability is really important for use cases like industrial-grade motor drives or power converters that require both speed and higher precision.

For example, authors of [79] describe how an FPGA implemented a vector-based algorithm to control a 200 kHz motor drive system based on GaN technology achieving a level of precision that traditional microcontroller units (MCUs) cannot match. Additionally, thanks to FPGAs' reprogrammable nature it is convenient updating and modifying algorithms for controlling without changing hardware, making them highly adaptable to evolving requirements.

We can show the parallel processing nature of FPGAs as another advantage, that allows them to handle multiple tasks simultaneously [80]. This phenomenon especially benefits use cases where real-time signal processing is necessary. Moreover, efficiency and performance for inverters are enhanced thanks to the outstanding ability of FPGAs to switch at high rates. Inverters relying on GaN technology benefit from this phenomenon and have better overall performance for systems of power electronics [81].

On the other hand, application of FPGAs might come with challenges. Designing and programming them is complex and often requires expertise in HDLs like VHDL or Verilog, leading to longer development times and increased costs compared to microcontroller-based solutions. Additionally, FPGAs generally have a higher upfront cost than microcontrollers or DSPs, which, while offset by their performance and flexibility, is still an important consideration. Furthermore, in use cases that do not fully utilize parallel processing units, FPGAs can use more power than highly optimized alternatives relying on microcontrollers.

State of Art of FPGA controlled Inverters

Recent studies have shown the effective employment of FPGAs in diverse inverter, renewable systems, and motor driving applications.

Research [82] explores development of a speed control method on a FPGA board using the method named FOC. Another component which is called Sliding Mode Observer is integrated with sensor less FOC as well as loop based on a phase lock. They generate speed data using the NIOS II processor while other topologies rely on an implementation powered by an FPGA board. The results demonstrated smoother back-EMF waveforms during transitions from stationary to acceleration. Similarly, authors of [83] design and evaluate a control of 3-phase BLDC motors on top of FPGA boards and highlight its advantages for power and safety then compare it against systems based on conventional microcontrollers. Despite all mentioned benefits, microcontrollers remain a practical choice in some scenarios due to their faster development time and lower costs.

Combining FPGA board-based control mechanisms with inverters relying on GaN technology has addressed a wide array of challenges in power-electronics. Some examples may be to enhance efficiency of these devices as well as making them compact, simultaneously achieving better response times for them. For instance, authors of [64] describe a 9-level FCML inverter controlled by an FPGA, achieving frequencies for high switching rates of up to 4 MHz with accurate strategies of modulation, leading to a maximum efficiency exceeding 99 percent. Similarly, authors of [68] present an inverter for thirteen (13) level FCML relying on lower voltage GaN-FETs (Figure 2.5), where thanks to using FPGA based control it reduces commutation loop inductance and supports a 120 kHz switching frequency. This design makes it minimum for the size of passive components as well as it achieves a total harmonic distortion as low as 0.7 percent.

In [85], FPGAs were shown to produce high-frequency PWM signals that are crucial to the dynamic performance of power devices relying on GaN technology. These signals help to accurately control and switch fast, leading to a better performance for GaN powered inverters.

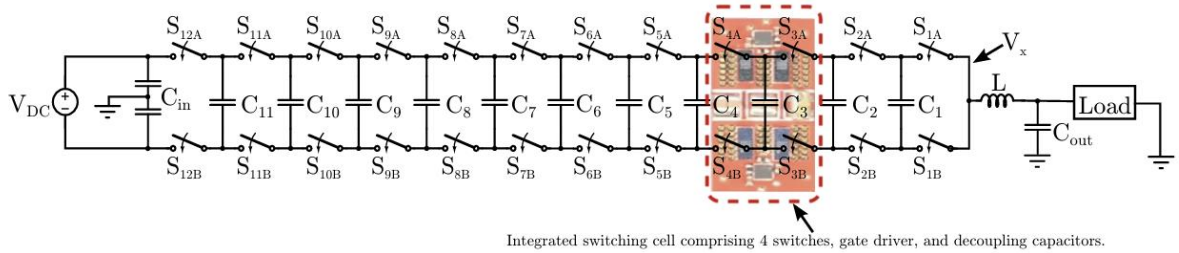

Figure 2.5 Inverter schematic [84]

The study in [79] focused to implement a vector-based algorithm for controlling a system of drive for a motor powered by GaN relying on an FPGA board, achieving a switching frequency of 200 kHz. This implementation leads to an improved power density as well as reduced losses for switching. Being based on the Xilinx ZYNQ-7000 controller, the system integrates control algorithms for motors directly on the FPGA board which in turn makes it possible to operate at extremely high speed but accurately. Results from their experiments confirmed the system might precisely deliver controlling motor in a single-cycle, and it showcases the reliability and efficiency of the algorithm.

Another study in [86] applied a Field-Oriented Control algorithm to a 3-phase induction motor, where inverter using GaN technology along with FPGA board for control and benefited from the advantages of technology to achieve frequencies for switching of up to 100 kHz, significantly higher than the range of from 2 to 20 kHz typical for transistors using IGBT materials. The authors implemented their system with VHDL on a Zybo Z7010 FPGA of Xilinx, decreased switching-losses and improved density. (HIL) simulations validated the design, confirming its effective operation at high frequencies. The FPGA's processing time, measured at 0.22 µs, as depicted in Figure 2.6, demonstrated the capability for the system to handle high frequency switching efficiently. Recent advancements highlight the successful integration of FPGAs in controlling inverters, particularly for renewable energy systems and motor drives.

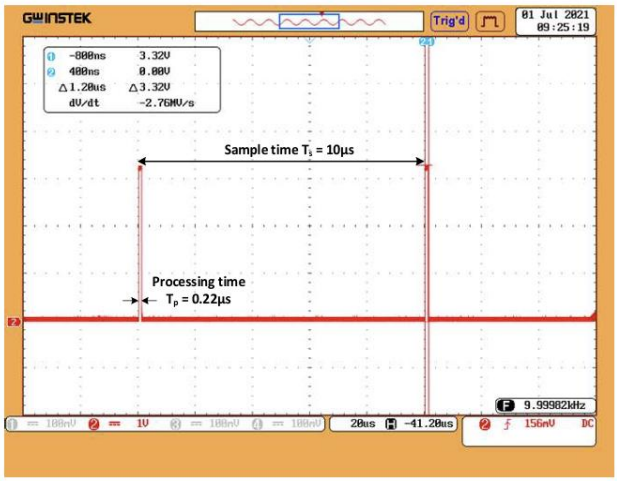

Figure 2.6 Processing time [86]

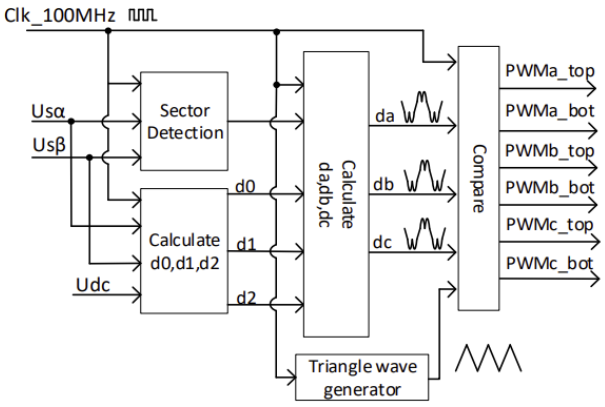

Figure 2.7 Implementation of SVPWM in FPGA [86]

For example, the study in [87] examined an FPGA-controlled 3-phase 3-level Voltage Source Inverter (VSI) designed to connect arrays of multi-string photovoltaic devices into the grid for the power, as illustrated in Figure 8. Their system aimed to ensure efficient conversion from DC to AC while keeping reliability as well as performance when varying conditions happen. It includes a global inverter, MPPT capabilities for every string of PV, and a tightly-decoupled controller for the grid to synchronize. The control algorithms were implemented on the evaluation kit branded Virtex-6 ML605 using the System Generator made by Xilinx, which enabled precise control and rapid processing. Simulation and hardware co-simulation results validated the system, achieving a frequency for the grid of 50 Hz where a THD fewer 2.5 percent.

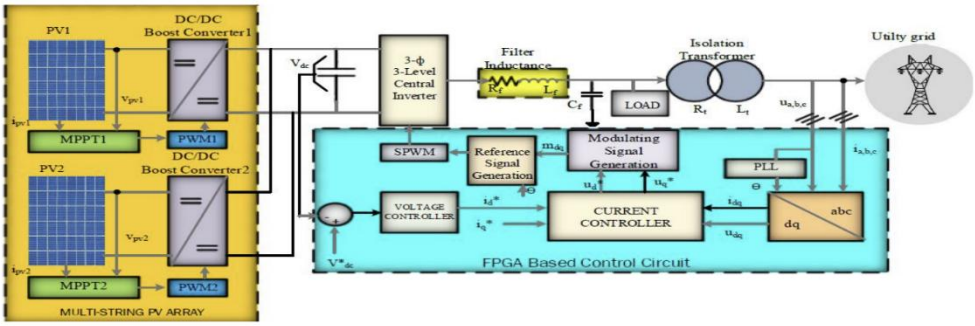

Figure 2.8 Proposal for the system structure [87]

Additionally, GaN based inverters, researchers extensively studied FPGA controllers for SiC-based inverters. The work in [88] focused on improving speed of computations and precision of the control using an implementation of MPC implemented on an FPGA board for a quasi-Z-source inverter relying on SiC technology (qZSI). The FPGA's parallel processing reduced computation times to just 1 µs, enabling support for the high switching frequencies typical of SiC devices. The MPC algorithm was validated through MATLAB Simulink simulations and experimental testing on a 3-phase qZSI based on SiC as depicted in Figure 2.9 which demonstrates control consistency along with ripple for the output current minimal value. These findings emphasize FPGAs' position to enhance the performance and efficiency to compute in SiC-based power converters.

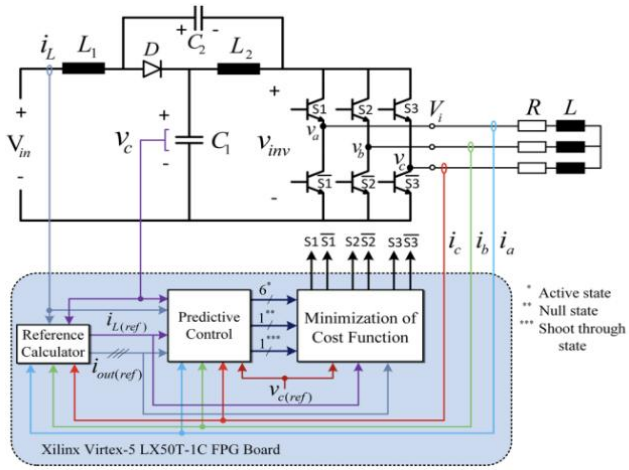

Figure 2.9 Structure of FPGA controlled SiC based qZSI [88]

Another study [89] introduced a CHIL (Controller Hardware-in-the-Loop) testbed for validating both basic and advanced control strategies for SiC-based PV string inverters (Figure 10). The testbed was developed to provide an effective in terms of financial costs as well as environmentally friendly to test switching really fast controls for inverters not causing damage for actual physical hardware. It used an Opal-RT OP5607 board for FPGA expansion unit to simulate in real time, achieving a 20 kHz switching frequency with a 500 ns time step. Results demonstrated that the testbed for CHIL closely matched setups of experiments, with a maximum efficiency for the inverter 99 percent aligning with measurements of experiments of 98.2 percent. This paper demonstrated FPGAs' ability to deliver precise control for high-speed power electronics and validated the CHIL testbed to be a robust amenity to develop and test complex control logic for inverters.

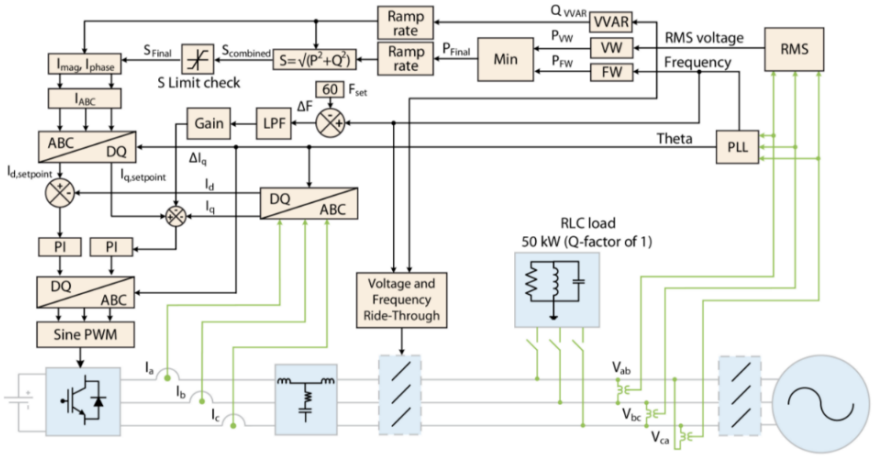

Figure 2.10 Diagram for the Inverter Control [89]

Beyond inverters powered by GaN/SiC, research has also explored the use of FPGAs in inverters relying on Si IGBT technology [90-92]. For instance, paper [90] developed an FPGA-controlled system to regulate the frequency as well as the voltage in VSIs employing an Adaptive Linear Neuron with Frequency Locked Loop structure. FPGA's capabilities in high-speed computation and real-time signal processing enabled it to decompose harmonically and synchronize accurately, leading to significant improvements for the system performance even under harsh weather or environmental situations.

Similarly, the work in [91] implemented a complete system to control a connected to grid and controlled by current VSI (Figure 2.11). This system focused on phase tracking with a Phase Locked Loop, logic for inverter control, and active & reactive power flow management. This system uses a module named compactRIO along with code that is automatically generated by VHDL LabView and achieves fault detection with very high speed as well as burst logging. Tested on a setup which is joined to the grid in 30 kW, model successfully maintaining power factor of unity and handles step changes in power flow which is active. These results highlighted superior speed and flexibility of FPGA-based controllers compared to traditional microprocessor systems.

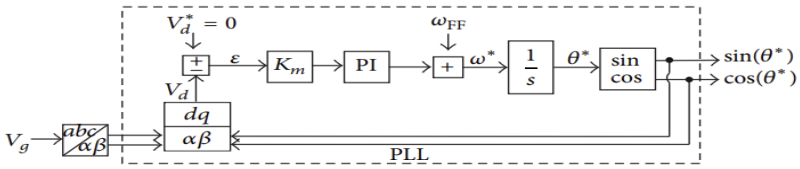

Figure 2.11 Block diagram of PLL for grid-phase tracking [75]

CHAPTER THREE
RESEARCH APPROACH

Architecture

This chapter outlines the methodology adopted for designing, modeling, and controlling the 3-phase grid connected photovoltaic system as proposed before, that is demonstrated in Figure 3.1. The grid connected system that we proposed comprises four main parts: the PV generator, DC-DC boost-converter with MPPT, DC/AC-inverter, and LCL filter, all working cohesively to extract, process, and deliver renewable energy to the grid. The PV generator converts solar irradiance into a DC voltage and current. We fed the variable DC output into the boost-converter, which increases the voltage level to the value desired meanwhile implementing the MPPT-algorithm using the P & O method to ensure maximally extracting the power from the array PV. We then supply the boosted DC voltage to the DC-AC inverter, which converts it into a three-phase sinusoidal AC signal suitable for grid injection. The inverter is controlled via voltage/current regulation loops in the dq-reference frame and synchronized to the grid using a PLL to ensure phase as well as frequency alignment.

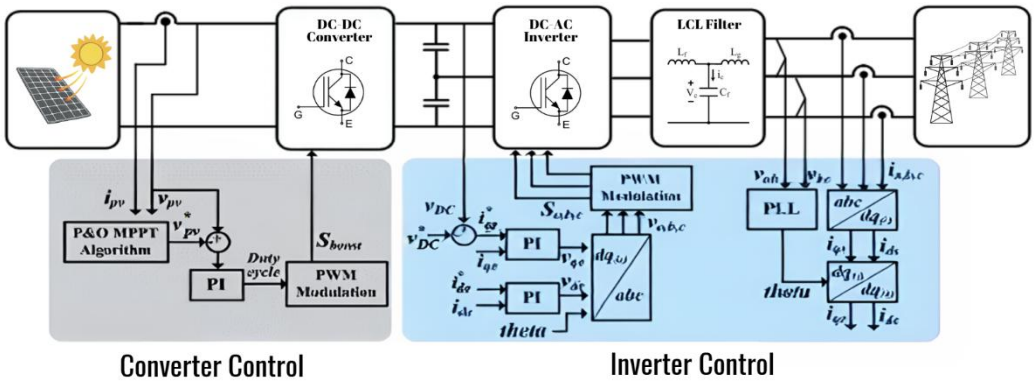

Figure 3.1. The proposed diagram of 3-phase grid-connected system

Finally, we attenuate high frequency switching harmonics from the inverter using an LCL filter to ensure the output current meets THD and power quality grid standards. These components work together to support efficiently integrating solar based power to the grid.

The electrical specifications for our system are provided in Table 3.1.

Table 3.1. System Electrical Specification

Parameter	The System	Symbol	Value
Open Circuit Voltage	Solar Panel	V_{OC}	480 V
Voltage at MPP		VMPP	405 V
Rated power		P_{rated}	100 kW
Switching frequency	Converter	f_{BC}	10 kHz
Inductance		C_{BC}	3227 uF
Capacitance		L_{BC}	1.45 mH
Output voltage		V_{BC}	600 V
Switching frequency	Inverter	f_{SW}	100 kHz
Capacitance	LCL filter	C_{LCL}	100 uF
Inductance		L_{LCL}	500 uH

PV Generator

The PV generator, which forms the proposed system's primary energy source, is modeled based on the single-diode equivalent circuit (Figure 3.2). This model effectively captures the nonlinear electrical behavior of photovoltaic cells.

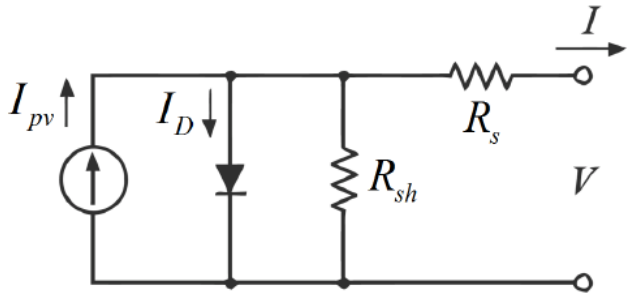

Figure 3.2. Equivalent circuit of Solar panel (real case) [93]

The SunPower SPR-225E-WHT-D module is used for the PV generator. It consists of a series-parallel arrangement of PV modules, as depicted in Figure 14. The modules are organized into 10 series-connected modules per string and 47 parallel strings. This configuration ensures sufficient output voltage and current for the system's requirements under varying environmental conditions.

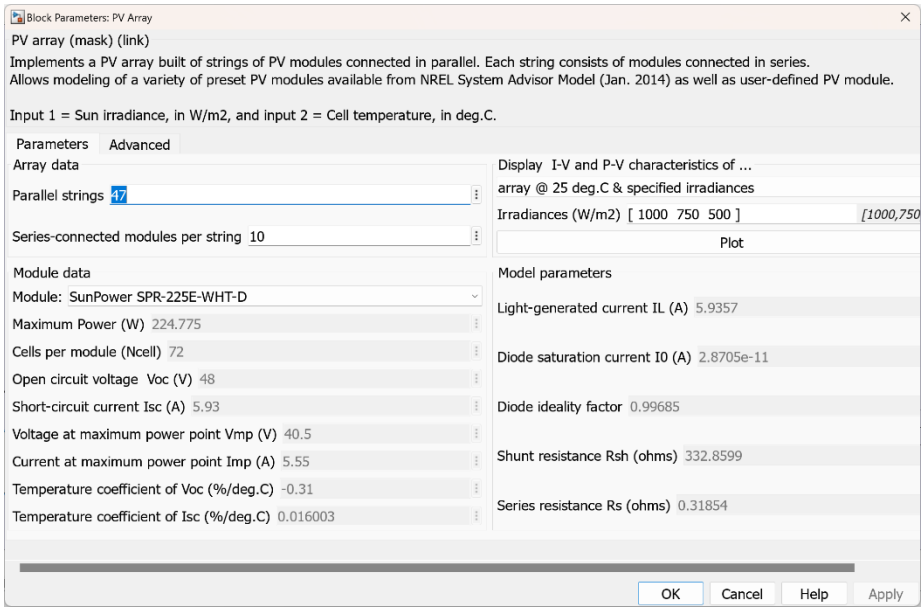

Figure 3.3. Key Parameters of PV array

The (I/V) and (P/V) characteristic for an array PV can be identified by the environmental aspects such as irradiance of solar and temperature of the cell. The governing equations for the PV generator outputting current and voltage are as follows:

Current through the diode [94]:

$$I_D = I_0 \left[\exp\left(\frac{q(V + IR_s)}{kT} \right) - 1 \right] \quad [94]$$

Output current of the PV generator:

$$I = I_L - I_D - \frac{V + IR_s}{R_{sh}} \quad [94]$$

Where:

- I_0 is current for diode's saturation.
- I_L is current generated by light, proportional to solar irradiance.
- T is cell-temperature measured Kelvin.
- R_s & R_{sh} represent the series and shunt resistances, respectively.
- q is the charge for the electron (1.6×10^{-19} C).
- k is the constant for the Boltzmann (1.38×10^{-23} J/K).

The PV generator's performance is characterized by its I/V and P/V curves, as depicted in Figure 15. These curves illustrate the impact of varying solar irradiance on the output characteristics:

- At a high irradiance of 1000 W/m^2, the PV array delivers maximum power (P_{max}) of approximately 100 kW, corresponding to a voltage close to the maximum power point (V_{MPPT}).

- When irradiance reduces to 750 W/m^2 or 500 W/m^2, both the output current and power decrease significantly, indicating the importance of MPPT to dynamically adjust the operating voltage and maximize energy extraction.

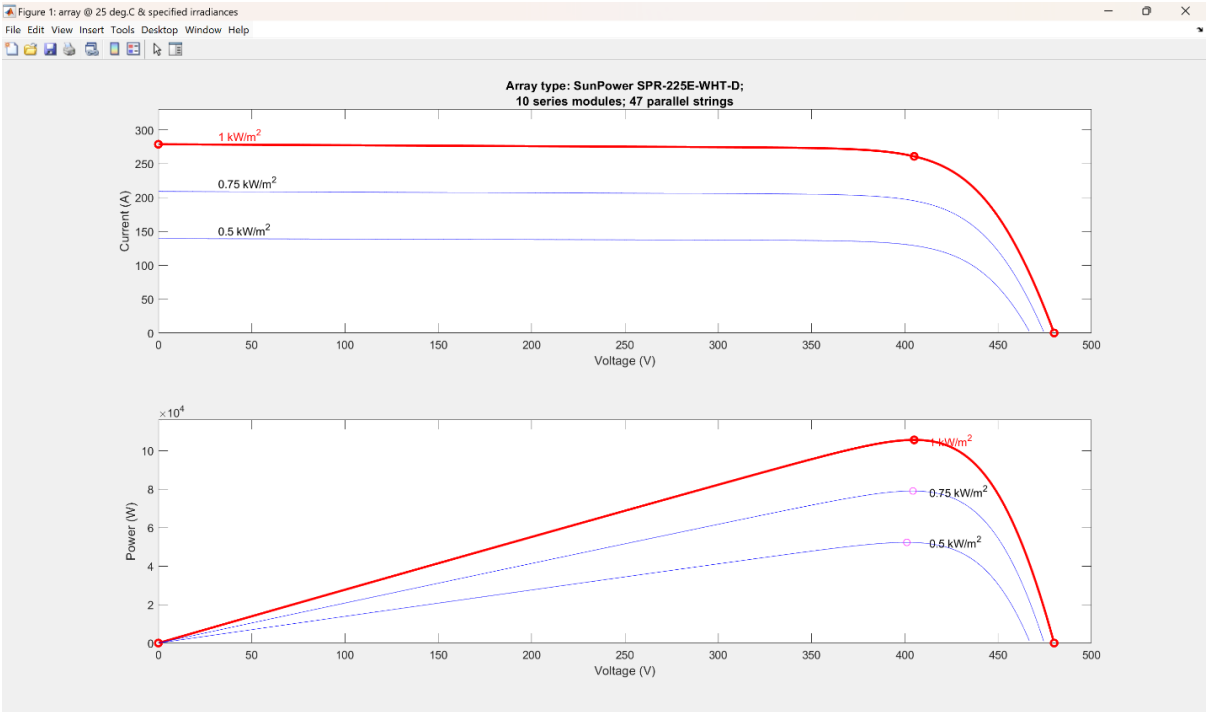

Figure 3.4. I-V & P-V characteristic curves

DC-DC Boost Converter with MPPT

It is an integral component of the proposal for two stage grid connected photovoltaic system – PV system. Its primary function is to step-up the lower voltage DC output from PV into a higher, stable DC-voltage level required by the inverter. The design incorporates MPPT algorithm to dynamically adjust the operating point of the PV, ensuring maximal power extraction with changing environmental situations such as solar irradiance, temperature and so on.

The boost converter comprises essential components, including an inductor, a capacitor, a device for switching, and a diode. The operational principle is based on energy storage in the

inductor while the online state for the switch as well as energy-transfer to the load while the offline state. A capacitor smoothens output voltage by filtering ripples. The converter design parameters are calculated as follows [95]:

- Inductor Sizing:

$$L = \frac{V_{in}(V_{out} - V_{in})}{f_{BC} \mathrm{V_{out}} \Delta I} \quad [95]$$

- Capacitor Sizing:

$$C = \frac{I(V_{out} - V_{in})}{f_{BC} \mathrm{V_{out}} \Delta V} \quad [95]$$

where V_{out} is output voltage 600V, I is an output current $\left(\frac{100\ kW}{600\ V} = 166\ A\right)$, ΔV is allowable ripple voltage (6 V), ΔI is allowable ripple current (20 A), and V_{in} is operating range for solar panel.

In this system, the frequency for switching is set at 10 kHz, and the designed values are calculated:

- Inductance: L=1.45 mH
- Capacitance: C=3227 μF

These parameters ensure efficient operation of the converter with minimal ripple in the output V and I, meeting the demands of the downstream inverter.

The MPPT functionality is implemented using the P&O algorithm, a widely adopted method due to its simplicity and efficiency. It is also capable of adjusting to rapid changes in irradiance and temperature. This algorithm perturbs the working voltage of the PV array and observes the resultant changes regarding power. The algorithmic steps are as follows (refer to Figure 3.5 for the flowchart):

1. Measure the PV array's voltage (V_{PV}) & current (I_{PV}).
2. Calculate instantaneous power [94]:

$$P = V_{PV} \cdot I_{PV} \quad [94]$$

3. Compare the current power (P_n) with the previous power (P_{n-1}):
    - If $P_n > P_{n-1}$, adjust the reference voltage (V_{ref}) in the same direction as previous perturbations.
    - Otherwise, reverse the direction of perturbation.
4. Repeat the process to ensure system is operational at the maximal power point.

The reference voltage (V_{ref}) is used for adjusting the duty-cycle (D) from the boost converter's switching device via a PI-controller. Duty cycle controls storage for the energy and transfer in the inductance L, thereby regulating the voltage at the output.

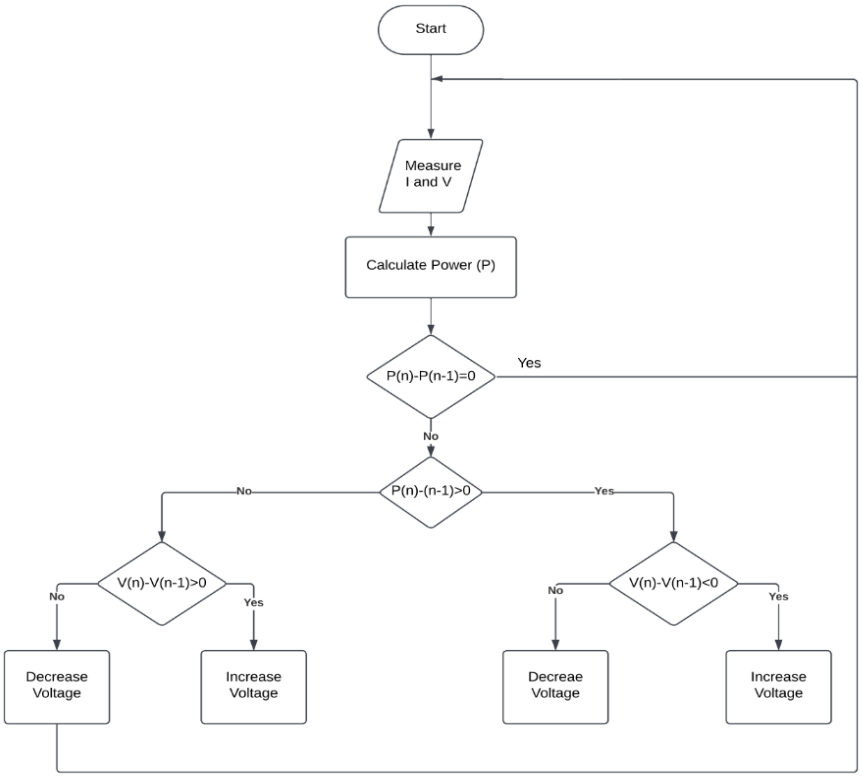

Figure 3.5 Flowchart of P&O Algorithm [95]

Based on flowchart in Figure 3.5, the algorithm for P&O is implemented via the boost controller. Measured PV-voltage (V_{PV}) and current (I_{PV}) are inputted to MPPT block, which calculates the optimal V_{ref}. The controller PI minimizes the error between V_{ref} and the actual V_{PV}, producing a controlling signal determining duty-cycle (D) by comparing the controller output with the sawtooth waveform. This cycle of duty is used for generating PWM signals to switch device in the boost conversion.

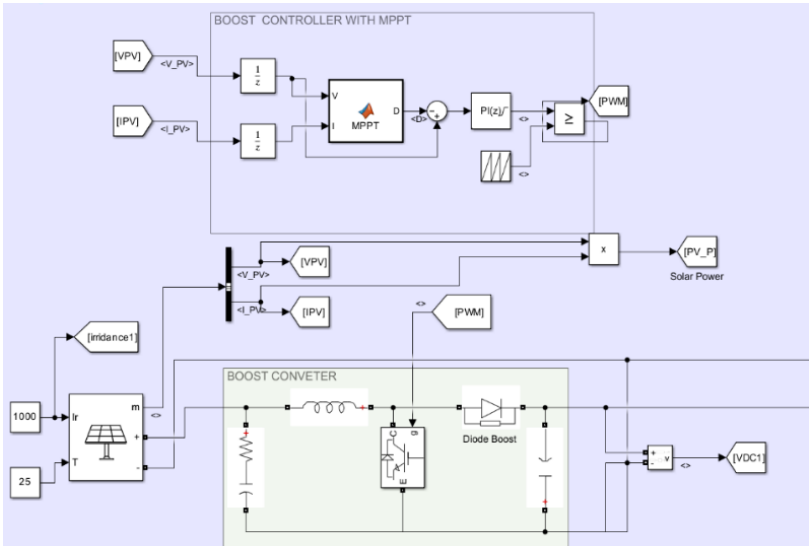

Figure 3.6. Structure of the DC-DC converter

Inverter

The strategy of the inverter controller has a crucial point in ensuring the DC powder obtained from the array PV is efficiently converted into a high-quality three-phase AC output that meets grid standards. The inverter system (Figure 3.7) ensures proper output grid synchronization, Vand I regulation, and DC/AC conversion using PWM.

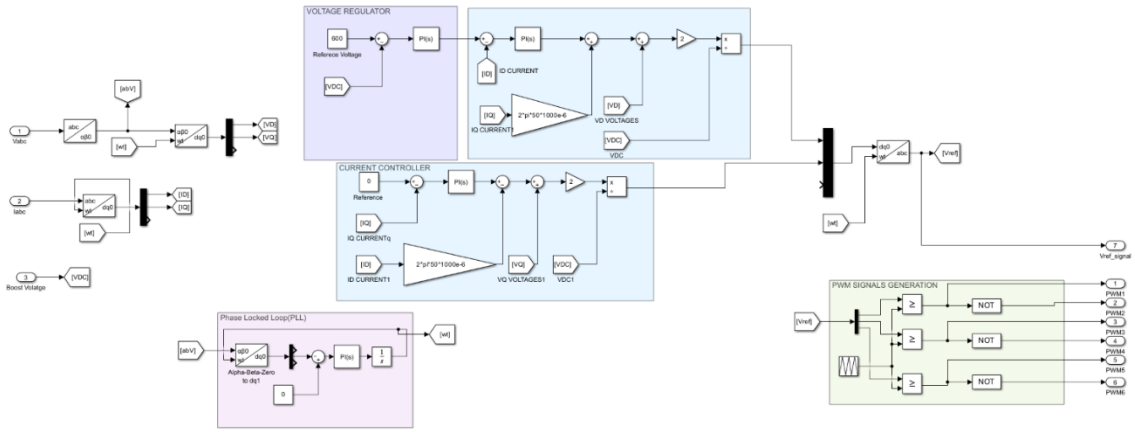

Figure 3.7. Simulink Structure for the Inverter System

- The **voltage-regulator** ensures DC-link-voltage (V_{DC}) remains the same at the reference value, which is essential for stable inverter operation. This is implemented using a controller (PI) for error minimization of the measuring DC-voltage (V_{DC}) and the ref-value ($V_{DC,ref}$).

The control signal outputted by the controller - PI serves as the reference for the d-axis current (I_d^*):

$$I_d^* = PI\left(V_{DC,ref} - V_{DC}\right) \qquad [95]$$

The output for the voltage controller feeds into the I controller, ensuring that the active powder flow to grid is appropriately regulated.

- The **current-control** block regulates the output I to the designed inverter in the d-q synchronous reference-frame, where:

  Id: Controls the power component – active.

  Iq: Controls the power component – reactive.

  The current control consists of:

  1. Control Loop for the Inner Current: Tracks the d- & q- axis reference currents (I_d^* and I_q^*) using PI controllers to minimize tracking errors.

  2. Feedforward Decoupling: To decouple the d-q components and improve the dynamic response, a compensation term $2\pi fLI_q$ increased by the direct(d)-axis control and $2\pi fLI_q$ to the quadrature(q)-axis control, where f is grid frequency and L is filter inductance.

The output voltages for d q frame (Vd / Vq) are calculated as follows:

$$V_d = PI(I_d^* - I_d) + 2\pi f L I_q + V_{DC} \quad [95]$$

$$V_q = PI\left(I_q^* - I_q\right) - 2\pi f L I_d \quad [95]$$

These voltages are transformed back to the abc-frame by inverse Park transformation, providing the reference voltages ($V_{ref,abc}$) for the PWM block.

- The **Phase Locked Loop** is used for aligning the output for the inverter-voltage with grid-voltage. PLL operates as follows:

    1. The three phase grid voltages (V_{abc}) first converted into stationary αβ-frame using Clarke-transformation.

    2. The αβ components are transformed to the rotating d-q reference-frame using:

$$V_d = V_\alpha \cos(\theta) + V_\beta \sin(\theta) \quad [95]$$
$$V_q = -V_\alpha \sin(\theta) + V_\beta \cos(\theta) \quad [95]$$

    3. A controller of PI is applied for minimizing the Vq-component, ensuring Vq→0, which aligns the d- axis with the grid-voltage-vector.

    4. The phase angle θ generated by the PLL is used for coordinate transformations and synchronization.

- The **reference voltages** in the abc-frame ($V_{ref,abc}$) are compared against a high-frequency carrier signal (sawtooth) for generation of the PWM signals for six IGBTs. PWM signals control switching states for inverter, producing sinusoidal AC-output.

The modulation indices m_d and m_q for PWM generation are derived as:

$$m_d = \frac{2V_d}{V_{DC}} \qquad m_q = \frac{2V_q}{V_{DC}} \quad [95]$$

These indices are sent to the multiplexer and transformed back to the abc-frame for the generation of switching pulses.

PWM signals are used for six Insulated Gate Bipolar Transistors (IGBTs) arranged in a three-leg bridge topology, as shown in Figure 3.8.

Six IGBTs are used, labeled S_1 to S_6, where:

- **Top-switches:** S1, S3, S5 (linked to V_{DC+}).
- **Bottom-switches:** S2, S4, S6 (linked to V_{DC-}).

The PWM signals determine when each switch turns ON or OFF, producing a series of high-frequency pulses that approximate sinusoidal AC waveforms at the output terminals (V_R, V_Y, V_B). They ensure complementary operation between the top as well as bottom end switches of same leg. For every leg, the upper as well as lower switches must not be *on* simultaneously to prevent short circuits.

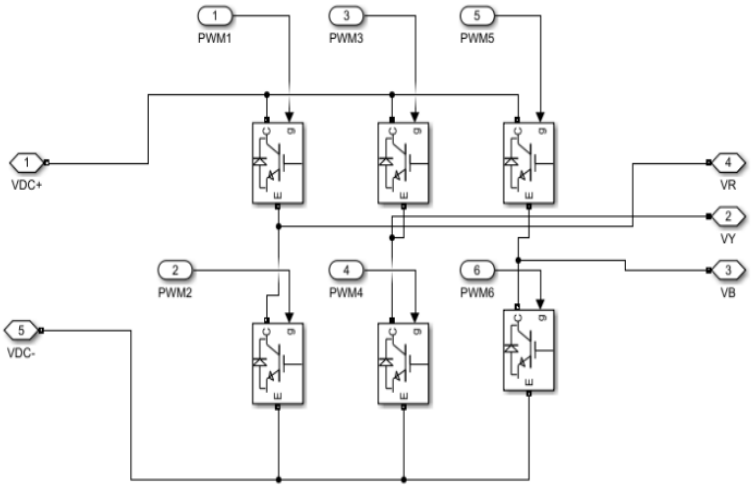

Figure 3.8. PWM switching

The overall inverter control flow can be shown as summary with the followings:

1. Measure grid voltages (V_{abc}) and inverter currents (I_{abc}).
2. Use PLL to extract the phase angle (θ) and synchronize with the grid.
3. Transform the measured currents as well as voltages into the d-q reference frame.
4. Use the voltage regulator to generate reference d- axis current (I_d^*).
5. Regulate the d-q currents using PI controllers to generate V_d and V_q.
6. Transform V_d and V_q to the abc- frame via the inverse Park transformation.
7. Generate PWM switching signals based on the reference voltages ($V_{ref,abc}$).

Finally, it generates three-phase AC voltages, phase-shifted by (120°) to ensure a balanced three phase output. These voltages are then passed over the LCL filter to exclude high-frequency harmonics and ensure grid compliance.

LCL Filter

The LCL filter is an important component in the proposed grid-tied system, designed to mitigate the high-frequency harmonics generation of the operation for switching inverter. By attenuating these harmonics, the LCL filter ensures compliance with grid standards for power quality and total harmonic distortion (THD), thereby improving the total performance and efficacy for the system.

LCL filter consists of two inductance elements (L1 / L2) and single capacitance (C), wired as depicted in the system diagram (Figure 3.9). The key functions are:

- Harmonic Attenuation: Reduction of high-order-harmonics generated during PWM switching in the inverter.
- Improved Power Quality: Ensures the grid current is nearly sinusoidal with minimal distortion.

- Dynamic Performance: Improves harmonic attenuation compared to traditional L-filters as well as decreasing component size & cost.

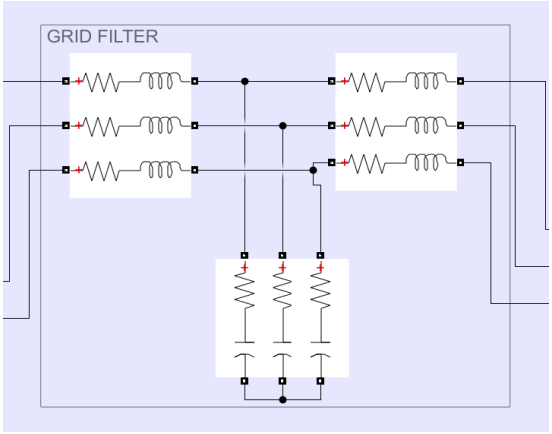

Figure 3.9. Developed LCL filter in Simulink

Design for the LCL filter requires determining the proper values for L_1, L_2, and C to meet the filtering requirements without causing resonance issues. The calculations are as follows:

- Capacitor design

  According to the reactive power, the filter capacitance C is calculated:

$$Q = \frac{V^2}{\frac{1}{2\pi f C}} = \%5 \cdot S \qquad [95]$$

$$C = \frac{0.05 \cdot S}{V^2 \cdot 2\pi f} \qquad [95]$$

  Substituting:

  - S = 100 kVA (system apparent power),
  - V = 230 V (grid voltage),
  - f = 50 Hz (grid frequency),

$$C = \frac{0.05 \cdot 100000}{230^2 \cdot 2\pi \cdot 50} \approx 300\mu F$$

  As a result, C for each capacitor is 100.28 µF.

- Inductor Design

  To determine the filter inductances (L_1 and L_2), grid-current ripple and resonance frequency must be taken into account.

  Minimum Inductor value is:

$$L = \frac{1}{(2\pi f_s) \cdot I_g \cdot \left(1 - \left(\frac{2\pi f_r}{2\pi f_s}\right)^2\right)} \quad [95]$$

where:

- fs=10 kHz (switching frequency),
- fr=1 kHz (resonance frequency),
- Ig=0.434 A,
- Vx=207 V,

$$L_{min} = \frac{1}{(2\pi \cdot 10000) \cdot 0.434 \cdot \left(1 - \left(\frac{2\pi \cdot 1000}{2\pi \cdot 100}\right)^2\right)} \approx 76.68\ \mu H$$

Maximum Inductor value is:

$$L_{max} = 0.2 \cdot \frac{V_{grid}}{2\pi f I} \quad [95]$$

where:

- Vgrid=230V,
- I=144.4A,

$$L_{max} = 0.2 \cdot \frac{230}{2\pi \cdot 50 \cdot 144.4} \approx 1\ mH$$

Distributing this between two inductors:

$$L_1 = L_2 = \frac{L_{max}}{2} = \frac{1mH}{2} = 500\ \mu H$$

FPGA-in-the-Loop Approach

In this section, we will go over the steps to take in order to set up and validate a FIL simulation in Matlab - Simulink for a 3-phase grid-tied inverter control system. By bridging the gap between software-based simulation and complete hardware deployment, FIL simulation enables the execution of control techniques on FPGA hardware in real-time. We put it through its paces under realistic time and resource limitations. We construct the model initially, and then we transform it into a fixed-point format. After that, you'll need to deploy the model to the hardware and configure the operational frequencies. We monitor the time it takes to run and analyze the data to ensure the system is running well after deployment. The FIL system incorporates a HIL technique, and we use Simulink to simulate the power circuit while the controlling scheme runs on the FPGA hardware. With the FPGA generating control signals like PWM outputs in real time, the software (Simulink) and hardware (FPGA) function as a coprocessor in a continuous feedback error exchange.

The relationship among the sampling (f_s) and the switching frequency (f_{sw}) is critical to ensure accurate timing and synchronization of system. The following diagram illustrates the FIL simulation flow (Figure 3.10):

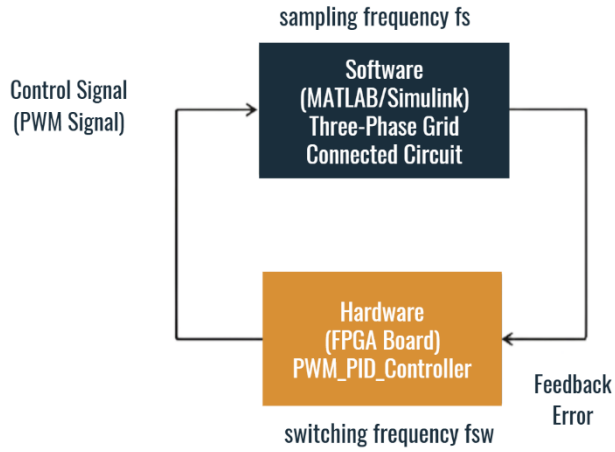

Figure 3.10. Block Diagram for FIL simulation

- Inverter Control on FPGA: Critical tasks like signal transformations, PI control, and PWM generation run on the FPGA to ensure real-time execution.

- Circuit Simulation in Simulink: The inverter switching behavior, LCL filter dynamics, and grid interface are simulated in Simulink to mimic the physical power circuit.

- Feedback Loop: Synchronization between FPGA and Simulink enables the exchange of feedback signals and control outputs, maintaining a closed-loop operation.

FPGA board components

The **Nexys A7-100T** is a board for development designed by **Digilent** incorporating the **Xilinx Artix-7 XC7A100T-1-CSG324 FPGA**, provides extensive computational resources, peripherals, and efficient memory for real-time applications. Its powerful DSP slices, GPIO capabilities, and robust connectivity interfaces make it ideal for implementing and validating control systems, especially when combined with tools like **MATLAB/Simulink** for simulation and hardware deployment.

The key components of the board are based on the reference-manual [96]:

- A 100 MHz crystal oscillator provides the master clock for the FPGA.
- DDR2 SDRAM: 128 MB of synchronous dynamic RAM for storing and accessing large datasets.
- USB-JTAG Port: Facilitates programming and debugging of the FPGA using Xilinx Vivado or ISE software.
- USB-UART Port: Enables serial communication with external systems, such as a host PC running MATLAB/Simulink.
- Pmod Connectors: Four Pmod interfaces for connecting peripheral devices like sensors, ADCs, and actuators.
- XADC (Analog-to-Digital Converter) embedded in the FPGA provides a twelve-bit ADC with up to 1 MSPS sample rate.

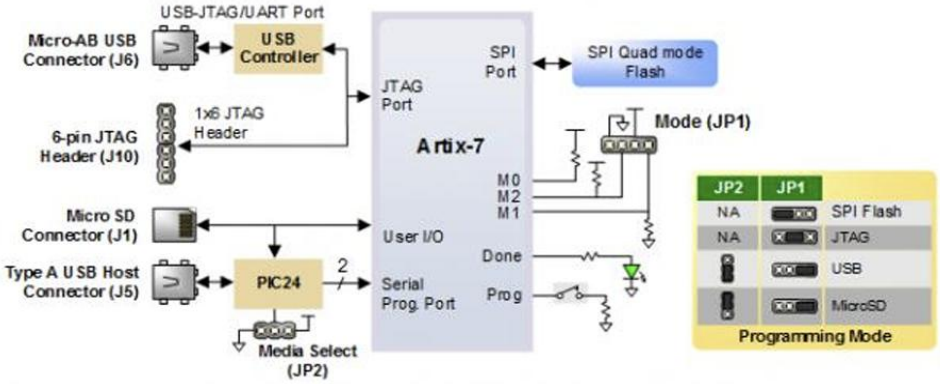

Figure 3.11. Options to configure FPGA [96]

The computer commonly communicates with FPGAs via the JTAG interface. To program JTAG, a bitstream-file with extension **.bit** is transmitted throughout the personal development computer, will be referred as PC, to board of FPGA via either the onboard **Digilent USB-JTAG** (port J6) or an external programmer board designated for JTAG, like the **Digilent JTAG-HS2**, connected to port coded J10.

The available resources of Artix-7 FPGA are [96]:

- 15,850 Programmable Logic Slices, each with four (4) six (6) input LUTs and eight (8) flip-flops (x 8,150 slices)
- 4,860 Kbits of fast block RAM (x 2,700 Kbits)
- Six clock management tiles, each with phase locked loop (PLL)
- 240 DSP slices (x 120 DSPs)
- Internal clock speeds exceeding 450 MHz
- Dual-channel, 1 MSPS internal analog to digital converter (XADC)

Sampling and Switching Frequencies

The correct configuration of sampling frequency (f_s) and the switching frequency (f_{sw}) is crucial for stable and accurate control:

Switching frequency (f_{sw}):

- Determines the frequency of PWM signal generation.
- Higher f_{sw} reduces output voltage harmonics but increases switching losses.

Sampling Frequency (f_s):

- Controls the rate at which the control algorithm updates signals (e.g., duty cycles).
- The sampling frequency must satisfy:

$$f_s \geq n \cdot f_{sw} \qquad [95]$$

The relationship between f_{sw} and f_s directly impacts system performance:

- Higher f_{sw} reduces output voltage/current harmonics and improves power quality but increases switching losses.
- Higher f_s ensures the control algorithm updates before each switching event and enhances real-time response and stability but increases computational requirements.

In Simulink, the sampling and switching frequencies are configured as follows:

- PWM Generation uses a Repeating Sequence block to set f_{sw} of the inverter. For example, if f_{sw} = 100kHz, the block generates carrier waves at this frequency. Before HDL coder, switching frequency is defined in this block:

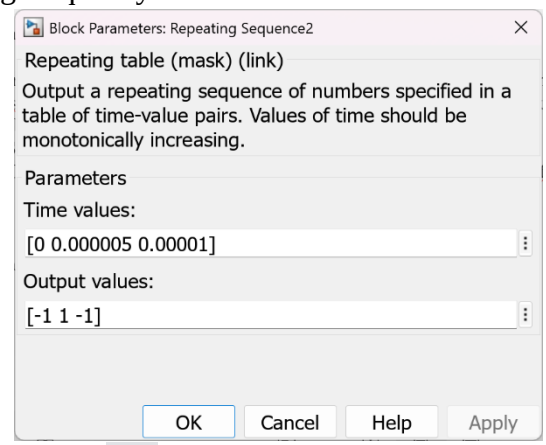

Figure 3.12. Setting Switching Frequency

- Sampling Frequency is set in Model Settings → Configuration Parameters → Solver:

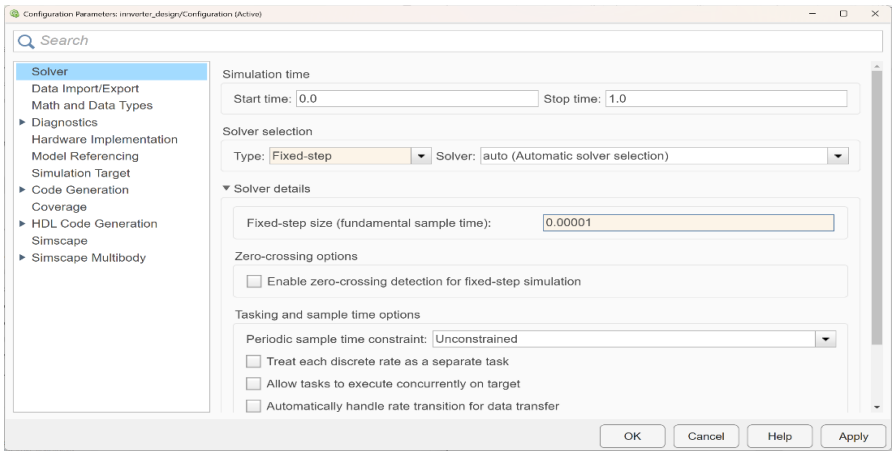

Figure 3.13. Setting Sampling Frequency

Hardware Deployment and Execution

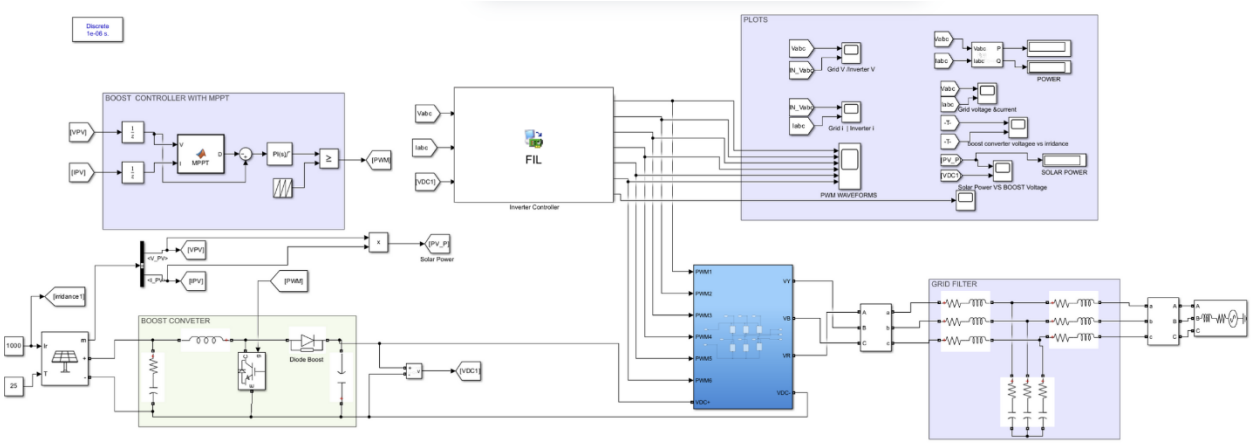

Figure 3.14. FIL Simulation

We conduct the FIL simulation within the MATLAB/Simulink environment, targeting the Nexys A7-100T FPGA, which features the Artix-7 XC7A100T-1-CSG324 device. The process of deploying the control algorithm to FPGA hardware consists of the following steps:

1. Since FPGA hardware is resource-constrained, fixed-point arithmetic is preferred over floating-point for computational efficiency. We replace all floating-point computations with fixed-point equivalents for efficient FPGA resource utilization.

2. HDL Coder is used to convert the fixed-point Simulink model into synthesizable HDL code – Verilog.

3. In HDL Workflow Advisor, FPGA hardware (Artix-7 XC7A100T) is selected, and input/output signals are mapped to FPGA pins.

4. FPGA is connected to the host computer via USB interface and HDL code is loaded onto the FPGA using tools FIL Wizard.

5. The generated FIL Block is used in Simulink to establish communication between the FPGA and Simulink:

    - Simulink sends measurements to the FPGA.
    - FPGA computes the control algorithm and returns the results to Simulink.

6. The simulation is started, and the real time performance for the controlled system is measured.

Execution Time Measurement

Execution time implies the time taken for FPGA to do computation one iteration of control algorithm. It ensures that the control computation is completed within the sampling period.

Execution Time Analysis can be done using Simulink Profiler:

1. Enable the Simulink Profiler under the Debug tab.
2. Run the simulation to collect execution time data for each block.
3. Analyze the results:
    - Ensure the execution time is less than the sampling period:
    - If the execution time exceeds Ts, optimize the control algorithm by reducing complexity or refining fixed-point settings.

The Profiler report provides a detailed analysis of the time needed to run each block, allowing to determine where the calculation is limited.

Resource Utilization

Resource usage - the total amount of hardware resources that are consumed, can be determined as the designed inverter control scheme is deployed to FPGA. These include logic elements, memory blocks, DSP-slices, flip-flops, and I/O pins which are key components on the FPGA. As a result, it is possible to analyze the degree to which the FPGA components are used effectively by the algorithm or designed control system that has been implemented. Therefore, this is an essential metric for the research as analysis result is used to decide whether or not the design can be deployed & run on the FPGA by adhering to the timing constraints and preserving the computation-efficiency.

Tools provided by FPGA manufacturers, such as Xilinx Vivado, and Simulink HDL Coder application allow to ascertain the consumption of resources. The report can be accessed through the Synthesis and Analysis step of the HDL Workflow Advisor.

By doing this study, we can be confident the design is practical and will work well in the actual world. Excessive resource utilization triggers iterative improvements that make logic simple, share hardware, or optimize fixed-point arithmetic. Power system (FIL) simulation is both efficient and accurate when resource consumption is balanced with execution time.

Processor-in-the-Loop Approach

The effectiveness of control algorithms in real-time can be validated through the use of PIL simulation in MATLAB/Simulink by executing the algorithms on the external target hardware. In order to complete the method, we configure the Simulink model, generate the PIL block, and evaluate execution performance. When it comes to ensuring that the control system functions effectively, processing time and resource usage are two of the most important measures. Through careful adjustment of the sample frequency and thorough data examination, the system is able to fulfill the real-time criteria for the implementation of trustworthy hardware. In order to validate system functionality before completing hardware deployment, PIL is essential. This helps to reduce errors, and the amount of time needed for development.

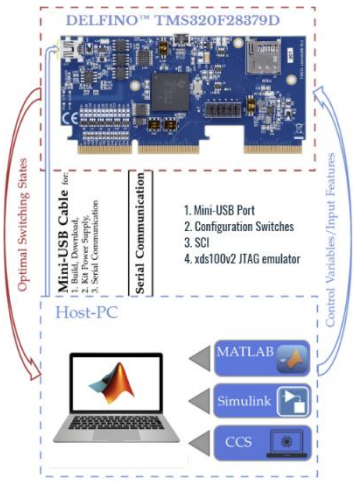

Figure 3.15. Physical Arrangement for PIL simulation

In the figure above (Figure 3.15), two key components communicate with each other:

1. Software (Simulink): Simulates the power circuit and provides inputs to the controller.
2. Hardware (Target Microcontroller): Runs the control algorithm (PWM control) and generates control outputs.

The feedback loop ensures synchronization between software and hardware, where the control signal (PWM) is updated based on real-time feedback errors.

Microcontroller Description

The TMS320F28379D microcontroller from the Texas Instruments Delfino family is a high-performance DSP with an integrated microcontroller. Designed for real-time control applications, it is particularly suitable for power electronics, motor control, and signal processing due to its powerful computing capabilities, peripheral interfaces, and extensive memory resources.

According to the reference-manual [97], the key components are:

- Dual C28x CPU cores operating at 200 MHz for parallel processing.
- Flash Memory: 1 MB (organized into 16 sectors) for program storage.
- SRAM: 164 KB on-chip RAM for fast data access.
- ROM: Bootloader and system initialization code are preloaded in the on-chip ROM.
- Four twelve (12) bit ADC modules withup to 3 MSPS sample rate for signal acquisition.
- Serial Communication: SCI (UART), SPI, I2C, and CAN for interfacing with external devices.

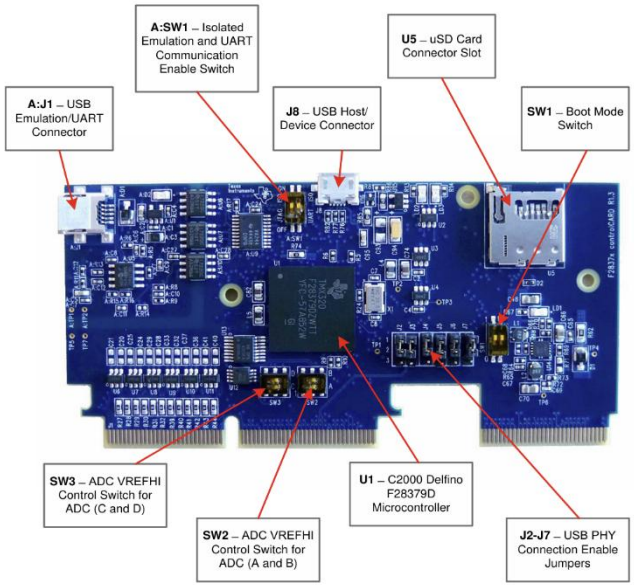

Figure 3.16. Main components of controlCARD [97]

Procedure for PIL Simulation

1. A subsystem is created for the control algorithm, which will be converted into the PIL block. Then, all data types are converted in the control subsystem to single precision.

2. The simulation solver is switched from continuous-time to discrete-time in Model Configuration Parameters → Solver → Fixed-Step.

3. **Mini-USB cable is used** to connect the target hardware to the host PC. The TMS320F28379D communicates with the host PC through the **SCI**. uses GPIO pins (specifically GPIO-72 and GPIO-84) to determine the boot mode. These pins are configured via the SW1 jumper on the control CARD, which specifies the source from which the microcontroller fetches its initial instructions. The boot mode selection is summarized in the table below:

Table 3.2. Boot Mode Switch SW1 [97]

Mode #	Switch Position 1 (GPIO-72)	Switch Position 2 (GPIO-84)	Boot from
00	0	0	Parallel I/O
01	0	1	Boot from SCI
02	1	0	Wait Boot Mode
03	**1**	**1**	**Get Mode (Flash by default)**

For PIL simulation, 2 modes are typically used [97]:

- Boot from SCI (Mode 01): Enables serial communication boot loading, where instructions are transmitted via the SCI interface from an external host (e.g., Simulink/CCS).
- Flash Boot Mode (Mode 03): Executes preloaded instructions from the on-chip Flash memory. This mode is typically used for standalone execution of the control algorithm.

4. A PIL block is automatically generated using Deploy to Hardware → Deploy Selected Subsystem to Hardware and replaces the original Control Subsystem. Simulink generates a Code Generation Report, which includes the generated C code. Original control subsystem is replaced with the generated PIL block.

5. Simulink communicates with the MCU using the SCI connection to send/receive control signals, variables, and input/output data.

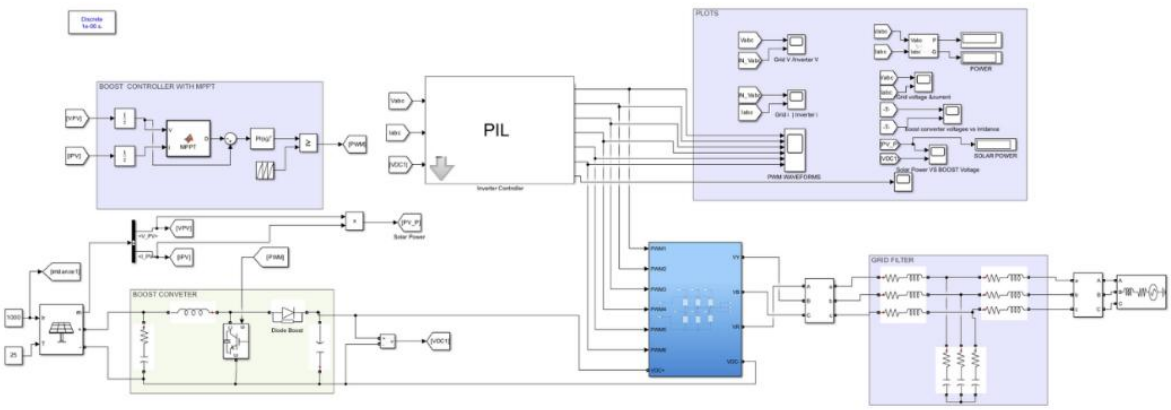

Figure 3.17. PIL Simulation

Execution Time

Execution time measures the time taken for the target controller to execute the control algorithm.

1. Enable "Measure Task Execution Time" in the Model Configuration Parameters → Code Generation → Verification.

2. View the task execution time in the Diagnostic Viewer or the Report generated by Simulink.

Resource Utilization

When implementing a control algorithm on the microcontroller, it is essential to evaluate the memory and hardware resource utilization.

- Flash Memory Usage: Indicates the percentage of program memory consumed by the control code.
- SRAM Usage: Reflects the real-time memory requirements for variable storage and computation.
- CPU Load: Monitors the percentage of processor resources utilized during execution.

Code Composer Studio (CCS) and Simulink generate reports on Flash, RAM, and CPU usage during code compilation.

CHAPTER FOUR
RESEARCH RESULTS AND ANALYSIS OF RESULTS

The Voltage and Grid plots are compared for $f_{sw} = 100\ and\ f_s = 10\ ,20\ ,100\ ,200\ kHz$

- **FIL Simulation**

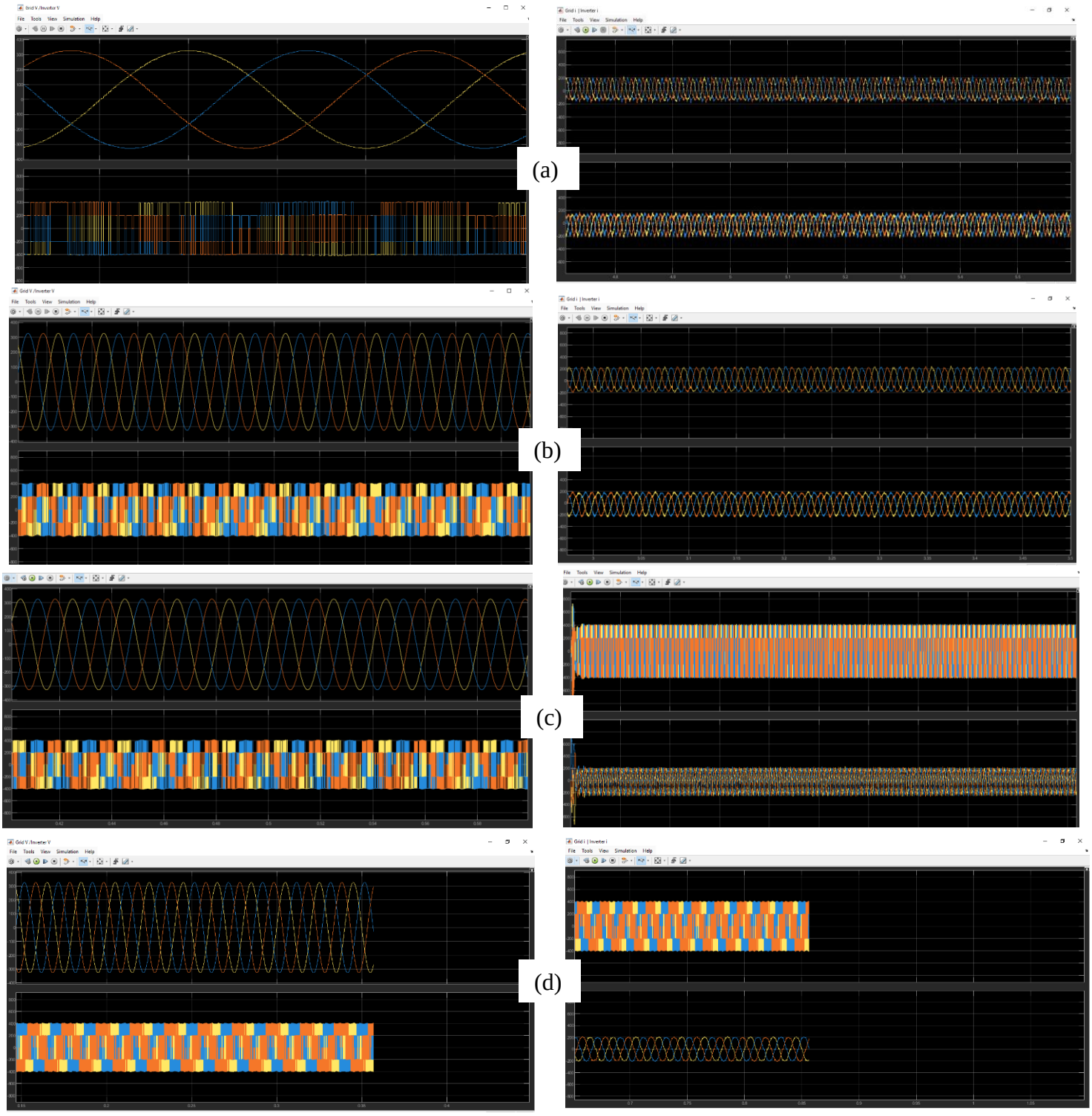

Figure 4.1. Grid/Inverter Voltage and Current plots in FIL simulation for different sampling frequencies (a) fs = 10 kHz, (b) fs = 20 kHz, (c) fs =100 kHz, (d) fs = 200 kHz

- **PIL Simulation**

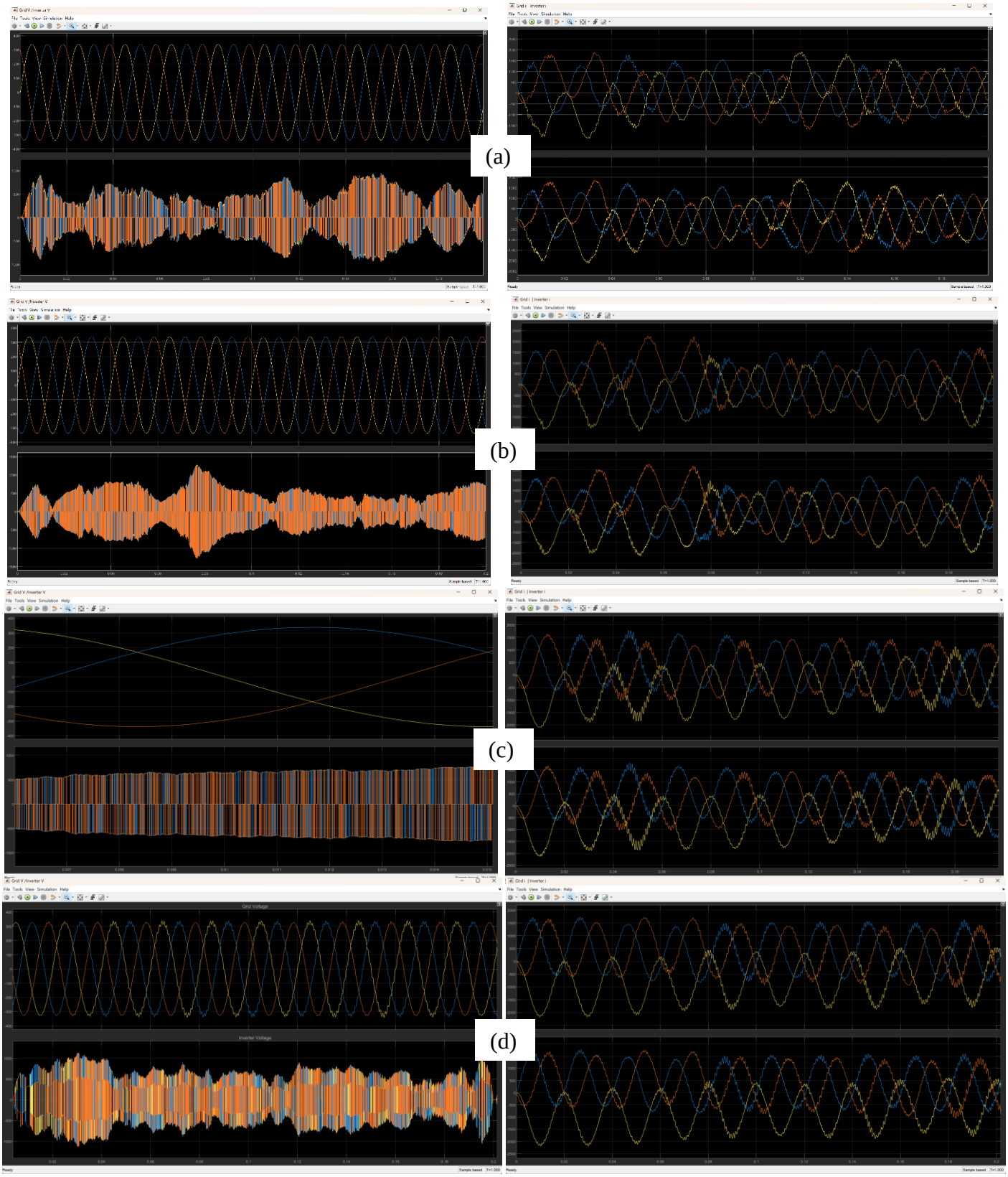

Figure 4.2. Grid/Inverter Voltage and Current plots in PIL simulation for different sampling frequencies (a) fs = 10 kHz, (b) fs = 20 kHz, (c) fs =100 kHz (d) fs = 200 kHz

The FIL simulation results demonstrate significant differences in the behavior of grid and inverter voltage and current as the sampling frequency increases. At fs=10 kHz, the waveforms exhibit noticeable distortions and irregularities due to inadequate sampling resolution, leading to inaccurate reproduction of control signals. As the sampling frequency increases to fs=20 kHz, the waveforms improve, showing smoother transitions and better tracking of the desired sinusoidal shape. At fs=100 kHz and 200 kHz, the performance further enhances, with minimal distortion and almost ideal sinusoidal waveforms. In the PIL simulation, similar trends are observed as in the FIL.

When comparing the PIL and FIL simulation results (Figure 4.1 and 4.2), differences in the accuracy and noise levels are evident. FIL simulations generally produce smoother waveforms with less noise, as the FPGA's real-time processing handles feedback and control with high precision. In contrast, PIL simulations, which involve communication delays and processing within the microcontroller, introduce minor distortions and oscillations, particularly at lower sampling frequencies.

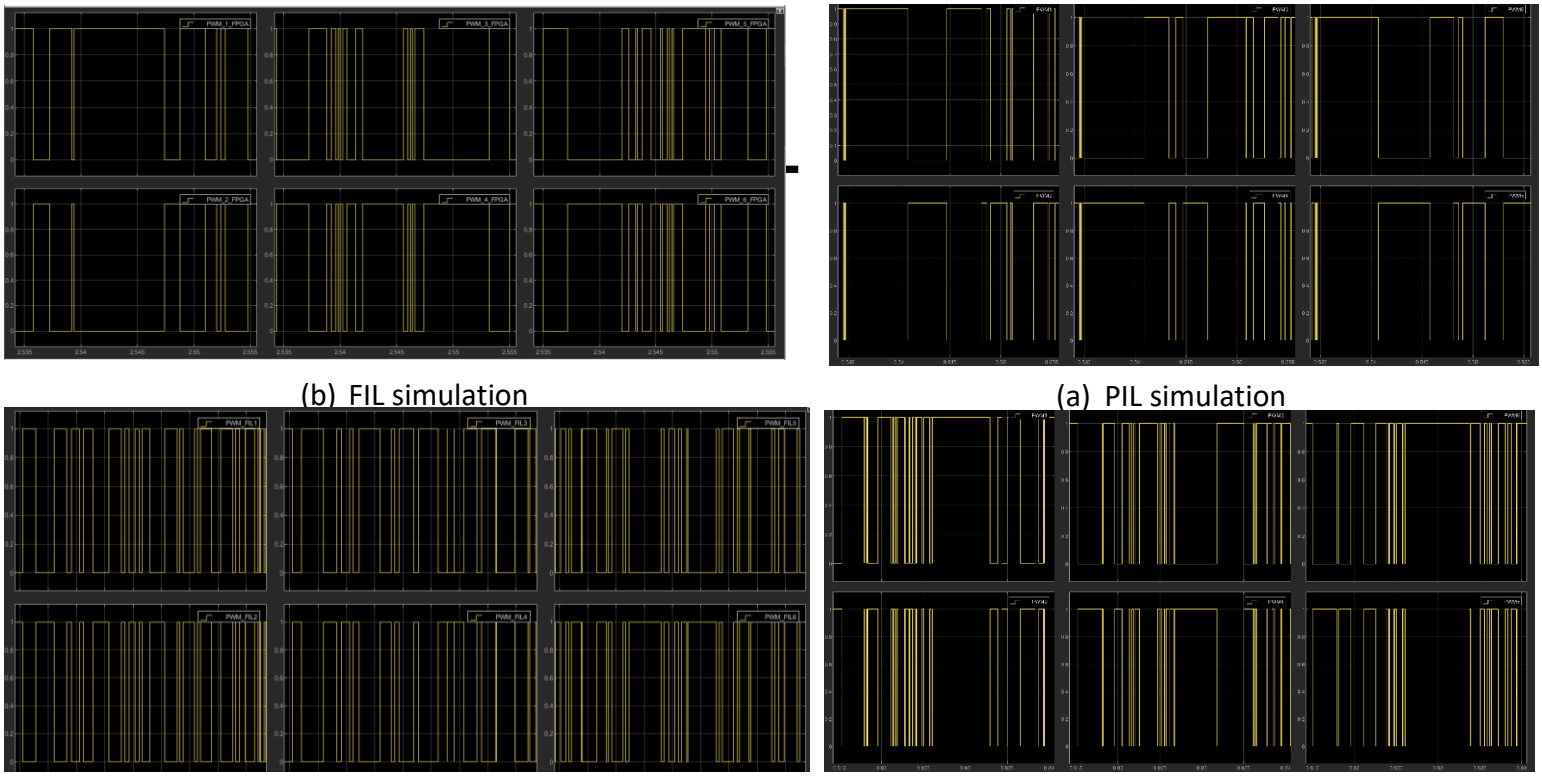

Figure 4.3. PWM signals of (a) FIL and (b) PIL simulation at frequencies fs = 10 kHz and 20 kHz

A comparison of the PWM signals produced by the PIL and FIL simulations is shown in Figure 4.3, with the sampling frequencies being 10 kHz and 20 kHz. When the frequency is set to 10 kHz, the PIL simulation displays PWM signal patterns that are more clearly defined and steady, on the other hand, the FIL simulation displays higher fidelity with decreased distortions as a result of the exact timing capabilities of the FPGA technology. The PIL simulation at 20 kHz shows a slight response lag because of the microcontroller's constrained processing power and slower execution speed. As a result, the frequency of PWM duty cycle changes. However, FIL simulation maintains consistent signal patterns, demonstrating the FPGA's ability to manage greater sampling rates with reduced delay.

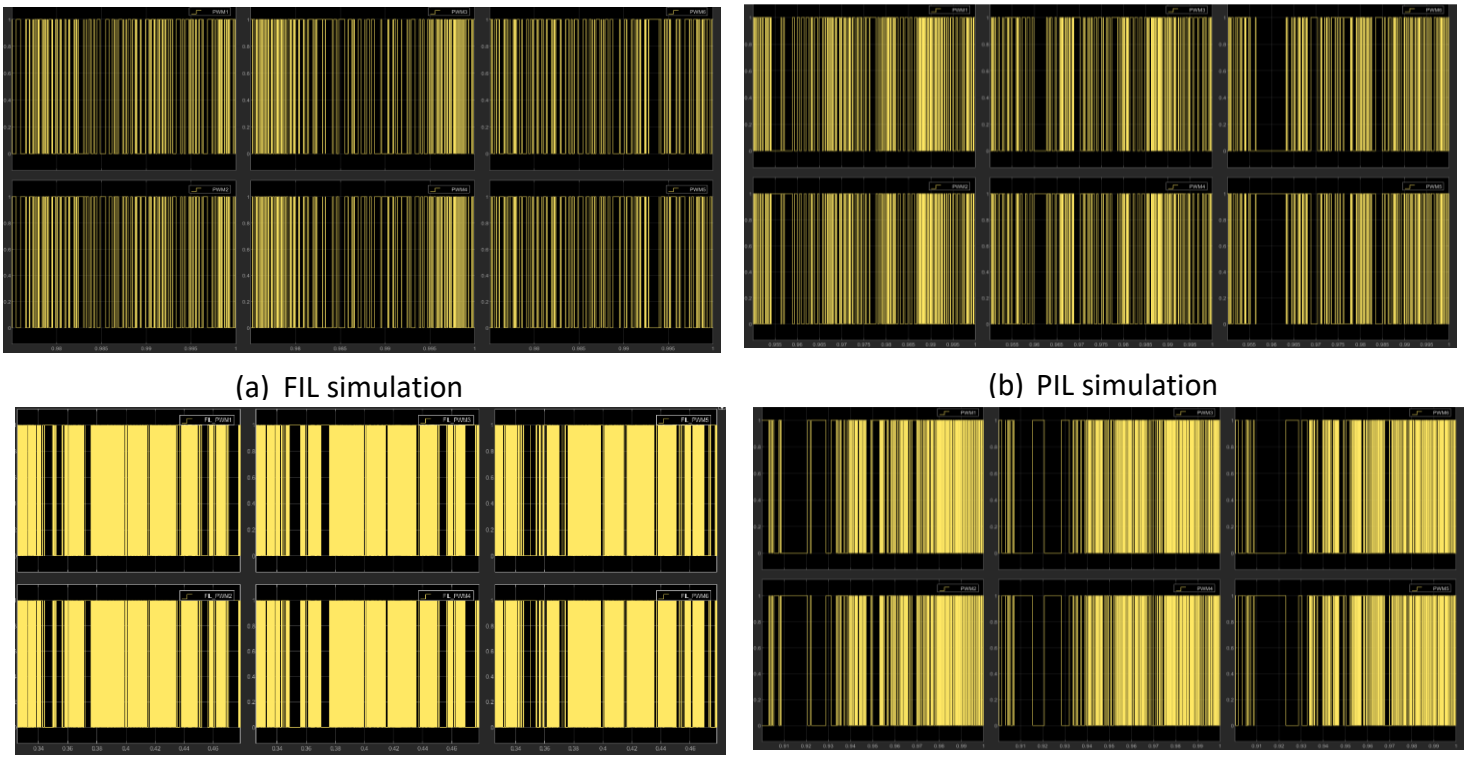

Figure 4.4. PWM signals of (a) FIL and (b) PIL simulation at frequencies fs = 100 kHz and 200 kHz

As can be seen from Figure 4.4, the PWM duty cycles at faster sampling frequencies - 100 kHz and 200 kHz are the most prominent. Due to the MCU's constrained processing capacity and other limitations, PIL simulation starts to show significant faults and fails to maintain perfect timing. On the other hand, FIL simulation effectively use advantage of the FPGA's rapid processing & execution and parallel computing features for the generation of accurate and exact PWM patterns. The MCU's inability to manage frequencies of this scale can be seen from the PIL simulation's inability to produce reliable PWM signals at 200 kHz while FIL simulation shows that the FPGA can withstand very high switching & sampling frequencies without affecting the signal integrity, proving that it is durable.

Efficiency

Table 4.1. System Efficiency of FIL & PIL simulation at different sampling frequencies

Sampling frequency (Hz)	Efficiency – PIL (%)	Efficiency – FIL (%)
10k	88.12	90.07
20k	90.34	92.44
100k	90.63	92.69
200k	93.12	95.07

A comparison of the effectiveness of PIL and FIL simulations is presented in Table 4.1, which summarizes the results for a variety of sample frequencies. Despite the fact that FIL consistently achieves better efficiency, both of them demonstrate that efficiency increases with increasing sample frequency. It appears that FIL is superior to PIL when it comes to managing lower frequencies, as the PIL simulation barely reaches 88.12% at 10 kHz, whilst FIL achieves a rate of 90.07% at this frequency. Despite the fact that FIL is able to achieve 95.07% at 200 kHz, PIL is only able to accomplish 93.12% at that frequency. As a result, it has been proved that FPGA is able to perform well because of its excellent computing speed and accuracy.

In conclusion, the system's energy conversion capability can be highlighted as the FPGA's increased computational efficiency and precision results in reduced losses and higher efficiency at all sampling frequencies.

Timing Analysis and Resource Utilization

- **PIL Simulation**

Table 4.2. Timing Analysis of PIL Simulation for different frequencies

Sampling Frequency	Task	Max. Exec. Time (ns)	Avg. Exec.Time (ns)	Call	Total (ms)
10 kHz	ControlBlock_initialize	98555	98555	1	729.28
	ControlBlock_step	73065	72920.359	10001	
	ControlBlock_terminate	360	360	1	
20 kHz	ControlBlock_initialize	98555	98555	1	1457.76
	ControlBlock_step	73065	72884.569	20001	
	ControlBlock_terminate	360	360	1	
100 kHz	ControlBlock_initialize	97790	97790	1	6634.59
	ControlBlock_step	66480	66345.22	100001	
	ControlBlock_terminate	95	95	1	
200 kHz	ControlBlock_initialize	98555	98555	1	14574.64
	ControlBlock_step	73065	72872.826	200001	
	ControlBlock_terminate	360	360	1	

Table 4.3. Resource Utilization of PIL Simulation for different frequencies

Sampling Frequency	Average CPU Utilization (%)	Maximum CPU Utilization (%)
10 kHz	72.92	73.06
20 kHz	145.8	146.1
100 kHz	663.5	664.8
200 kHz	1457	1461

Table 4.2 presents the execution times for the PIL simulation across different sampling frequencies (10kHz, 20kHz, 100kHz, 200kHz). As the sampling frequency increases, the number of calls to the ControlBlock_step function proportionally increases, resulting in higher cumulative execution times. For instance, at 10 kHz, the total execution time for ControlBlock_step is 729.28 ms, while at 200 kHz, it rises significantly to 14,574.64 ms. The initialization and termination tasks exhibit fixed execution times, independent of the sampling frequency, at 98,555 ns and 360 ns, respectively. This consistent behavior underscores the scalability of the initialization and termination functions, while the significant rise in cumulative execution time at higher frequencies emphasizes the computational demand placed on the system at higher sampling rates.

Table 4.3 shows that the average and maximum CPU utilizations remain relatively moderate at lower sampling frequencies, such as 10 kHz and 20 kHz (e.g., 72.92% and 73.06% at 10 kHz). However, as the sample frequency goes up till 100kHz and 200kHz, resource utilization grows dramatically, with a maximum CPU utilization reaching 1,461.00 % at 200 kHz.

(a)

Check Description	Results
Overall average CPU utilization lower than threshold (70%)	Failed
Overall maximum CPU utilization lower than threshold (70%)	Failed
All average execution times are shorter than their task period	Passed
All maximum execution times are shorter than their task period	Passed

(b)

Check Description	Results
Overall average CPU utilization lower than threshold (70%)	Failed
Overall maximum CPU utilization lower than threshold (70%)	Failed
All average execution times are shorter than their task period	Failed
All maximum execution times are shorter than their task period	Failed

Figure 4.5. Feasibility Check of PIL Simulation for (a) fs=10 kHz, (b) fs=20,100,200kHz

The feasibility check results, as summarized in Figure 22, evaluate the performance of the PIL simulation based on CPU utilization and execution time constraints. For sampling frequencies of 10kHz, 20kHz, 100kHz,200kHz, the overall average and maximum CPU utilizations consistently exceed the 70% threshold, resulting in failed feasibility checks for resource usage at all frequencies. Despite this, the execution times—both average and maximum—remain within the bounds of their respective task periods for lower frequencies, ensuring timely operation. However, at higher frequencies, such as 200 kHz, the combination of high CPU utilization and increasing execution demands leads to feasibility failures in all categories. This underscores the challenge of balancing computational efficiency and resource limitations when scaling sampling frequencies, highlighting the need for either hardware upgrades or optimized algorithms to ensure reliable real-time performance.

- **FIL Simulation**

Table 4.4. Timing Analysis of FIL Simulation for different frequencies

Sampling Frequency	Task	Propagation (ns)	Delay (ns)	Calls	Total (ms)
10 kHz	FILBlock	1618.533	99.101	100005	171.772
20 kHz	FILBlock	1850.746	99.101	200005	389.979
100 kHz	FILBlock	1850.746	99.101	1000005	1949.857
200 kHz	FILBlock	1819.198	99.101	2000005	3836.608

The timing analysis of the FIL simulation for different sampling frequencies, as outlined in Table 4.4, highlights a significant relationship between sampling frequency and overall simulation time. At 10 kHz sampling frequency, the total execution time for the FIL block is 171.772 ms, which increases progressively as the sampling frequency rises. For instance, at 20 kHz, the total time reaches 389.979 ms, and at 100 kHz, it escalates further to 1949.857 ms. Finally, at 200 kHz, the total execution time peaks at 3836.608 ms. The increase in total simulation time with higher frequencies is due to the increased number of calls to the FIL block, which demands more computational effort per sampling period. Both propagation time and delay remain consistent across frequencies, indicating that the variations in total execution time are primarily influenced by the number of iterations performed rather than inherent block processing delays.

Table 4.5. Slice Logic

Site-Type	Used	Fixed	Avail.	Util. %
Slice LUTs*	15371	0	63400	24.24
LUT Logic	14888	0	63400	23.48
LUT Memory	483	0	19000	2.54
LUT Distributed RAM	0	0		
LUT Shift Register	483	0		
Slice Register	10950	0	126800	8.64
Register Flip Flop	10950	0	126800	8.64
Registers Latch	0	0	126800	0
F7 Muxes	1020	0	31700	3.22
F8 Muxes	278	0	15850	1.75

Table 4.6. Resource utilization for FIL simulation

Site Type	Used	Fixed	Available	Util%
Block RAM Tile	1	0	135	0.74
RAMB36/FIFO*	0	0	135	0
RAMB18	2	0	270	0.74
RAMB18E1 only	2			
DSPs	166	0	240	69.17
DSP48E1 only	166			

The resource utilization for FIL simulation, as illustrated in Tables 4.5 and 4.6, emphasizes the efficiency and physical resource demands of the implementation. In terms of Slice Logic (Table 4.5), the design uses 24.24% of the Slice LUTs availability and 8.64%Slice Registers, demonstrating a moderate resource demand. LUTs used for logic dominate the utilization at 23.48%, while those used as memory contribute minimally at 2.54%. The utilization of F7 and F8 multiplexers remains low at 3.22% and 1.75%, respectively. For memory and DSP resources (Table 4.6), only 0.74% of the available block RAM tiles are used, reflecting minimal memory requirements. However, the DSP utilization is higher, with 166 out of 240 DSP48E1 blocks used, corresponding to 69.17% utilization. This indicates that while the design is computationally intensive, it effectively balances its hardware resources without exceeding critical thresholds.

CHAPTER FIVE
DISCUSSION AND CONCLUSION

In this paper, a complete investigation into the design, modeling, and performance assessment of FPGA for GaN-based 3-phase inverters is carried out. Through the utilization of PIL and FIL simulations, our research aims to demonstrate the advantages of FPGA designs in comparison to conventional microcontroller-based programs. PIL simulation performed on the TMS320F28379D MCU demonstrates the constraints that sequential execution and computing resources impose on systems that are based on MCU programming. FIL simulations, on the other hand, make use of the simultaneous processing features of FPGAs in order to obtain greater performance and demonstrate the potential of FPGAs to address the issues that are associated with high-frequency inverter control.

In order to eliminate the issues of faster-frequency switching and efficient power conversion, we build and analyze inverter-control techniques for this type of inverters that are based on FPGAs and MCUs. PIL & FIL simulations are going to be done and evaluated as part of this work. The purpose of this work is to evaluate the performance of both control systems under different sampling frequencies. The development of control algorithms, the use of fixed-point arithmetic to transform them into formats that are compatible with hardware, and the deployment of these algorithms on the FPGA and MCU technologies for real-time performance are all essential activities.

We conduct timing assessments and resource allocation metrics to check whether systems are feasible at various frequencies or not. The advantages of FPGAs over conventional MCUs in the high-frequency applications were demonstrated through assessments of PWM signal quality, efficiency, running time and other factors. Based on research finding, FPGA is an excellent choice for having accurate regulation and high-reliability for contemporary power electronic systems because of its features of simultaneous processing and its improved resource efficiency.

In order to achieve accurate regulation in power electronic systems, especially GaN based systems, it is essential to have faster sampling than the switching frequency. The control system is able to adjust PWM signals and conduct control techniques many times throughout each switching period when it is programmed to use a higher sampling frequency. In addition to reducing distortion and increasing efficiency, this ensures that waveform tracking is correct. When the sample frequencies are less than switching, on the other hand, there is a delay in the updates, which ultimately results in inconsistencies with the switching states. It is tested by research that it leads to harmonic distortion, efficiency reduction, instability, PWM signal quality degradation and so on. The FPGA-based system is an alternative to the conventional MCU-based systems that we suggest as a solution to the problems that have been stated above.

We found that FPGA-based controllers can handle higher switching frequencies and improve efficacy, trustworthiness, and scalability. Because of these advantages, GaN-based inverters are appropriate for use in modern power conversion applications including sustainable renewable energy systems and battery-powered electric cars. Although they have its limitations, such as its simulation settings and hardware shortage, the study provides the framework for the future research. This consists of real-world evaluation and FPGA layout improvements for complex control algorithms to develop power electronics technology.

Comparison

The timing assessment of PIL and FIL simulations varies significantly due to different hardware and execution methods: PIL simulations at higher sample rates require more time on the MCU because of its limited processing capabilities, as evidenced by the length of the ControlBlock_step work at 200 kHz, which is 14,574.64 ms, indicating the lengthy execution and increased calls of this conventional microcontroller; FIL simulations, on the other hand, are more computationally efficient and demand a shorter period to execute at the same sample rates because the FPGA can perform multiple processes simultaneously. At 200 kHz, the FILBlock functioning takes only 3836.608 ms, substantially faster than PIL.

Models made with PIL and FIL use assets in very different ways since microcontrollers and FPGAs are assembled in in a variety of ways. When the frequency is sped up, PIL models have trouble finding resources. The highest CPU usage is more than 100% at 100 kHz (664.8%) and 200 kHz (1461%), so the tests can't happen. FIL models, on the other hand, create fair ways to use resources. It shows that the FPGA implementation can handle jobs that need a lot of computing power while continuing to stay within acceptable utilization limits by using 24.24% of the available Slice LUTs and 69.17% of the DSP resources. Quick and real-time apps work better with FIL because it makes better use of resources and can cope with higher rates. This is not the case with PIL models; the microcontroller sets the limits for them.

Future Research Directions

There are several directions for the further future that can be suggested:

1. Real-world testing: It can be done by developing the circuit using passive components, GaN transistors and so on. This helps to ensure the design functions are as intended in real-world scenarios.

2. Real-world challenges: Heat management of GaN-technology, EMI in high-frequency situations, and maintenance of system stability under load & grid circumstances under different load distribution.

3. ML & AI Integration: It is useful to test predictive control systems that integrate AI and ML. They analyze systems in various environments.

REFERENCES

[1] N. Mohan, W. P. Robbins, and T. M. Undeland, Power electronics: converters, applications, and design, 3rd ed. John Wiley & Sons, Inc, 2002.

[2] M. H. Rashid, *Power Electronics: Circuits, devices, and applications*. Prentice Hall, 2011.

[3] B. J. Baliga, *Fundamentals of Power Semiconductor Devices*. Springer, 2018.

[4] D. Reusch and J. Strydom, "Understanding the effect of PCB layout on circuit performance in a High-Frequency Gallium-Nitride-Based Point of Load converter," *IEEE Transactions on Power Electronics*, vol. 29, no. 4, pp. 2008–2015, Jun. 2013, doi: 10.1109/tpel.2013.2266103.

[5] A. Lidow, J. Strydom, M. De Rooij, and D. Reusch, *GAN transistors for efficient power conversion*. John Wiley & Sons, pp. 1018, 2014.

[6] U. K. Mishra, P. Parikh, and N. Y.-F. Wu, "AlGaN/GaN HEMTs-an overview of device operation and applications," *Proceedings of the IEEE*, vol. 90, no. 6, pp. 1022–1031, Jun. 2002, doi: 10.1109/jproc.2002.1021567.

[7] L. Corradini, D. Maksimović, P. Mattavelli, and R. Zane, *Digital control of High-Frequency Switched-Mode power converters*. 2015. doi: 10.1002/9781119025498.

[8] F. Blaabjerg, R. Teodorescu, M. Liserre, and A. V. Timbus, "Overview of control and grid synchronization for distributed power generation systems," *IEEE Transactions on Industrial Electronics*, vol. 53, no. 5, pp. 1398–1409, Oct. 2006, doi: 10.1109/tie.2006.881997.

[9] R. P. Singh, A. M. Khambadkone, G. S. Samudra and Y. C. Liang, "An FPGA based digital control design for high-frequency DC-DC converters," *2006 37th IEEE Power Electronics Specialists Conference*, Jeju, Korea (South), 2006, pp. 1-7, doi: 10.1109/pesc.2006.1712133.

[10] T. Kahl and S. Dieckerhoff, "Comparison of FPGA- and microcontroller-based control of a high-dynamic power electronic converter," 2017 IEEE 18th Workshop on Control and Modeling for Power Electronics (COMPEL), Stanford, CA, USA, 2017, pp. 1-6, doi: 10.1109/COMPEL.2017.8013288.

[11] G. H. M. Tavares, M. L. G. Salmento, W. J. Paula, D. d. C. Pereira and H. A. C. Braga, "Implementation of a high frequency PWM signal in FPGA FOR GaN power devices switching," *2017 Brazilian Power Electronics Conference (COBEP)*, Juiz de Fora, Brazil, 2017, pp. 1-7, doi: 10.1109/COBEP.2017.8257309.

[12] A. Rajalakshmi and A. Kavitha, "Implementation of Low Cost FPGA Based Digital Modulation Techniques for the Suppression of EMI and Improvement of Power Factor for Three-Phase Grid Connected P-V Inverters," *2018 IEEE International Conference on Power Electronics, Drives and Energy Systems (PEDES)*, Chennai, India, 2018, pp. 1-6, doi: 10.1109/PEDES.2018.8707801.

[13] "1-kW, Bidirectional, Three-Phase ANPC Based on GaN Reference Design," Design Guide: TIDA-010210, Texas Instruments, Mar. 2021.

[14] G. Lei, J. Zhu, Y. Guo, C. Liu, and B. Ma, "A review of Design Optimization Methods for Electrical machines," *Energies*, vol. 10, no. 12, p. 1962, Nov. 2017, doi: 10.3390/en10121962.

[15] R. Chen and Y.-Y. Tzou, "Design and implementation of a Three-Phase Active T-Type NPC inverter for Low-Voltage microgrids," Energy and Power Engineering, vol. 09, no. 04, pp. 70–77, Jan. 2017, doi: 10.4236/epe.2017.94b009.

[16] M. Vicente, A. Imperadore, F. X. C. Da Fonseca, M. Vieira, and J. Cândido, "Enhancing islanded Power Systems: Microgrid modeling and evaluating system

benefits of ocean renewable energy integration," *Energies*, vol. 16, no. 22, p. 7517, Nov. 2023, doi: 10.3390/en16227517.

[17] L. Asiminoaei, P. Rodriguez and F. Blaabjerg, "Application of Discontinuous PWMModulation in Active Power Filters," in *IEEE Transactions on Power Electronics*, vol. 23, no. 4, pp. 1692-1706, July 2008, doi: 10.1109/TPEL.2008.924599.

[18] A. Cetin and M. Ermis, "VSC-Based D-STATCOM With Selective Harmonic Elimination," in *IEEE Transactions on Industry Applications*, vol. 45, no. 3, pp. 1000-1015, May-June 2009, doi: 10.1109/TIA.2009.2018926.

[19] D. Grahame Holmes and Thomas A. Lipo, "Pulse Width Modulation for Power Converters: Principles and Practice", in IEEE Press series on power engineering, 2003

[20] J. Holtz, "Pulsewidth modulation-a survey," in *IEEE Transactions on Industrial Electronics*, vol. 39, no. 5, pp. 410-420, Oct. 1992, doi: 10.1109/41.161472.

[21] J. Holtz, "Pulsewidth modulation for electronic power conversion," in *Proceedings of the IEEE*, vol. 82, no. 8, pp. 1194-1214, Aug. 1994, doi: 10.1109/5.301684.

[22] C. Lascu, L. Asiminoaei, I. Boldea and F. Blaabjerg, "High Performance Current Controller for Selective Harmonic Compensation in Active Power Filters," in *IEEE Transactions on Power Electronics*, vol. 22, no. 5, pp. 1826-1835, Sept. 2007, doi: 10.1109/TPEL.2007.904060.

[23] C. Lascu, L. Asiminoaei, I. Boldea and F. Blaabjerg, "Frequency Response Analysis of Current Controllers for Selective Harmonic Compensation in Active Power Filters," in *IEEE Transactions on Industrial Electronics*, vol. 56, no. 2, pp. 337-347, Feb. 2009, doi: 10.1109/TIE.2008.2006953.

[24] Man-Chung Wong, Zheng-Yi Zhao, Ying-Duo Han and Liang-Bing Zhao, "Three-dimensional pulse-width modulation technique in three-level power inverters for three-phase four-wired system," in *IEEE Transactions on Power Electronics*, vol. 16, no. 3, pp. 418-427, May 2001, doi: 10.1109/63.923775.

[25] G. S. Kulothungan, A. Edpuganti, A. K. Rathore, J. Rodriguez and D. Srinivasan, "Hybrid SVM-SOPWM Modulation of Current-Fed Three-level Inverter for High Power Application," in *IEEE Transactions on Industry Applications*, vol. 55, no. 4, pp. 4344-4358, July-Aug 2019, doi: 10.1109/TIA.2019.2912967.

[26] Yining Guo, "Applications of power electronics technology: Advanced inverters", in Journal of Physics: vol. 2649, June 2023, doi: 10.1088/1742-6596/2649/1/012051.

[27] A. Bier, V. S. Nguyen, S. Catellani and J. Martin, "Control of a Two-Stage, Single-Phase Grid-Tied, GaN Based Solar Micro-Inverter," *2020 22nd European Conference on Power Electronics and Applications (EPE'20 ECCE Europe)*, Lyon, France, 2020, pp. P.1-P.10, doi: 10.23919/EPE20ECCEEurope43536.2020.9215822.

[28] N. Bouraoui, "Design and Simulation of a Silicon Carbide MOSFET Inverter for Electric Vehicle Traction Applications using Python," *2023 International Conference on Electrical Engineering and Advanced Technology (ICEEAT)*, Batna, Algeria, 2023, pp. 1-5, doi: 10.1109/ICEEAT60471.2023.10426033.

[29] J. Lu, R. Hou, P. Di Maso and J. Styles, "A GaN/Si Hybrid T-Type Three-Level Configuration for Electric Vehicle Traction Inverter," *2018 IEEE 6th Workshop on Wide Bandgap Power Devices and Applications (WiPDA)*, Atlanta, GA, USA, 2018, pp. 77-81, doi: 10.1109/WiPDA.2018.8569194.

[30] M. Z. Zizoui, B. Tabbache, F. Belkhiri and M. E. H. Benbouzid, "Maximum constant boost control of 9-switch z-source power inverter-based electric vehicles," *2017 5th International Conference on Electrical Engineering - Boumerdes (ICEE-B)*, Boumerdes, Algeria, 2017, pp. 1-6, doi: 10.1109/ICEE-B.2017.8192231.

[31] M. Uğur, H. Saraç and O. Keysan, "Comparison of Inverter Topologies Suited for Integrated Modular Motor Drive Applications," *2018 IEEE 18th International Power Electronics and Motion Control Conference (PEMC)*, Budapest, Hungary, 2018, pp. 524-530, doi: 10.1109/EPEPEMC.2018.8521918.

[32] A. K. Abdelsalam, M. I. Masoud, M. S. Hamad and B. W. Williams, "Modified Indirect Vector Control Technique for Current-Source Induction Motor Drive," in *IEEE Transactions on Industry Applications*, vol. 48, no. 6, pp. 2433-2442, Nov.-Dec. 2012, doi: 10.1109/TIA.2012.2227132.

[33] A. Lidow, J Strydom, M De Rooij, and D Reusch, *GaN transistors for efficient power conversion*. Chichester, West Sussex: John Wiley & Sons Ltd, Cop, p.266, 2014.

[34] S. Ji, S. Zheng, F. Wang and L. M. Tolbert, "Temperature-Dependent Characterization, Modeling, and Switching Speed-Limitation Analysis of Third-Generation 10-kV SiC MOSFET," in *IEEE Transactions on Power Electronics*, vol. 33, no. 5, pp. 4317-4327, May 2018, doi: 10.1109/TPEL.2017.2723601.

[35] X. Huang, F. C. Lee, Q. Li and W. Du, "MHz GaN-based interleaved CRM bi-directional buck/boost converter with coupled inductor," *2015 IEEE Applied Power Electronics Conference and Exposition (APEC)*, Charlotte, NC, USA, 2015, pp. 2075-2082, doi: 10.1109/APEC.2015.7104635.

[36] Z. -L. Zhang, Z. Dong, X. -W. Zou and X. Ren, "A Digital Adaptive Driving Scheme for eGaN HEMTs in VHF Converters," in *IEEE Transactions on Power Electronics*, vol. 32, no. 8, pp. 6197-6205, Aug. 2017, doi: 10.1109/TPEL.2016.2619911.

[37] Z. -L. Zhang, Z. Dong, D. -D. Hu, X. -W. Zou and X. Ren, "Three-Level Gate Drivers for eGaN HEMTs in Resonant Converters," in *IEEE Transactions on Power Electronics*, vol. 32, no. 7, pp. 5527-5538, July 2017, doi: 10.1109/TPEL.2016.2606443.

[38] J. Lu, H. K. Bai, S. Averitt, D. Chen and J. Styles, "An E-mode GaN HEMTs based three-level bidirectional DC/DC converter used in Robert Bosch DC-grid system," *2015 IEEE 3rd Workshop on Wide Bandgap Power Devices and Applications (WiPDA)*, Blacksburg, VA, USA, 2015, pp. 334-340, doi: 10.1109/WiPDA.2015.7369254.

[39] X. Zou, Z. Zhang, Z. Dong, Y. Zhou, X. Ren and Q. Chen, "A 10-MHz eGaN FETs based isolated class-Φ2 DCX," *2016 IEEE Applied Power Electronics Conference and Exposition (APEC)*, Long Beach, CA, USA, 2016, pp. 2518-2524, doi: 10.1109/APEC.2016.7468219.

[40] Y. Uemoto *et al.*, "Gate Injection Transistor (GIT)—A Normally-Off AlGaN/GaN Power Transistor Using Conductivity Modulation," in *IEEE Transactions on Electron Devices*, vol. 54, no. 12, pp. 3393-3399, Dec. 2007, doi: 10.1109/TED.2007.908601.

[41] R. Mitova, R. Ghosh, U. Mhaskar, D. Klikic, M. -X. Wang and A. Dentella, "Investigations of 600-V GaN HEMT and GaN Diode for Power Converter

Applications," in *IEEE Transactions on Power Electronics*, vol. 29, no. 5, pp. 2441-2452, May 2014, doi: 10.1109/TPEL.2013.2286639.

[42] W. Zhang, F. Wang, D. J. Costinett, L. M. Tolbert and B. J. Blalock, "Investigation of Gallium Nitride Devices in High-Frequency LLC Resonant Converters," in *IEEE Transactions on Power Electronics*, vol. 32, no. 1, pp. 571-583, Jan. 2017, doi: 10.1109/TPEL.2016.2528291.

[43] X. Huang, Z. Liu, Q. Li and F. C. Lee, "Evaluation and Application of 600 V GaN HEMT in Cascode Structure," in *IEEE Transactions on Power Electronics*, vol. 29, no. 5, pp. 2453-2461, May 2014, doi: 10.1109/TPEL.2013.2276127.

[44] E. Gurpinar and A. Castellazzi, "Single-Phase T-Type Inverter Performance Benchmark Using Si IGBTs, SiC MOSFETs, and GaN HEMTs," in *IEEE Transactions on Power Electronics*, vol. 31, no. 10, pp. 7148-7160, Oct. 2016, doi: 10.1109/TPEL.2015.2506400.

[45] J. Millán, P. Godignon, X. Perpiñà, A. Pérez-Tomás and J. Rebollo, "A Survey of Wide Bandgap Power Semiconductor Devices," in *IEEE Transactions on Power Electronics*, vol. 29, no. 5, pp. 2155-2163, May 2014, doi: 10.1109/TPEL.2013.2268900.

[46] X. She, A. Q. Huang, Ó. Lucía and B. Ozpineci, "Review of Silicon Carbide Power Devices and Their Applications," in *IEEE Transactions on Industrial Electronics*, vol. 64, no. 10, pp. 8193-8205, Oct. 2017, doi: 10.1109/TIE.2017.2652401.

[47] E. A. Jones, F. F. Wang and D. Costinett, "Review of Commercial GaN Power Devices and GaN-Based Converter Design Challenges," in *IEEE Journal of Emerging and Selected Topics in Power Electronics*, vol. 4, no. 3, pp. 707-719, Sept. 2016, doi: 10.1109/JESTPE.2016.2582685.

[48] H. A. Mantooth, K. Peng, E. Santi and J. L. Hudgins, "Modeling of Wide Bandgap Power Semiconductor Devices—Part I," in *IEEE Transactions on Electron Devices*, vol. 62, no. 2, pp. 423-433, Feb. 2015, doi: 10.1109/TED.2014.2368274.

[49] S. Dutta and A. K. Yadav, "Power Loss Comparison of Si-IGBT SiC-MOSFET And GaN Based 3-level Diode Clamped Inverter fed Induction Motor Drive," *2023 IEEE International Conference on Power Electronics, Smart Grid, and Renewable Energy (PESGRE)*, Trivandrum, India, 2023, pp. 1-6, doi: 10.1109/PESGRE58662.2023.10405187.

[50] "Comparative Analysis of Si- and GaN-Based Single-Phase Transformer-Less PV Grid-Tied Inverter," *Electronics*, vol. 7, no. 3, p. 34, Mar. 2018, doi: https://doi.org/10.3390/electronics7030034.

[51] X. Huang, Z. Liu, Q. Li and F. C. Lee, "Evaluation and Application of 600 V GaN HEMT in Cascode Structure," in *IEEE Transactions on Power Electronics*, vol. 29, no. 5, pp. 2453-2461, May 2014, doi: 10.1109/TPEL.2013.2276127.

[52] Sartori, H.C.; Beltrame, F.; Martins, M.L.; Baggio, J.E.; Pinheiro, J.R. Evaluation of an optimal design for a single-phase boost PFC converter (CCM) considering different magnetic materials core. In Proceedings of the 2013 Brazilian Power Electronics Conference, Gramado, Brazil, 27–31 October 2013; pp. 1304–1310.

[53] Z. Yang, J. Chen, P. Williford and F. Wang, "Cost Comparison Between GaN-based and Si-based 4.5-kW Single-phase Inverters," *2020 IEEE Workshop on Wide*

Bandgap Power Devices and Applications in Asia (WiPDA Asia), Suita, Japan, 2020, pp. 1-6, doi: 10.1109/WiPDAAsia49671.2020.9360282.

[54] F. Hattori, H. Umegami, and M. Yamamoto, “Multi-resonant gate drive circuit of isolating-gate GaN HEMTs for tens of MHz,” IET Circuits, Devices & Systems, vol. 11, no. 3, pp. 261–266, 2017.

[55] G. Zulauf, Z. Tong, J. D. Plummer and J. M. Rivas-Davila, "Active Power Device Selection in High- and Very-High-Frequency Power Converters," in *IEEE Transactions on Power Electronics*, vol. 34, no. 7, pp. 6818-6833, July 2019, doi: 10.1109/TPEL.2018.2874420.

[56] J. Lautner and B. Piepenbreier, "Analysis of GaN HEMT switching behavior," *2015 9th International Conference on Power Electronics and ECCE Asia (ICPE-ECCE Asia)*, Seoul, Korea (South), 2015, pp. 567-574, doi: 10.1109/ICPE.2015.7167840.

[57] J. L. Bastos, H. P. Figueroa and A. Monti, “FPGA implementation of neural network-based controllers for power electronics applications,” Twenty-First Annual IEEE Applied Power Electronics Conference and Exposition, APEC pp. 1-6, doi: 10.1109/APEC.2006.1620729.

[58] E. T. Mekonnen, J. Katcha and M. Parker, “An FPGA-based digital control development method for power electronics,” IECON 38th Annual Conference on IEEE Industrial Electronics Society, 2012, pp. 222-226, doi: 10.1109/IECON.2012.6388804.

[59] P. Le-Huy, S. Guerette, L. A. Dessaint and H. Le-Huy, “Real-Time Simulation of Power Electronics in Power Systems using an FPGA,” Canadian Conference on Electrical and Computer Engineering, pp. 873-877, doi: 10.1109/CCECE.2006.277356.

[60] E. Monmasson, L. Idkhajine, I. Bahri, M. Naouar and L. Charaabi, “Design methodology and FPGA-based controllers for Power Electronics and drive applications,” 5th IEEE Conference on Industrial Electronics and Applications, 2010, pp. 2328-2338, doi: 10.1109/ICIEA.2010.5515585.

[61] M. V. Chung, D. T. Anh and P. Vu, “A finite set-model predictive control based on FPGA flatform for eleven-level cascaded H Bridge inverter fed induction motor drive,” International Journal of Power Electronics and Drive System (IJPEDS), vol. 12, no. 2, pp. 845-856, 2021.

[62] W. Zhao, B. H. Kim, A. C. Larson and R. M. Voyles, “FPGA implementation of closed-loop control system for small-scale robot,” ICAR '05. Proceedings, 12th International Conference on Advanced Robotics, 2005, pp. 70-77, doi: 10.1109/ICAR.2005.1507393.

[63] Y. P. Siwakoti and G. E. Town, “Design of FPGA-controlled power electronics and drives using MATLAB Simulink,” IEEE ECCE Asia Downunder, 2013, pp. 571-577, doi: 10.1109/ECCE-Asia.2013.6579155.

[64] K. Sugahara, S. Oida and T. Yokoyama, “High performance FPGA controller for digital control of power electronics applications, IEEE 6th International Power Electronics and Motion Control Conference, 2009, pp. 1425-1429, doi: 10.1109/IPEMC.2009.5157608.

[65] T. Kahl and S. Dieckerhoff, “Comparison of FPGA- and microcontroller-based control of a high-dynamic power electronic converter,” in Proceedings of 2017 IEEE

18th Workshop on Control and Modeling for Power Electronics (COMPEL), Stanford, CA, USA, Jul. 2017, pp. 1–6.

[66] S. Lucia, D. Navarro, Ó. Lucía, P. Zometa, and R. Findeisen, “Optimized FPGA implementation of model predictive control for embedded systems using high-level synthesis tool,” in IEEE Transactions on Industrial Informatics, vol. 14, no. 1, pp. 137–145, Jan. 2018.

[67] W. Tu, G. Luo, Z. Chen, C. Liu, and L. Cui, “FPGA implementation of predictive cascaded speed and current control of PMSM drives with two-time-scale optimization,” in IEEE Transactions on Industrial Informatics, vol. 15, no. 9, pp. 5276–5288, Sept. 2019.

[68] S. Wendel, A. Dietz, and R. Kennel, “FPGA based finite-set model predictive current control for small PMSM drives with efficient resource streaming,” in Proceedings of 2017 IEEE International Symposium on Predictive Control of Electrical Drives and Power Electronics (PRECEDE), Pilsen, Czech Republic, Sept. 2017, pp. 66–71.

[69] T. J. Vyncke, S. Thielemans, and J. A. Melkebeek, “Finite-set model based predictive control for flying-capacitor converters: cost function design and efficient FPGA implementation,” in IEEE Transactions on Industrial Informatics, vol. 9, no. 2, pp. 1113–1121, May 2013.

[70] Z. Zhang, F. Wang, T. Sun, J. Rodríguez, and R. Kennel, “FPGA-based experimental investigation of a quasi-centralized model predictive control for back-to-back converters,” in IEEE Transactions on Power Electronics, vol. 31, no. 1, pp. 662–674, Jan. 2016.

[71] E. Lupon, S. Busquets-Monge, and J. Nicolas-Apruzzese, “FPGA implementation of a PWM for a three-phase DC-AC multilevel active clamped converter,” in IEEE Transactions on Industrial Informatics, vol. 10, no. 2, pp. 1296–1306, May 2014.

[72] J. Holtz, M. Höltgen, and J. O. Krah, “A space vector modulator for the high-switching frequency control of three-level SiC inverters,” in IEEE Transactions on Power Electronics, vol. 29, no. 5, pp. 2618–2626, May 2014.

[73] M. Salmento. Synchronization and signal detection for impulsive UWB PLC systems, Master Thesis. Federal University of Juiz de Fora, 2014.

[74] Nantional Instruments. FPGA Fundamentals (white papper). http://www.ni.com/white-paper/ 6983/en/, 2012. Accessed: 2017-06-15.

[75] Uwe Meyer-Baese. Digital Signal Processing with Field Programmable Gate Arrays. Springer Publishing Company, Incorporated, 3rd edition, 2007.

[76] R. Srivastava, Y. K. Chauhan and B. Kumar, "Generation of PWM using Verilog in FPGA," *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, Chennai, India, 2016, pp. 4593-4597, doi: 10.1109/ICEEOT.2016.7755586.

[77] N. Moonen, R. Vogt-Ardatjew and F. Leferink, "Simulink-Based FPGA Control for EMI Investigations of Power Electronic Systems," in *IEEE Transactions on Electromagnetic Compatibility*, vol. 63, no. 4, pp. 1266-1273, Aug. 2021, doi: 10.1109/TEMC.2020.3042301.

[78] S. Sen, P. Saha and S. Saha, "FPGA-Supported HDL Approach to Implement Reversible Logic Gate-Based ALU," *2023 11th International Conference on Internet*

of Everything, Microwave Engineering, Communication and Networks (IEMECON), Jaipur, India, 2023, pp. 1-5, doi: 10.1109/IEMECON56962.2023.10092307.

[79] Z. Luo, Y. Li, X. Zhang, and K. Wang, "Vector control implementation in field programmable gate array for 200 kHz GaN-based motor drive systems," *Journal of engineering*, vol. 2018, no. 13, pp. 650–653, Jan. 2018, doi: https://doi.org/10.1049/joe.2018.0032.

[80] M. Nitzsche, J. Haarer, V. Ketchedjian and J. Roth-Stielow, "Design of a 9-Level Flying Capacitor Inverter with Very Fast Response of the Controller," *IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society*, Singapore, 2020, pp. 2424-2429, doi: 10.1109/IECON43393.2020.9254392.

[81] B. Tufekci, B. Onal, H. Dere and H. F. Ugurdag, "Efficient FPGA Implementation of Field Oriented Control for 3-Phase Machine Drives," *2020 IEEE East-West Design & Test Symposium (EWDTS)*, Varna, Bulgaria, 2020, pp. 1-5, doi: 10.1109/EWDTS50664.2020.9224884.

[82] Y. Kung, Hoang Than, Y. Lin, and L. Huang, "FPGA Based Speed Controller Design for a Ceiling Fan Motor," in Proc. Int. Future Energy Electronics Conference (IFEEC) and ECCE Asia, pp. 30–34, 2017.

[83] S. Suneeta, R. Srinivasan, and R. Sagar, "FPGA Based Control Method for Three Phase BLDC Motor," Int. Journal of Electrical and Computer Engineering, vol. 6, pp. 1434–1440, 2016.

[84] C. B. Barth *et al*., "Design and Control of a GaN-Based, 13-Level, Flying Capacitor Multilevel Inverter," in *IEEE Journal of Emerging and Selected Topics in Power Electronics*, vol. 8, no. 3, pp. 2179-2191, Sept. 2020, doi: 10.1109/JESTPE.2019.2956166.

[85] G. H. M. Tavares, M. L. G. Salmento, W. J. Paula, D. d. C. Pereira and H. A. C. Braga, "Implementation of a high frequency PWM signal in FPGA FOR GaN power devices switching," *2017 Brazilian Power Electronics Conference (COBEP)*, Juiz de Fora, Brazil, 2017, pp. 1-7, doi: 10.1109/COBEP.2017.8257309.

[86] Tung Duong Do, Nam Duong Le, Vu Hoang Phuong, and Nguyen Tung Lam, "Implementation of FOC algorithm using FPGA for GaN-based three phase induction motor drive," Bulletin of Electrical Engineering and Informatics, vol. 11, no. 2, pp. 636–645, Apr. 2022, doi: https://doi.org/10.11591/eei.v11i2.3569.

[87] S. Jena, G. Panda and R. Peesapati, "Real-time analysis and simulation of multi-string grid connected photovoltaic inverter using FPGA," 2016 IEEE 6th International Conference on Power Systems (ICPS), New Delhi, India, 2016, pp. 1-6, doi: 10.1109/ICPES.2016.7584068.

[88] M. Mosa, G. M. Dousoky and H. Abu-Rub, "A novel FPGA implementation of a model predictive controller for SiC-based Quasi-Z-Source inverters," *2014 IEEE Applied Power Electronics Conference and Exposition - APEC 2014*, Fort Worth, TX, USA, 2014, pp. 1293-1298, doi: 10.1109/APEC.2014.6803473.

[89] A. Singh and K. Prabakar, "Controller-Hardware-in-the-Loop Testbed for Fast-Switching SiC-Based 50-kW PV Inverter," IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society, Washington, DC, USA, 2018, pp. 1109-1115, doi: 10.1109/IECON.2018.8591991.

[90] C. Guzmán, A. Cardenas and K. Agbossou, "Control of voltage source inverter using FPGA implementation of ADALINE-FLL," IECON 2012 - 38th Annual

Conference on IEEE Industrial Electronics Society, Montreal, QC, Canada, 2012, pp. 3037-3042, doi: 10.1109/IECON.2012.6389413.

[91] R. Ekström and M. Leijon, “FPGA Control Implementation of a Grid-Connected Current-Controlled Voltage-Source Inverter,” Journal of Control Science and Engineering, vol. 2013, pp. 1–10, 2013, doi: https://doi.org/10.1155/2013/713293.

[92] S. Jena, G. Panda, and R. Peesapati, “FPGA-based implementation for improved control scheme of grid-connected PV system with 3-phase 3-level NPC-VSI,” International Journal of Circuit Theory and Applications, vol. 46, no. 4, pp. 942–964, Feb. 2018, doi: https://doi.org/10.1002/cta.2448.

[93] H. Park, Y.-J. Kim, and H. Kim, “PV cell model by single-diode electrical equivalent circuit,” *Journal of Electrical Engineering and Technology*, vol. 11, no. 5, pp. 1323–1331, Sep. 2016, doi: 10.5370/jeet.2016.11.5.1323.

[94] R. A. Messenger and A. Abtahi, *Photovoltaic systems engineering*. CRC Press, 2017.

[95] N. Mohan, T. M. Undeland, and W. P. Robbins, *Power Electronics: Converters, Applications, and Design*, 4th ed. Hoboken, NJ, USA: Wiley, 2019.

[96] Digilent, Inc., "Nexys A7™ FPGA Board Reference Manual," Rev. D.3, Jan. 25,2022.

[97] Texas Instruments, "Delfino™ TMS320F28379D controlCARD R1.3," User's Guide, SPRUI76B, Rev. B, June 2022.

APPENDIX A, Coding Sheet

FIL Simulation

```
-- ----------------------------------------------
-- File Name: fix_point_converter_inverter_controller1_fil.vhd
-- Created:   17-Dec-2024 15:33:17
-- Copyright  2024 MathWorks, Inc.
-- ----------------------------------------------

LIBRARY IEEE;
USE IEEE.std_logic_1164.all;
USE IEEE.numeric_std.ALL;


-- Required Xilinx Library
USE IEEE.STD_LOGIC_ARITH.all;
USE IEEE.STD_LOGIC_UNSIGNED.all;

LIBRARY UNISIM;
USE UNISIM.VComponents.all;

ENTITY fix_point_converter_inverter_controller1_fil IS
PORT (
    sysrst                  : IN  std_logic;
    sysclk                  : IN  std_logic
);
END fix_point_converter_inverter_controller1_fil;

ARCHITECTURE rtl of fix_point_converter_inverter_controller1_fil IS

COMPONENT clk_wiz_0 IS
PORT (
    clk_in1                 : IN  std_logic;
    reset                   : IN  std_logic;
    clk_out1                : OUT std_logic;
    locked                  : OUT std_logic
);
END COMPONENT;

COMPONENT BSCANE2 IS
PORT (
    TDO                     : IN  std_logic;
    TCK                     : OUT std_logic;
    DRCK                    : OUT std_logic;
    RESET                   : OUT std_logic;
    SHIFT                   : OUT std_logic;
    UPDATE                  : OUT std_logic;
    TDI                     : OUT std_logic;
```

```
    TMS                 : OUT std_logic;
    CAPTURE             : OUT std_logic;
    SEL                 : OUT std_logic;
    RUNTEST             : OUT std_logic
);
END COMPONENT;

COMPONENT jtag_mac IS
PORT (
    chif_din_ready      : IN  std_logic;
    chif_dout_valid     : IN  std_logic;
    sel                 : IN  std_logic;
    sys_rst             : IN  std_logic;
    tck                 : IN  std_logic;
    chif_dout           : IN  std_logic_vector(7 DOWNTO 0);
    chif_clk            : IN  std_logic;
    tdi                 : IN  std_logic;
    chif_reset          : OUT std_logic;
    chif_din_valid      : OUT std_logic;
    tdo                 : OUT std_logic;
    chif_din            : OUT std_logic_vector(7 DOWNTO 0);
    chif_dout_ready     : OUT std_logic;
    chif_simcycle       : OUT std_logic_vector(15 DOWNTO 0)
);
END COMPONENT;

COMPONENT mwfil_chiftop IS
PORT (
    clk                 : IN  std_logic;
    reset               : IN  std_logic;
    din                 : IN  std_logic_vector(7 DOWNTO 0);
    din_valid           : IN  std_logic;
    dout_ready          : IN  std_logic;
    simcycle            : IN  std_logic_vector(15 DOWNTO 0);
    din_ready           : OUT std_logic;
    dout                : OUT std_logic_vector(7 DOWNTO 0);
    dout_valid          : OUT std_logic
);
END COMPONENT;

  SIGNAL din            : std_logic_vector(7 DOWNTO 0); -- std8
  SIGNAL din_valid      : std_logic; -- boolean
  SIGNAL din_ready      : std_logic; -- boolean
  SIGNAL dout           : std_logic_vector(7 DOWNTO 0); -- std8
  SIGNAL dout_valid     : std_logic; -- boolean
  SIGNAL dout_ready     : std_logic; -- boolean
  SIGNAL simcycle       : std_logic_vector(15 DOWNTO 0); -- std16
  SIGNAL dutClk         : std_logic; -- boolean
  SIGNAL chif_reset     : std_logic; -- boolean
  SIGNAL locked         : std_logic; -- boolean
```

```
  SIGNAL mac_reset                : std_logic; -- boolean
  SIGNAL CAPTURE                  : std_logic; -- boolean
  SIGNAL DRCK                     : std_logic; -- boolean
  SIGNAL RESET                    : std_logic; -- boolean
  SIGNAL RUNTEST                  : std_logic; -- boolean
  SIGNAL SEL                      : std_logic; -- boolean
  SIGNAL SHIFT                    : std_logic; -- boolean
  SIGNAL TCK                      : std_logic; -- boolean
  SIGNAL TDI                      : std_logic; -- boolean
  SIGNAL TMS                      : std_logic; -- boolean
  SIGNAL UPDATE                   : std_logic; -- boolean
  SIGNAL TDO                      : std_logic; -- boolean
  SIGNAL dcm_reset                : std_logic; -- boolean

BEGIN

u_clk_wiz_0: clk_wiz_0
PORT MAP(
      clk_out1          => dutClk,
      locked            => locked,
      clk_in1           => sysclk,
      reset             => dcm_reset
);

u_BSCANE2: BSCANE2
PORT MAP(
      TCK               => TCK,
      DRCK              => DRCK,
      RESET             => RESET,
      SHIFT             => SHIFT,
      UPDATE            => UPDATE,
      TDI               => TDI,
      TDO               => TDO,
      TMS               => TMS,
      CAPTURE           => CAPTURE,
      SEL               => SEL,
      RUNTEST           => RUNTEST
);

u_jtag_mac: jtag_mac
PORT MAP(
      chif_din_ready    => din_ready,
      chif_dout_valid   => dout_valid,
      chif_reset        => chif_reset,
      sel               => SEL,
      chif_din_valid    => din_valid,
      sys_rst           => mac_reset,
      tck               => TCK,
      tdo               => TDO,
      chif_dout         => dout,
```

```
        chif_din            => din,
        chif_dout_ready     => dout_ready,
        chif_clk            => dutClk,
        tdi                 => TDI,
        chif_simcycle       => simcycle
);

u_mwfil_chiftop: mwfil_chiftop
PORT MAP(
        clk                 => dutClk,
        reset               => chif_reset,
        din                 => din,
        din_valid           => din_valid,
        din_ready           => din_ready,
        dout                => dout,
        dout_valid          => dout_valid,
        dout_ready          => dout_ready,
        simcycle            => simcycle
);

mac_reset <=  NOT locked;
dcm_reset <=  NOT sysrst;

END;


-- ----------------------------------------------
-- File Name: jtag_mac_fifo_wrapper.vhd
-- Created:   17-Dec-2024 15:33:16
-- Copyright  2024 MathWorks, Inc.
-- ----------------------------------------------

LIBRARY IEEE;
USE IEEE.std_logic_1164.all;
USE IEEE.std_logic_unsigned.all;
USE IEEE.numeric_std.ALL;
USE IEEE.std_logic_arith.all;

entity jtag_mac_fifo_wrapper is
    port (
    rst : IN STD_LOGIC;
    wr_clk : IN STD_LOGIC;
    rd_clk : IN STD_LOGIC;
    din : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
    wr_en : IN STD_LOGIC;
    rd_en : IN STD_LOGIC;
    dout : OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
    full : OUT STD_LOGIC;
    empty : OUT STD_LOGIC;
    almost_full: OUT STD_LOGIC;
```

```
    rd_data_count : OUT STD_LOGIC_VECTOR(11 DOWNTO 0)
);
end jtag_mac_fifo_wrapper;

architecture Behavioral of jtag_mac_fifo_wrapper is

COMPONENT jtag_mac_fifo
  PORT (
    rst : IN STD_LOGIC;
    wr_clk : IN STD_LOGIC;
    rd_clk : IN STD_LOGIC;
    din : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
    wr_en : IN STD_LOGIC;
    rd_en : IN STD_LOGIC;
    dout : OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
    full : OUT STD_LOGIC;
    almost_full: OUT STD_LOGIC;
    empty : OUT STD_LOGIC;
    rd_data_count : OUT STD_LOGIC_VECTOR(11 DOWNTO 0)
  );
END COMPONENT;

begin

u_jtag_mac_fifo : jtag_mac_fifo
port map(
    rst => rst,
    wr_clk => wr_clk,
    rd_clk => rd_clk,
    din => din,
    wr_en => wr_en,
    rd_en => rd_en,
    dout => dout,
    full => full,
    empty => empty,
    almost_full => almost_full,
    rd_data_count => rd_data_count

);

end Behavioral;


-- ----------------------------------------------
-- File Name: mwfil_bus2dut.vhd
-- Created:   17-Dec-2024 15:33:16
-- Copyright  2024 MathWorks, Inc.
-- ----------------------------------------------

LIBRARY IEEE;
```

```
USE IEEE.std_logic_1164.all;
USE IEEE.std_logic_unsigned.all;
USE IEEE.numeric_std.ALL;

entity mwfil_bus2dut is
generic(
          WORDSIZE : integer range 1 to 256 := 64;
   OUTWORD: integer range 0 to 2047 := 3);
port (
   -- Interface with the WORDSIZE-bit data bus
   din        : in std_logic_vector(WORDSIZE-1 downto 0);
   din_valid   : in std_logic;
   din_ready   : out std_logic;

   -- Interface with the b2d fifo
   fifo_data   : out std_logic_vector(OUTWORD*WORDSIZE-1 downto 0);
   fifo_wrreq  : out std_logic;
   fifo_full   : in  std_logic;

   -- Control signals
   clk : in std_logic;
   reset: in std_logic
);
end mwfil_bus2dut;

architecture rtl of mwfil_bus2dut is

constant OUTWIDTH: integer := OUTWORD*WORDSIZE;
signal shiftreg, shiftreg_next: std_logic_vector(OUTWIDTH-1 downto 0);
signal counter,counter_inc: integer range 0 to 1023;

begin

din_ready <= not fifo_full;

shiftreg_next(OUTWIDTH-1 downto OUTWIDTH-WORDSIZE) <= din;
shiftreg_next(OUTWIDTH-WORDSIZE-1 downto 0) <= shiftreg(OUTWIDTH-1 downto
WORDSIZE);
fifo_data <= shiftreg_next;

fifo_wrreq <= '1' when (counter_inc = OUTWORD and din_valid = '1')
           else '0';

counter_inc <= counter + 1;

process(clk)
begin
   if clk'event and clk = '1' then
      if reset = '1' then
         counter <= 0;
```

```
        elsif din_valid = '1' then
            -- Load data
            shiftreg <= shiftreg_next;
            if counter_inc = OUTWORD then
                counter <= 0;
            else
                counter <= counter_inc;
            end if;
        end if;
    end if;
end process;

end architecture;


-- -------------------------------------------
-- File Name: mwfil_controller.vhd
-- Created:   17-Dec-2024 15:33:16
-- Copyright  2024 MathWorks, Inc.
-- -------------------------------------------


LIBRARY IEEE;
USE IEEE.std_logic_1164.all;
USE IEEE.std_logic_unsigned.all;
USE IEEE.numeric_std.ALL;
USE IEEE.std_logic_arith.all;

entity mwfil_controller is
generic(
    HASENABLE: integer range 0 to 1 := 1;
    FREERUNNING: integer range 0 to 1 := 0
);
port(
    -- Interface with B2D
    b2d_fifo_empty  : in std_logic;
    b2d_fifo_rdreq  : out std_logic;

    -- Interface with D2B
    d2b_fifo_wrreq  : out std_logic;
    d2b_fifo_full   : in std_logic;

    -- DUT controller signals
    dut_enable      : out std_logic;

    -- Free running signals
    dut_din_valid   : out std_logic;
    dut_din_ready   : in std_logic;
    dut_dout_valid  : in std_logic;
    dut_dout_ready  : out std_logic;
```

```
   -- Control signals
   clk : in std_logic;
   reset: in std_logic;
   simcycle: in std_logic_vector(15 downto 0)
   );
end mwfil_controller;

architecture rtl of mwfil_controller is

signal simrem, simrem_next : std_logic_vector(15 downto 0);

signal dut_enable_int      : std_logic;
signal b2d_fifo_rdreq_int  : std_logic;
signal zeros16: std_logic_vector(15 downto 0);
signal reset_d1            : std_logic;
begin

-- constant zero
zeros16 <= conv_std_logic_vector(0,16);

LOCKEDSTEP_CTRL: if FREERUNNING=0 generate
dut_enable <= dut_enable_int;

simrem_next <= simrem - '1' when (dut_enable_int = '1' and simrem > zeros16)
            else simrem;

-- We only read data from b2d fifo when (1) that fifo is nonempty, and
-- (2) we are done with the current or all previous data
b2d_fifo_rdreq_int <= '1' when simrem_next = zeros16 and b2d_fifo_empty = '0'
                          else '0';


b2d_fifo_rdreq <= b2d_fifo_rdreq_int;

-- We write to the fifo whenever DUT is enabled
d2b_fifo_wrreq <= dut_enable_int;

process(clk)
begin
   if clk'event and clk='1' then
        reset_d1 <= reset;
      if reset = '1' then
         simrem   <= (others => '0');
            if HASENABLE = 1 then
               dut_enable_int <= '0';
            else
               -- In case of synchronous reset and without clock enable,
               -- We need to toggle enable signal, which is used as clock.
               dut_enable_int <= reset and (not reset_d1);
```

```
            end if;
        else
            dut_enable_int <= '0';
            -- read new data in
            if b2d_fifo_rdreq_int = '1' then
                if HASENABLE = 1 then
                    dut_enable_int <= not d2b_fifo_full;
                else
                    dut_enable_int <= (not d2b_fifo_full) and (not dut_enable_int);
                end if;
                simrem <= simcycle;
            else
                simrem <= simrem_next;
                if(simrem_next > zeros16) then
                    if HASENABLE = 1 then
                      dut_enable_int <= not d2b_fifo_full;
                    else
                      dut_enable_int <= (not d2b_fifo_full) and (not dut_enable_int);
                    end if;
                end if;
            end if;
        end if;
    end if;
end process;
end generate;

FREERUNNING_CTRL: if FREERUNNING=1 generate
dut_enable <= '1';

simrem_next <= simrem - '1' when simrem > zeros16
            else simrem;

-- We only read data from b2d fifo when (1) that fifo is nonempty, and
-- (2) we are done with the current or all previous data
b2d_fifo_rdreq_int <= '1' when simrem_next = zeros16 and b2d_fifo_empty = '0' and
dut_din_ready = '1'
                          else '0';

b2d_fifo_rdreq <= b2d_fifo_rdreq_int;

process(clk)
begin
    if clk'event and clk='1' then
        if reset = '1' then
            simrem   <= (others => '0');
            dut_din_valid <= '0';
        else
            -- read new data in
            if b2d_fifo_rdreq_int = '1' then
                simrem <= simcycle;
```

```
            else
               simrem <= simrem_next;
            end if;
            dut_din_valid <= b2d_fifo_rdreq_int;
         end if;
      end if;
   end process;

   -- todo: dut_dout_valid  dut_dout_ready
   d2b_fifo_wrreq <= '0'; -- <=dut_dout_valid;

   end generate;

   end architecture;


   -- -------------------------------------------------
   -- File Name: mwfil_dut2bus.vhd
   -- Created:   17-Dec-2024 15:33:16
   -- Copyright  2024 MathWorks, Inc.
   -- -------------------------------------------------

   LIBRARY IEEE;
   USE IEEE.std_logic_1164.all;
   USE IEEE.numeric_std.ALL;

   entity mwfil_dut2bus is
   generic(
            WORDSIZE : integer range 1 to 256 := 64;
            OUTWORD: integer range 0 to 2047 := 2);
   port (
            -- Interface with the DUT
            d2b_fifo_q      : in std_logic_vector(OUTWORD*WORDSIZE-1 downto 0);
            d2b_fifo_rdreq  : out std_logic;
            d2b_fifo_empty  : in std_logic;

            -- Interface with the WORDSIZE-bit bus
            dout        : out std_logic_vector(WORDSIZE-1 downto 0);
            dout_valid  : out std_logic;
            dout_ready  : in  std_logic;

            -- Control signals
            clk : in std_logic;
            reset: in std_logic
            );
   end mwfil_dut2bus;

   architecture rtl of mwfil_dut2bus is

   constant OUTWIDTH: integer := OUTWORD*WORDSIZE;
```

```
signal shiftreg : std_logic_vector(OUTWIDTH-1 downto 0);

signal remword, remword_next: integer range 0 to OUTWORD+1;
signal d2b_fifo_valid : std_logic;
signal d2b_fifo_rdreq_int: std_logic;
begin
d2b_fifo_rdreq <= d2b_fifo_rdreq_int;
-- we can only read from d2b fifo when the current shiftreg is about to shift out entirely, or this
is no data in shiftreg
d2b_fifo_rdreq_int   <= '1' when (remword_next = 0  and d2b_fifo_empty = '0') else
                  '0';
-- Calculate the remaining word in shiftreg after the current clock cycle
remword_next <= OUTWORD -1 when d2b_fifo_valid = '1' and dout_ready = '1' else
            OUTWORD    when d2b_fifo_valid = '1' else
            remword -1 when dout_ready = '1' and remword >=1 else
            remword;
process(clk)
begin
   if clk'event and clk='1' then
      if reset = '1' then
         remword <= 0;
         dout    <= (others =>'0');
      else
         d2b_fifo_valid <= d2b_fifo_rdreq_int;
         dout_valid  <= '0';
         remword <= remword_next;
         if d2b_fifo_valid = '1' and dout_ready = '1' then
            -- Load and shift at the same time
            dout       <= d2b_fifo_q(WORDSIZE-1 downto 0);
            if OUTWORD >1 then
               shiftreg(OUTWIDTH-WORDSIZE-1 downto 0) <= d2b_fifo_q(OUTWIDTH-1
downto WORDSIZE);
            end if;
            dout_valid  <= '1';
         elsif d2b_fifo_valid = '1' and dout_ready = '0' then
            -- Only load the shift register
            shiftreg  <= d2b_fifo_q;
         elsif dout_ready = '1' and remword>0 then
            -- Shift out one word
            dout_valid <= '1';
            dout <= shiftreg(WORDSIZE-1 downto 0);
            if OUTWORD >1 then
               shiftreg(OUTWIDTH-WORDSIZE-1   downto   0)   <=   shiftreg(OUTWIDTH-1
downto WORDSIZE);
            end if;
         end if;
      end if;
   end if;
end process;
end architecture;
```

PIL Simulation

```
/*
 * File: ControlBlock.c
 */

#include "ControlBlock.h"
#include "IQmathLib.h"
#include "rtwtypes.h"
#include <string.h>
#include "ControlBlock_private.h"

/* Block states (default storage) */
DW_ControlBlock_T ControlBlock_DW;

/* External inputs (root inport signals with default storage) */
ExtU_ControlBlock_T ControlBlock_U;

/* External outputs (root outports fed by signals with default storage) */
ExtY_ControlBlock_T ControlBlock_Y;

/* Real-time model */
static RT_MODEL_ControlBlock_T ControlBlock_M_;
RT_MODEL_ControlBlock_T *const ControlBlock_M = &ControlBlock_M_;

/* Model step function */
void ControlBlock_step(void)
{
  int64_T tmp;
  int64_T tmp_0;
  int32_T HwModeRegister2_DSTATE_h_tmp;
  int32_T HwModeRegister2_DSTATE_o_tmp;
  int32_T HwModeRegister_DSTATE_f_tmp;
  int32_T HwModeRegister_DSTATE_m_tmp;
  int32_T rtb_DiscreteTimeIntegrator_reg;
  int32_T rtb_HwModeRegister1;
  int32_T rtb_HwModeRegister1_c;
  int32_T rtb_ProportionalGain_a;
  int32_T rtb_ProportionalGain_i;
  int32_T rtb_Sum4;
  int32_T rtb_Switch_o_idx_0;
  int32_T rtb_Switch_o_idx_1;
  int32_T rtb_TmpSignalConversionAtKalp_0;
  int32_T rtb_TmpSignalConversionAtKalp_1;
  int32_T rtb_acos_d;
  int32_T rtb_convert_pu_out_dtc;
  int32_T rtb_convert_pu_out_dtc_a;
  int32_T rtb_sum_Qs;
  int32_T rtb_sum_Qs_g;
```

```
int32_T rtb_sum_Qs_m;
uint32_T rtb_Conversion;
uint32_T rtb_Switch1;
int16_T idxDelay;
uint16_T rtb_Sum_de_idx_1_tmp;
uint16_T rtb_Sum_idx_1_tmp;
boolean_T rtb_Compare;

/* DataTypeConversion: '<S24>/Gain_out_dtc' incorporates:
 *  Delay: '<S24>/PipelineRegister'
 *  Sum: '<S20>/sum_Qs'
 */
rtb_sum_Qs = (int32_T)(int64_T)(ControlBlock_DW.PipelineRegister_DSTATE >> 32U);

/* DataTypeConversion: '<S24>/Gain1_out_dtc' incorporates:
 *  DataTypeConversion: '<S46>/convert_pu_out_dtc'
 *  Delay: '<S24>/PipelineRegister1'
 */
rtb_convert_pu_out_dtc = (int32_T)(int64_T)
  (ControlBlock_DW.PipelineRegister1_DSTATE >> 31U);

/* Outport: '<Root>/v_ref2' incorporates:
 *  DataTypeConversion: '<S46>/convert_pu_out_dtc'
 *  Delay: '<S24>/delayMatch'
 *  Delay: '<S24>/delayMatch1'
 *  Delay: '<S24>/delayMatch2'
 *  Sum: '<S20>/sum_Qs'
 *  Sum: '<S24>/Sum1_stage2'
 *  Sum: '<S24>/Sum1_stage3'
 *  Sum: '<S24>/Sum2_stage2'
 *  Sum: '<S24>/Sum2_stage3'
 */
ControlBlock_Y.v_ref2[0] = ControlBlock_DW.delayMatch2_DSTATE_fe[0U];
ControlBlock_Y.v_ref2[1] = (int32_T)((int32_T)(rtb_sum_Qs +
  rtb_convert_pu_out_dtc) + ControlBlock_DW.delayMatch_DSTATE_m[0U]);
ControlBlock_Y.v_ref2[2] = (int32_T)((int32_T)(rtb_sum_Qs -
  rtb_convert_pu_out_dtc) + ControlBlock_DW.delayMatch1_DSTATE_iu[0U]);

/* Product: '<S24>/Gain' incorporates:
 *  Delay: '<S24>/HwModeRegister'
 *  Delay: '<S24>/HwModeRegister1'
 *  Delay: '<S24>/PipelineRegister'
 */
ControlBlock_DW.PipelineRegister_DSTATE = (int64_T)((int64_T)
  ControlBlock_DW.HwModeRegister_DSTATE_f5 * (int64_T)
  ControlBlock_DW.HwModeRegister1_DSTATE_i1);

/* Product: '<S24>/Gain1' incorporates:
 *  Delay: '<S24>/HwModeRegister2'
 *  Delay: '<S24>/HwModeRegister3'
```

```
 *  Delay: '<S24>/PipelineRegister1'
 */
ControlBlock_DW.PipelineRegister1_DSTATE = (int64_T)((int64_T)
  ControlBlock_DW.HwModeRegister2_DSTATE_p * (int64_T)
  ControlBlock_DW.HwModeRegister3_DSTATE_jf);

/* Switch: '<S34>/Switch' incorporates:
 *  DataTypeConversion: '<S33>/dsin_out_dtc'
 *  DataTypeConversion: '<S33>/qcos_out_dtc'
 *  Delay: '<S33>/PipelineRegister1'
 *  Delay: '<S33>/PipelineRegister2'
 *  Sum: '<S33>/sum_beta'
 */
ControlBlock_DW.HwModeRegister1_DSTATE_i1 = (int32_T)((int32_T)(int64_T)
  (ControlBlock_DW.PipelineRegister2_DSTATE >> 14U) + (int32_T)(int64_T)
  (ControlBlock_DW.PipelineRegister1_DSTATE_l >> 14U));

/* Update for Delay: '<S24>/HwModeRegister3' incorporates:
 *  DataTypeConversion: '<S33>/dcos_out_dtc'
 *  DataTypeConversion: '<S33>/qsin_out_dtc'
 *  Delay: '<S33>/PipelineRegister'
 *  Delay: '<S33>/PipelineRegister3'
 *  Switch: '<S34>/Switch'
 *  UnaryMinus: '<S34>/Unary_Minus'
 */
ControlBlock_DW.HwModeRegister3_DSTATE_jf = (int32_T)((int32_T)(int64_T)
  (ControlBlock_DW.PipelineRegister3_DSTATE >> 14U) - (int32_T)(int64_T)
  (ControlBlock_DW.PipelineRegister_DSTATE_ow >> 14U));

/* Update for Delay: '<S33>/PipelineRegister' incorporates:
 *  Delay: '<S33>/HwModeRegister'
 *  Delay: '<S33>/HwModeRegister1'
 *  Product: '<S33>/dcos'
 */
ControlBlock_DW.PipelineRegister_DSTATE_ow = (int64_T)((int64_T)
  ControlBlock_DW.HwModeRegister_DSTATE_mx * (int64_T)
  ControlBlock_DW.HwModeRegister1_DSTATE_i);

/* Update for Delay: '<S33>/PipelineRegister1' incorporates:
 *  Delay: '<S33>/HwModeRegister2'
 *  Delay: '<S33>/HwModeRegister3'
 *  Product: '<S33>/dsin'
 */
ControlBlock_DW.PipelineRegister1_DSTATE_l = (int64_T)((int64_T)
  ControlBlock_DW.HwModeRegister2_DSTATE_j * (int64_T)
  ControlBlock_DW.HwModeRegister3_DSTATE_b);

/* Update for Delay: '<S33>/PipelineRegister2' incorporates:
 *  Delay: '<S33>/HwModeRegister4'
 *  Delay: '<S33>/HwModeRegister5'
```

```
   *  Product: '<S33>/qcos'
   */
  ControlBlock_DW.PipelineRegister2_DSTATE = (int64_T)((int64_T)
    ControlBlock_DW.HwModeRegister4_DSTATE_o * (int64_T)
    ControlBlock_DW.HwModeRegister5_DSTATE_p);

  /* Update for Delay: '<S33>/PipelineRegister3' incorporates:
   *  Delay: '<S33>/HwModeRegister6'
   *  Delay: '<S33>/HwModeRegister7'
   *  Product: '<S33>/qsin'
   */
  ControlBlock_DW.PipelineRegister3_DSTATE = (int64_T)((int64_T)
    ControlBlock_DW.HwModeRegister6_DSTATE_i * (int64_T)
    ControlBlock_DW.HwModeRegister7_DSTATE);

  /* Delay: '<S33>/delayMatch' incorporates:
   *  Delay: '<S33>/HwModeRegister1'
   */
  ControlBlock_DW.HwModeRegister1_DSTATE_i =
    ControlBlock_DW.delayMatch_DSTATE_h[0];

  /* Delay: '<S33>/delayMatch1' incorporates:
   *  Delay: '<S33>/HwModeRegister7'
   */
  ControlBlock_DW.HwModeRegister7_DSTATE                                    =
ControlBlock_DW.delayMatch1_DSTATE_g
    [0];

  /* Delay: '<S33>/delayMatch2' incorporates:
   *  Delay: '<S33>/HwModeRegister5'
   */
  ControlBlock_DW.HwModeRegister5_DSTATE_p =
    ControlBlock_DW.delayMatch2_DSTATE_f[0];

  /* Sum: '<S27>/Sum6' incorporates:
   *  DataTypeConversion: '<S27>/Product1_out_dtc'
   *  Delay: '<S27>/PipelineRegister1'
   *  Delay: '<S27>/delayMatch1'
   */
  rtb_sum_Qs = (int32_T)((int32_T)(int64_T)
    (ControlBlock_DW.PipelineRegister1_DSTATE_p >> 14U) +
    ControlBlock_DW.delayMatch1_DSTATE_c[0]);
has popup
```