PROJECT WORK

ON

“HARDWARE SIMULATION FOR SPECTRUM ANALYSIS USING VITIS MODEL COMPOSER”

By

JARUPLAVATH UDAY KARTHIK (2214053)

AT

DEFENCE ELECTRONICS RESEARCH LABORATORY, HYDERABAD-05

Under the esteemed guidance of
Mr. Om Prakash RC, Sc- F, DLRL

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGY SILCHAR, ASSAM 788010

2025

DECLARATION

I hereby declare that the results embodied in this dissertation titled **“HARDWARE SIMULATION FOR SPECTRUM ANALYSIS USING VITIS MODEL COMPOSER”** is carried out by me during the year 2024 –2025 in partial fulfillment of the award of B.Tech (ELECTRONICS & COMMUNICATION ENGINEERING) from **“NATIONAL INSTITUTE OF TECHNOLOGY SILCHAR, ASSAM”.** I have not submitted the same to any other university or organization for the award of any other degree.

Jaruplavath Uday Karthik

ACKNOWLEDGEMENT

This is an acknowledgement of the intensive drive and technical competence of many individuals who have contributed to the success of my project.

We are grateful to **Sri N Srinivas Rao, DS & Director, DLRL**, Hyderabad and **Sri JRC Sarma, Sc 'F'**, Wing Head HRD and Members of the HRD for granting us permission for the practical training through development of this project in DLRL.

A Special note of Thanks to **Mrs UVV Krishnaveni, Sc-G¸** Gp Director of AIRBORNE AND NAVAL EW TECHNOLOGIES Directorate, DLRL who encouraged us in our work.

I am immensely thankful to **Mrs J Vijayalakshmi, Sc-F**, Division Head, of GROUND BASED ELINT Division, DLRL for giving me this opportunity and also providing the facilities at Division.

I am obliged and grateful to our guide **Sri Om Prakash RC, Sc-F** of GROUND BASED ELINT Division, DLRL for his valuable suggestions and sagacious guidance in all respects during the course of our training.

I would like to express our gratitude to my Sponsor of DLRL employee, who was very friendly and co-operative.

DLRL PROFILE

DEFENCE ELECTRONICS RESEARCH LABORATORY (D.L.R.L) was established in the year 1962 under the aegis of Defence Research and Development Organization (DRDO), Ministry of Defence, to meet the current and future needs of tri services Army, Navy and Air force equipping them with Electronics Warfare Systems.

DLRL has been entrusted with the primary responsibility of the design and development of Electronic Warfare Systems covering both Communication and RADAR Frequency bands.

DLRL consists of large number of dedicated technical and scientific manpower adequately supported by sophisticated hardware and software development facilities. Computers and dedicated Workstations are extensively used for, design and development of sub-systems. Main software required for various types of applications is developed in-house. The quality assurance group is responsible for quality assurance of software developed for Electronic Warfare Systems.

DLRL has number of supporting and technology groups to help the completion of the projects on time and to achieve a quality product. Some of the supporting and technology groups are Printed Circuited Board Group, Antenna Group, Microwave and Millimeter Wave Components Group, Mechanical Engineering Group, LAN, Human Resource Development Group etc. apart from work centers who carryout system design and development activities.

Long-Term self-reliance in Technologies / Systems has been driving principle in its entire development endeavor to make the nation self-reliant and independent.

In house Printed Circuit Board facilities provide faster realization of the digital hardware. Multi-layer Printed Circuit Board fabrication facilities are available to cater for a high precision and denser packaging.

The Antenna Group is responsible for design and development of wide variety of antennas covering a broad electromagnetic spectrum (HF to Millimeter Frequencies). The Group also develops RADOMES, which meet stringent environmental conditions for the EW equipment to suit the platform.

The MMW Group is involved in the design and development of MMW Sub-systems and also various Microwave Components like Solid State Amplifier, Switches, Couplers and Filters using the latest state-of-the–art technology.

The Hybrid Microwave Integrated Circuit Group provides custom-made microwave components and super components in the microwave frequency region using both thin film and thick film technology.

In the Mechanical Engineering Group, the required hardware for EW Systems is designed and developed and the major tasks involved include Structural and Thermal Engineering.

The Technical Information Center, the place of knowledge bank is well equipped with maintained libraries, books, journals, processing etc. Latest Technologies in the electronic warfare around the globe are catalogued and easily accessible.

The Techniques Division of ECM wing is one such work center where design and development of subsystems required for ECM applications are undertaken. ESM Work Centers design and development of DF Rx, Rx Proc etc. for various ESM Systems using state-art-of-the technology by employing various techniques to suit the system requirements by the end users. All the subsystems are designed and developed using microwave, and processor/DSP based Digital hardware in realizing the real time activities in Electronic Warfare

Most of the work centers are connected through DLRL LAN (Local LAN) for faster information flow and multi point access of information critical to the development activities. Information about TIC, stores and general administration can be downloaded easily.

The Human Resource Division play a vital role in conducting various CEP courses, organizing service and technical seminars to upgrade the knowledge of scientists in the laboratory.

DLRL has been awarded ISO 9001:2015 certification for Design and Development of Electronic System of assured quality for Defence Services; utilize advanced and cost-effective technologies & systems on time. DLRL shall comply with the requirements of quality Management Systems with a focus on its continual improvement.

ABSTRACT

Spectrum analysis plays a pivotal role in signal processing, enabling the decomposition of signals into their frequency components for applications ranging from telecommunications to audio processing and radar systems. This work presents a hardware simulation framework for spectrum analysis, developed using Vitis Model Composer, a model-based design tool that bridges high-level algorithm development with hardware implementation. The framework focuses on creating efficient signal processing models tailored for deployment on Xilinx FPGA platforms, ensuring real-time performance while adhering to hardware constraints like resource utilization and latency.

The simulation pipeline includes essential signal processing techniques such as Fast Fourier Transform (FFT), filtering, and windowing, all optimized for FPGA implementation. Vitis Model Composer is utilized to integrate MATLAB/Simulink workflows seamlessly into hardware simulation and verification processes. The proposed approach allows for rapid prototyping, co-simulation, and early-stage design validation, significantly reducing development time and improving accuracy.

A key contribution of this research is the optimization of computational pipelines for spectrum analysis, emphasizing scalability and power efficiency while maintaining precision. The results demonstrate that hardware-accelerated spectrum analysis achieves significant performance improvements compared to software-based implementations. This makes it suitable for real-time applications in wireless communication, medical imaging, and industrial automation.

By leveraging the advanced features of Vitis Model Composer, this study highlights the benefits of combining high-level abstraction with hardware-specific optimization to achieve effective and practical designs. The insights provided here underscore the importance of hardware-based spectrum analysis in modern signal processing, offering a pathway for developing innovative solutions in performance-critical domains.

TABLE OF CONTENTS

1. INTRODUCTION:-

➔ About Project in brief

This project, **“Hardware Simulation for Spectrum Analysis Using Vitis Model Composer,”** focuses on designing and implementing a high-performance, hardware-accelerated system for real-time spectrum analysis. Spectrum analysis is a vital process in signal processing, allowing the identification and interpretation of frequency components within a signal. Its applications span various fields, including wireless communication, radar systems, audio signal processing, and medical imaging. However, traditional software-based methods often face challenges such as high latency, limited scalability, and computational inefficiencies, making them unsuitable for real-time applications. To address these limitations, this project leverages **Vitis Model Composer**, a powerful model-based design tool for developing and simulating FPGA hardware implementations.

The primary objective of the project is to develop a system capable of efficiently processing input signals to extract their frequency spectra. The system employs essential signal processing techniques, including **Fast Fourier Transform (FFT)** for frequency domain conversion, **filtering** to remove noise, and **windowing** to reduce spectral leakage. These operations are implemented using FPGA hardware, ensuring low latency, high throughput, and optimized resource usage.

Vitis Model Composer plays a central role in the design and simulation process, enabling seamless integration of MATLAB/Simulink models with FPGA platforms. The tool’s ability to generate HDL code directly from high-level models simplifies the design workflow, reduces development time, and ensures compatibility with the target hardware. Co-simulation capabilities further allow for real-time validation, ensuring that the hardware implementation matches the desired system performance.

The project also emphasizes optimizing FPGA resources through techniques such as pipelining, parallel processing, and resource sharing. These strategies enable the system to meet real-time processing demands while minimizing power consumption and hardware costs. Performance metrics like latency, accuracy, and resource utilization are evaluated to validate the system.

This hardware-accelerated approach to spectrum analysis has significant potential for practical applications in telecommunications, industrial automation, IoT, and audio processing. By combining the modeling power of Vitis Model Composer with the computational efficiency of FPGA hardware, the project highlights a modern, scalable solution for real-time signal processing challenges. It demonstrates how advanced tools and techniques can address industry demands for high-speed, accurate, and resource-efficient systems, paving the way for further innovations in hardware-accelerated signal processing.

FIELD-PROGRAMMABLE GATE ARRAYS [FPGA]:

Field-Programmable Gate Arrays (FPGAs) are powerful and versatile integrated circuits used in digital electronics. Unlike traditional microprocessors or microcontrollers, FPGAs are reconfigurable hardware, allowing users to define their own custom digital logic and circuitry by programming the device rather than writing software.

FPGAs consist of a large array of programmable logic blocks, interconnects, and memory elements. These logic blocks are essentially collections of digital gates that can be configured to perform various functions. The interconnects provide the means to connect these logic blocks in a highly flexible manner.

FPGAs, or Field-Programmable Gate Arrays, are powerful integrated circuits designed to be highly customizable for a wide range of applications. Unlike traditional CPUs and GPUs, which are fixed in their functionality, FPGAs can be programmed and reconfigured by the user to perform specific tasks efficiently.

At their core, FPGAs consist of an array of logic gates and programmable interconnects, allowing users to create and modify digital circuits. This flexibility makes FPGAs ideal for tasks like signal processing, data acceleration, and hardware prototyping.

Programmers use hardware description languages like VHDL or Verilog to define the desired circuit's behaviour, and then a synthesis tool converts this into a configuration file. This file, known as a bitstream, configures the FPGA to execute the defined logic. This reprogrammability makes FPGAs valuable for rapidly evolving and specialized applications.

FPGAs are commonly used in industries such as telecommunications, robotics, aerospace, and high-performance computing. They are particularly useful in scenarios where speed, power efficiency, and customization are critical, enabling engineers and developers to implement complex algorithms and hardware solutions tailored to their specific needs.

ADVANTAGES OF FPGAs:

- **Flexibility:** FPGAs are highly versatile and can be adapted for various applications, from digital signal processing to image and video processing, communication systems, and more.
- **Parallel Processing:** They excel at parallel processing tasks, making them suitable for real-time and computationally intensive applications.
- **Hardware Acceleration:** FPGAs can accelerate specific tasks by offloading them from traditional CPUs, improving overall system performance.
- **Low Latency:** FPGAs offer low latency, making them ideal for real-time applications.
- **Reconfigurability:** FPGAs can be reprogrammed multiple times, allowing them to adapt to changing requirements and fix issues through remote updates.

DISADVANTAGES OF FPGAs:

- **Complexity:** Designing for FPGAs can be complex, requiring specialized knowledge and tools.

- **Cost:** FPGAs can be expensive compared to off-the-shelf microcontrollers or microprocessors.
- **Power Consumption:** Depending on usage, FPGAs can be power-hungry, especially when running at high clock speeds.
- **Limited Resources:** FPGAs have finite resources, so designing large and complex systems may require higher-end, more costly devices.

TOOLS & SOFTWARES TO PROGRAM AN FPGA:

- Programming a Field-Programmable Gate Array (FPGA) involves a series of steps, and there are several tools and software environments available to aid in this process. These tools are essential for designing, simulating, synthesizing, and configuring FPGA devices. The specific tools one uses can depend on the FPGA manufacturer and one's project's requirements, but some common ones include:

- VHDL or Verilog: Hardware Description Languages (HDLs) are used to describe the functionality of the FPGA. VHDL (VHSIC Hardware Description Language) and Verilog are two of the most widely used HDLs for FPGA development. You write code in one of these languages to define the logic and behavior of your digital circuit.

- **FPGA Synthesis Tools:** These tools take your HDL code and transform it into a netlist that describes the hardware resources used in the FPGA, like lookup tables, flip-flops, and interconnections. Popular synthesis tools include:

→ Xilinx Vivado: For Xilinx FPGAs.
→ Altera Quartus Prime: For Intel (formerly Altera) FPGAs.
→ Synopsys Synplify: Vendor-neutral synthesis tool.

- **Simulation Tools:** Before programming an FPGA, it's crucial to simulate your design to catch potential issues early. Popular simulation tools include:

→ ModelSim: A widely used simulation tool that supports both VHDL and Verilog.
→ Xilinx Vivado Simulator: Comes with the Vivado suite for Xilinx FPGAs.
→ Altera ModelSim: For Intel FPGAs.

- **FPGA Programming Tools:** These tools are used to program the bitstream onto the FPGA device. They may also allow you to configure and manage the device, as well as perform debugging. The choice of tool depends on the FPGA manufacturer:

→ Xilinx Vivado: For Xilinx FPGAs, you can use Vivado to program the bitstream onto the FPGA. Vivado also offers debugging and hardware-in-the-loop (HIL) capabilities.
→ Altera Quartus Prime Programmer: For Intel FPGAs.
→ Lattice Radiant Programmer: For Lattice FPGAs.
→ Xilinx Vitis: If you're targeting Xilinx FPGAs and are developing software-hardware co-designs, Vitis can be used.

- **Third-Party Design Tools:** Depending on your project's complexity, you might also use third-party design tools like MATLAB Simulink or high-level synthesis tools like Xilinx HLS or Vivado HLS to generate FPGA code from high-level algorithmic descriptions.

- **Development Boards and Hardware:** You'll need an FPGA development board to program and test your designs. The board typically includes the FPGA, interfaces, and sometimes additional components like LEDs, buttons, or sensors for testing and prototyping.

- **Text Editors and Integrated Development Environments (IDEs):** While not specific to FPGA development, you'll often use text editors or IDEs to write and manage your HDL code. Common choices include Visual Studio Code, Xilinx SDK, or Quartus Prime.

- **Version Control Tools:** It's crucial to keep track of changes in your code and collaborate with team members. Version control systems like Git are commonly used for this purpose.

LIST OF MANUFACTURERS OF FPGA:

Several manufacturers produce Field-Programmable Gate Arrays (FPGAs), each offering a range of FPGA products with various features and capabilities. Here are some of the prominent FPGA manufacturers:

- **Xilinx (now part of AMD)**: Xilinx is one of the leading FPGA manufacturers. They are known for their Virtex, Kintex, and Spartan families of FPGAs. Xilinx FPGAs are widely used in applications like data centers, telecommunications, aerospace, and automotive industries.

- **Intel (formerly Altera)**: Intel acquired Altera, a major FPGA manufacturer, and now offers FPGA products under the Intel brand. Their Stratix and Arria FPGA families are well-regarded for high-performance and reliability.

- **Lattice Semiconductor:** Lattice Semiconductor produces a range of FPGAs, including their popular ECP and iCE series. They are often chosen for low-power applications, such as mobile devices and IoT.

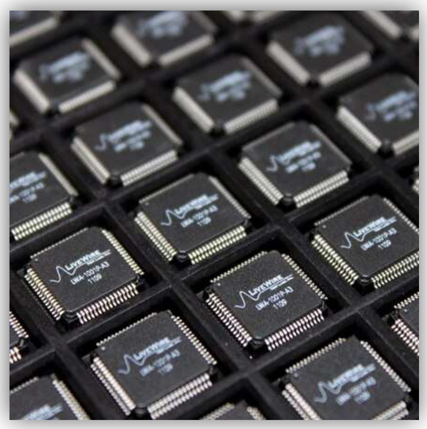
- **Microchip (formerly Microsemi):** Microchip acquired Microsemi, which specializes in FPGAs like the SmartFusion and IGLOO2 series. These FPGAs are often used in industrial, automotive, and communication applications.

- **QuickLogic:** QuickLogic is known for their FPGA and SoC (System-on-Chip) solutions, which are commonly used in consumer electronics, wearables, and IoT devices.

- **Achronix:** Achronix produces FPGAs such as Speedster and Speedcore, designed for high-performance and low-latency applications like networking and data acceleration.

- **Aldec, Inc:** Aldec offers FPGA development boards and simulation tools, in addition to their HES (Hardware Emulation Solution) series, which caters to hardware acceleration and emulation.

- **Enclustra:** Enclustra focuses on FPGA and SoC modules, targeting a wide range of applications, from embedded systems to high-performance computing.



- **Maxim Integrated (formerly Terasic):** Maxim Integrated produces FPGA development kits and platforms, including the popular Terasic DE-series development boards.

- **Actel (acquired by Microsemi):** While Actel itself no longer exists as an independent entity, some of their FPGA technology has been incorporated into Microsemi's product lineup.

APPLICATION-SPECIFIED INTEGRATED CIRCUITS [ASIC]:

An Application-Specific Integrated Circuit (ASIC) is a highly specialized type of microchip designed to perform a specific function or a closely related set of functions within an electronic system. Unlike general-purpose microprocessors or microcontrollers, which can handle a wide range of tasks, ASICs are tailored to excel at a single, predetermined task. This customization allows for maximum efficiency, performance, and cost-effectiveness in the context of the intended application.

ASICs are created by semiconductor engineers and designers who carefully craft the chip's architecture, logic, and physical layout to meet the unique requirements of the target application. This level of customization can lead to faster operation and more power-efficient performance compared to off-the-shelf components.

ASICs are commonly used in a variety of industries, including telecommunications, automotive, aerospace, consumer electronics, and industrial applications. For example, in a smartphone, ASICs may handle specific tasks like image processing or signal modulation. In network routers, ASICs manage data packet forwarding. In the automotive industry, they control engine functions or facilitate safety features. In each case, ASICs optimize performance, reduce power consumption, and often lead to cost savings.

The development of ASICs can be time-consuming and expensive due to the need for extensive design and fabrication processes. However, these costs are often justified by the benefits they offer in high-volume, specialized applications where efficiency and performance are paramount.

ADVANTAGES OF ASICs:

- **Optimized Performance:** ASICs are specifically designed for a single function or set of related functions, allowing them to outperform general-purpose processors in their dedicated tasks. This results in faster execution and better efficiency.

- **Power Efficiency**: Due to their specialization, ASICs consume less power compared to general-purpose CPUs. This makes them ideal for battery-powered devices and energy-efficient applications.

- **Cost-Effective at Scale:** In high-volume production, ASICs can be cost-effective. By eliminating unnecessary features and components, they reduce manufacturing and material costs, particularly when compared to using multiple general-purpose components to achieve the same functionality.

- **Compact Size:** ASICs can be designed to be very compact, which is essential in applications where space is limited, such as in mobile devices, embedded systems, and wearables.

- **Security:** In applications where security is critical, ASICs can be designed to provide robust hardware security features, making them resistant to attacks or tampering.

DISADVANTAGES OF ASICs:

- **High Development Cost:** Developing ASICs is expensive and time-consuming. It involves specialized expertise, the creation of custom designs, and manufacturing masks. This upfront investment can be prohibitive for small-scale projects or prototypes.

- **Lack of Flexibility:** ASICs are inflexible and can only perform the specific tasks they were designed for. If requirements change or the chip needs to be reconfigured, it may require a new ASIC design, incurring additional costs and time.

- **Long Lead Times:** The design and manufacturing process for ASICs can have long lead times. This can be problematic in industries with rapidly changing technologies or market demands.

- **Risk of Obsolescence:** Rapid advancements in technology can make ASICs obsolete faster than general-purpose components. When technology evolves, an ASIC may be too specialized to adapt to new requirements.

- **Limited Prototyping and Testing:** Due to the costs involved in ASIC development, prototyping and testing are challenging and costly. This can make it difficult to refine the design before committing to large-scale production.

TOOLS & SOFTWARES TO PROGRAM AN ASIC:

Programming an Application-Specific Integrated Circuit (ASIC) involves designing and configuring the chip's logic and functionality. This process typically requires specialized tools and software to create the ASIC design and then program it onto the physical hardware. Here are some of the essential tools and software commonly used in ASIC programming:

- **Electronic Design Automation (EDA) Tools:**

- **Synthesis Tools:** These tools take a high-level hardware description language (HDL) such as VHDL or Verilog and convert it into a gate-level representation that can be used to program the ASIC.

- **Place-and-Route Tools:** These tools determine the physical layout of the ASIC, placing logic gates and routing connections on the chip to optimize for performance, power, and size.

- **Simulation Tools:** Before actual fabrication, designers use simulation tools to test the ASIC's behavior and functionality in a virtual environment. This helps catch design errors and verify correct operation.

- **Static Timing Analysis (STA) Tools:** STA tools ensure that signals meet the required timing constraints and that the ASIC will operate correctly at the desired clock frequency.

- **Programming and Verification Tools:**

  - **Programming Software:**

  To program the synthesized ASIC design onto the physical chip, you'll need specialized programming software and hardware programmers, often provided by the ASIC manufacturer.

  - **JTAG Tools:**

  Joint Test Action Group (JTAG) interfaces are used for programming and debugging ASICs. JTAG tools allow for boundary scan testing, debugging, and in-system programming.

- **Hardware Description Languages (HDLs):**

  - **VHDL and Verilog:**

  These are the most common hardware description languages used for specifying the behavior and structure of the ASIC. Designers write code in these languages to describe the ASIC's logic and functionality.

- **ASIC Design Environment:**
  - **Integrated Development Environments (IDEs):**

  Some vendors offer specialized IDEs for ASIC design, incorporating various tools into a unified environment for efficient development.

- **Foundry-Specific Tools:**
- Each semiconductor foundry may provide proprietary design and verification tools tailored to their manufacturing process. These tools are used to ensure that the ASIC design is compatible with the foundry's technology.

- **Third-Party IP (Intellectual Property) Blocks:**
- Some ASIC designs incorporate third-party IP blocks, such as processors or memory modules. Tools for integrating and verifying these IP blocks are essential.

- **Design and Version Control Software:**

- Design teams often use version control software like Git to manage and track changes in the ASIC design, ensuring collaboration and maintaining a history of revisions.

- **Documentation and Design Flow Management:**
- Tools for managing design documentation and the overall design flow are crucial for ensuring a structured and efficient design process.

LIST OF MANUFACTURERS OF ASIC:

- **TSMC (Taiwan Semiconductor Manufacturing Company):**
- TSMC is one of the largest semiconductor foundries globally, providing ASIC manufacturing services for a broad customer base.
- They offer advanced process technologies and a wide range of services, from design and prototyping to high-volume production.

- **GlobalFoundries:**
- GlobalFoundries is another prominent semiconductor foundry that offers ASIC manufacturing services.
- They provide advanced technology nodes and customization options, supporting various industries, including consumer electronics and automotive.

- **Samsung Foundry:**
- Samsung Foundry, a part of Samsung Electronics, offers ASIC manufacturing services with access to their advanced semiconductor manufacturing processes.
- They serve a diverse customer base, including clients in AI, 5G, and IoT applications.

- **Intel Custom Foundry:**
- Intel Custom Foundry provides ASIC manufacturing services, leveraging Intel's advanced process technologies and manufacturing capabilities.
- They focus on various applications, such as networking, artificial intelligence, and high-performance computing.

- **Global ASIC Manufacturers:**

- Apart from foundries, there are companies like Xilinx, Microchip (formerly Atmel), and ON Semiconductor, known for offering both ASIC design and manufacturing services.
- These companies often have a portfolio of standard components and IP cores for customers to use in their ASIC designs.

- **Specialized ASIC Manufacturers:**
- Some companies specialize in providing ASIC design and manufacturing services for specific industries or applications. For example:

- ASIC North specializes in ASIC solutions for medical devices.
- Bitmain focuses on ASICs for cryptocurrency mining.
- Silicon Creations offers custom analog and mixed-signal ASIC design services.

- **Fabless Semiconductor Companies:**
- Many fabless semiconductor companies, such as NVIDIA, Qualcomm, and Broadcom, design ASICs for their products and then outsource the manufacturing to foundries.

- **Startups and Boutique ASIC Design Houses:**
- There is a growing number of small ASIC design firms and startups that cater to niche markets and provide specialized ASIC design services.

CORE DESCRIPTION FOR HARDWARE SIMULATION

Hardware simulation is a critical step in the design and development of electronic systems, allowing for the virtual modeling, testing, and optimization of hardware components or full systems before physical prototyping. This process ensures accuracy, efficiency, and robustness while reducing development costs and time-to-market.

Objectives

- **Functional Validation:** Ensures the hardware meets the design specifications and performs as intended across all operating scenarios.
- **Performance Evaluation:** Assesses the system's speed, throughput, latency, and power consumption under varying workloads.
- **Timing Verification:** Simulates clock domain interactions, propagation delays, and synchronization issues to avoid timing violations.
- **Fault Tolerance Analysis:** Identifies vulnerabilities to hardware faults and ensures the design’s ability to recover or handle failures gracefully.
- **Scalability Testing:** Evaluates how the hardware can scale with increasing workloads or additional components.

Simulation Workflow

1. **Model Creation:** Develop hardware models using Hardware Description Languages (HDLs) like Verilog, VHDL, or SystemVerilog.
2. **Testbench Development:** Build test environments to apply inputs and verify outputs against expected results.
3. **Simulation Execution:** Use simulation tools (e.g., ModelSim, Xilinx Vivado, Synopsys VCS) to evaluate behavior under various conditions.

4. **Result Analysis:** Analyze waveforms, debug failures, and optimize designs for better performance and reliability.

Key Features

- **Behavioral Simulation:** Validates logic and functionality without considering timing constraints.

- **Gate-Level Simulation:** Includes timing delays to verify the design post-synthesis or after layout.
- **System-Level Simulation:** Integrates multiple components (e.g., processors, memory, I/O) to verify system interactions.

- **Hardware/Software Co-Simulation:** Simultaneously tests hardware designs and embedded software for seamless integration.

Applications

- **ASIC and FPGA Design:** Ensures the correctness of custom chips and reconfigurable logic devices like the XCVU2104.
- **Embedded Systems:** Simulates processors, communication protocols, and peripherals in IoT, automotive, and aerospace applications.
- **Signal Processing:** Models complex algorithms for audio, video, and communication systems.
- **Power and Thermal Analysis:** Estimates energy consumption and heat dissipation to optimize for efficiency.

Software Required

Xilinx Tools

Xilinx is a Semiconductor company that primarily supplies Programmable Logic Devices. The company is known for the first commercially viable Field Programmable Gate Array (FPGA) and creating the first fabless manufacturing model. Xilinx sells a broad range of FPGAs, Complex Programmable Logic Devices (CPLDs), Design Tools, Intellectual Property and Reference Designs. The most popular families of the Xilinx FPGAs are Virtex, Kintex, Artix, Zynq, Spartan etc. Xilinx Provide different tools for different applications like for Embedded application we use Xilinx EDK, for Logic Based applications we use **I**ntegrated **S**ynthesis **E**nvironment and for Digital Signal Processing we use DSP Generator integrated with Matlab.

Xilinx ISE

Xilinx ISE (Integrated Synthesis Environment) is a discontinued software tool form Xilinx for Synthesis and Analysis of HDL designs, which primarily targets development of embedded firmware for Xilinx FPGA and CPLD integrated circuit (IC) product families. It was succeeded by Xilinx Vivado. ISE enables the developer to synthesis their designs, perform timing analysis, examine RTL diagrams, simulate a design’s reaction to different stimuli, and configure the target devise with the programmer. Other components shipped with the Xilinx ISE include the Embedded Development Kit (EDK), a Software Development Kit (SDK) and ChipScope Pro. The Xilinx ISE is primarily used for circuit synthesis and design, while ISIM or the Model-Sim logic simulator is used for system-level testing

Xilinx EDK (Embedded Development Kit)

Xilinx EDK (Embedded Development Kit) is a software suite provided by Xilinx, a leading semiconductor company specializing in programmable logic devices (FPGAs) and system-on-chip (SoC) solutions. EDK is specifically designed to support the development of embedded systems using Xilinx FPGAs and SoCs. The Xilinx EDK includes several key components that facilitate the development process:

- Xilinx Platform Studio (XPS)
- Software Development Kit (SDK)

Overall, the Xilinx EDK provides a comprehensive set of tools, IP cores, and development environments to streamline the process of developing embedded systems using Xilinx FPGAs and SoCs. It enables hardware and software designers to collaborate effectively and efficiently to build customized embedded solutions.

Xilinx Platform Studio (XPS)

Xilinx Platform Studio (XPS) is a key component of ISE Embedded Edition Design Suite, helping the hardware designer to easily build, connect and configure embedded processor-bases systems. XPS employs graphical design views and sophisticated correct-by-design wizards to guide developers through the steps necessary to create custom processor system within minutes. The true potential of XPS emerges with its ability to configure and integrate plug and play IP cores from the AMD Embedded IP catalog, with custom or 3rd party Verilog or VHDL designs. Firmware and software developers benefit from XPS integration with SDK which allows the automatic generation of critical system software such as boot loaders, bare metal BSP, and Linux BSPs.

Software Development Kit (SDK)

The Software Development Kit (SDK) is the Integrated Design Environment for creating embedded applications on any of the microprocessors from AMD: Zynq UltraScale+ MPSoC, Zynq 7000 SoCs, and the industry-leading MicroBlaze soft-core microprocessor. The SDK is the first application IDE to deliver true homogenous and heterogeneous multi-processor design, debug, and performance analysis. Benefits include:

- Zynq UltraScale+ MPSoC, Zynq 7000 SoCs, and MicroBlaze support
- Included with the Vivado Design Suite or available as a separate free download for embedded software developers
- Based on Eclipse 4.5.0
- Complete Integrated Design Environment (IDE) that directly interfaces to the Vivado embedded hardware design environment
- Editor, compilers, build tools, flash memory management, and JTAG debug integration

Xilinx Vivado

The Vivado Design Suite is a powerful FPGA design tool from Xilinx that is often used in conjunction with EDK. Vivado provides advanced capabilities for system-level design, synthesis, place-and-route, and bitstream generation for Xilinx FPGAs and SoCs. Vivado Design Suite is a software suite produced by Xilinx for synthesis and analysis of hardware description language (HDL) designs, superseding Xilinx ISE with additional features for system on a chip development and high-level synthesis. Like the later versions of ISE, Vivado includes the in-built logic simulator. Vivado also introduces high-level synthesis, with a toolchain that converts C code into programmable logic. Vivado supports Xilinx's 7-series and all the newer devices (UltraScale and UltraScale+ series). For development targeting older Xilinx's devices and CPLDs, the already discontinued Xilinx ISE has to be used.

VITIS MODEL COMPOSER

Vitis Model Composer is a cutting-edge design tool from Xilinx that enables efficient hardware modeling, simulation, and implementation for FPGA platforms. Integrated seamlessly with MATLAB® and Simulink®, it allows designers to use a model-based approach to develop, simulate, and verify complex systems such as signal processing, communications, and control systems. By providing a comprehensive library of prebuilt blocks for mathematical operations, logic, signal processing, and memory functions, Vitis Model Composer simplifies the creation of high-level hardware designs directly within the Simulink environment.

One of its standout features is its ability to automatically generate optimized HDL (Hardware Description Language) code from the model, eliminating the need for manual coding and reducing design complexity. This accelerates development cycles while ensuring robust and accurate implementations. Additionally, Vitis Model Composer supports hardware co-simulation, enabling real-time validation of the model on FPGA hardware, ensuring that the design performs as intended. Advanced optimization techniques such as pipelining, parallelism, and resource sharing are integrated into the tool, helping designers maximize FPGA resource efficiency while meeting performance requirements.

The tool also integrates with Xilinx's broader ecosystem, including Vivado and Vitis HLS, to provide a seamless workflow for hardware design and deployment. Its capabilities are widely applicable across industries, from signal processing and communication systems to control systems and machine learning applications.

In this project, Vitis Model Composer is utilized to design and simulate a hardware-based spectrum analysis system, demonstrating its ability to bridge high-level algorithm design with efficient hardware implementation. By streamlining the workflow and optimizing the design for FPGA deployment, the tool enables real-time, resource-efficient spectrum analysis. Its ease of use, speed, and precision make it an essential tool for modern FPGA-based system development.

LEARNING VIVADO

About Vivado and VHDL

VHDL code is a program that describes the logic function of any digital circuit using the VHDL language. The full form of VHDL is "VHSIC Hardware Description Language." VHSIC stands for "Very High-Speed Integrated Circuit." VHDL is a programming language used to describe the structure and behaviour of digital logic circuits. The VHDL code requires importing the libraries IEEE.STD_LOGIC_1164 and IEEE.STD_LOGIC_ARITH.

The Entity is used to specify the input and output ports of the circuit. An Entity usually has one or more ports that can be inputs (in), outputs (out), input-outputs (inout), or buffer. An Entity may also include a set of generic values that are used to declare properties of the circuit.

The Architecture statement describes the underlying functionality of the entity. A single VHDL entity shall have at least one architecture. It is possible to have more than one architecture for the same entity. Architecture is always related to an entity and describes the behaviour of that entity. An architecture can be written in one of the three basic coding styles:

- Dataflow Modeling
- Behavioral Modeling
- Structural Modeling

The difference between these styles is based on the type of concurrent statements used:

- A dataflow architecture uses only concurrent signal assignment statements.
- A behavioral architecture uses only process statements
- A structural architecture uses only component instantiation statements.

In VHDL, the process is the key structure in behavioral modeling. It is the only means by which the executable functionality of a component is defined. All components in the model must be defined using one or more processes for the model to be capable of being simulated. The process statement contains sequential statements and is only permitted inside an architecture.

When programming an FPGA through software such as Xilinx's Vivado, we need to inform the software what physical pins on the FPGA that we plan on using or connecting to in relation to the HDL code. One of the three types of constraint files in FPGA design, the user constraint file (. UCF). The file extension for a Xilinx Vivado constraint file is. XDC (Xilinx Design Constraints). All constraints added to the design are saved in the target XDC file.

Testbenches consist of non-synthesizable VHDL code which generate inputs to the design and checks that the outputs are correct. Since testbenches are used for simulation purpose only (not for synthesis), therefore full range of VHDL constructs can be used e.g., keywords 'assert', 'report' and 'for loops' etc. can be used for writing testbenches.

Component is a reusable VHDL module which can be declared with in another digital logic circuit using Component declaration of the VHDL Code. Instead of coding a complex design in single VHDL Code. we can divide the code in to sub modules as component and combine them using Port Map technique. VHDL Port Map is the Process of mapping the input/ Output Ports of Component in Main Module.

Simulation is a process of emulating real design behaviour in a software environment. Simulation helps verify the functionality of a design by injecting stimulus and observing the design outputs. The process of simulation includes:

- Creating test benches, setting up libraries and specifying the simulation settings for Simulation
- Generating a Netlist (if performing post-synthesis or post-implementation simulation)

RTL analysis in Vivado is an essential step in the design process of a digital circuit using Xilinx FPGAs. It involves analysing the Register Transfer Level (RTL) description of the circuit to ensure its correctness, performance, and adherence to design specifications. Vivado provides several RTL analysis features that help in validating the design before synthesis and implementation.

Vivado's RTL Viewer allows you to visualize the RTL description of your design, including the interconnections between modules, signal paths, and hierarchical structures. Vivado generates various reports to analyze the RTL design. The reports include information about the design hierarchy, module utilization, critical path analysis, timing constraints, and much more.

Synthesis is the process of transforming an RTL-specified design into a gate-level representation. Vivado synthesis is timing-driven and optimized for memory usage and performance. The Xilinx Vivado Design Suite enables implementation of UltraScale FPGA and Xilinx 7 series FPGA designs from a variety of design sources, including:

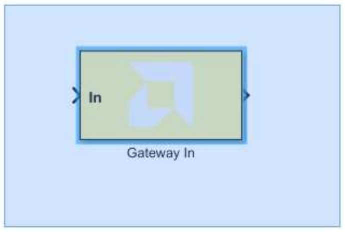
- RTL designs
- Netlist designs
- IP-centric design flows

Vivado implementation includes all steps necessary to place and route the netlist onto device resources, within the logical, physical, and timing constraints of the design. The Vivado Design Suite implementation is a timing-driven flow. The Vivado Design Suite implementation process transforms a logical netlist and constraints into a placed and routed design, ready for bitstream generation.

In Vivado, the process of generating a bitstream involves converting your RTL design, synthesized and implemented in Vivado, into a binary file that can be loaded onto a Xilinx FPGA or programmed into a Xilinx SoC (System-on-Chip) device. The bitstream file contains the configuration data necessary to define the behavior and interconnections of the FPGA's programmable logic elements. After successfully implementing the design and applying the necessary constraints, we can proceed to generate the bitstream.
Once the bitstream file is generated, it needs to be programmed onto the target FPGA or SoC device. This can be done using various methods depending on the device and the development board we are using. It typically involves connecting the board to our computer and using tools like Xilinx's Vivado Hardware Manager or Xilinx SDK to program the bitstream onto the device.

2.XILINX TOOLS FOR HARDWARE SIMULATION:

➔ Gateway In

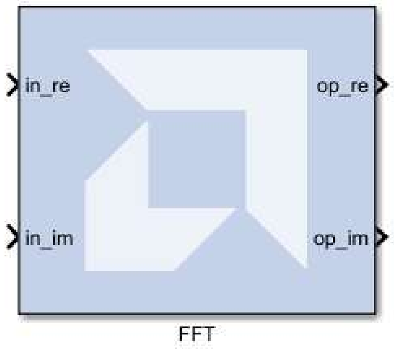


The **Gateway In** block in the Vitis Model Composer toolbox serves as the input interface for your FPGA design. Its key functions include:

1. **Signal Input**: Introduces signals from the Simulink environment into the hardware design.
2. **Data Formatting**: Converts Simulink signals into hardware-compatible formats such as fixed-point or bit-accurate representations.
3. **Design Partitioning**: Marks the boundary between Simulink simulation and FPGA/HDL synthesis.

It ensures your Simulink model's input signals are properly prepared for hardware implementation.

➔ _FFT



The **FFT (Fast Fourier Transform) Toolbox** in the Vitis Model Composer provides optimized blocks to perform efficient FFT operations for FPGA or SoC designs. These blocks are specifically designed for high-performance signal processing and are hardware-optimized.

Key Features:

1. **FFT Block**:
    - Implements the Fast Fourier Transform algorithm.
    - Supports fixed-point and floating-point data formats.
    - Configurable transform length, scaling, and precision settings.
    - Optimized for Xilinx FPGA architectures for low-latency and high throughput.

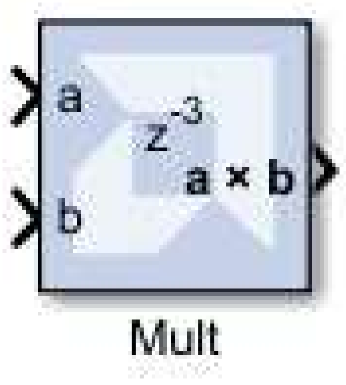
2. **IFFT Block**:
    - Performs the Inverse Fast Fourier Transform.
    - Configurable to match FFT settings for seamless integration.
3. **Parameter Tuning**:
    - Allows adjustment of parameters like:
        - Transform length.
        - Number of channels.
        - Data precision (bit-width).
    - Enables balancing between resource usage and performance.
4. **Applications**:
    - Ideal for signal processing tasks such as spectral analysis, filtering, OFDM communication, and radar systems.

Workflow:

- Drag and drop the **FFT/IFFT blocks** into your Simulink design.
- Configure the block parameters as per your application's requirements.
- Integrate with other Vitis Model Composer blocks for simulation and hardware deployment.

The FFT Toolbox accelerates the development of DSP applications by leveraging Xilinx's hardware acceleration capabilities.

➔ Multiplier



The **Multiplier** block in the Vitis Model Composer toolbox is a hardware-optimized block used to perform efficient multiplication operations in FPGA or SoC designs.

Key Features:

1. **High-Performance Multiplication**:
    - Performs multiplication of input signals with minimal latency.
    - Supports fixed-point, integer, and floating-point data types.
2. **Resource Optimization**:
    - Automatically maps the multiplication operation to the DSP slices available on Xilinx FPGAs, ensuring efficient resource utilization.

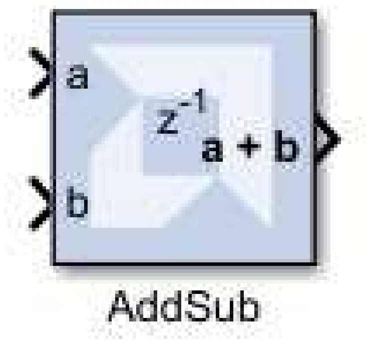
3. **Configurable Parameters**:
   - Allows you to set bit-widths, rounding modes, and saturation options for precise control over the operation.
   - Supports signed and unsigned multiplication.
4. **Pipelining Support**:
   - Enables the use of pipelining to improve performance by increasing clock speed and throughput.
5. **Applications**:
   - Widely used in digital signal processing, image processing, matrix operations, and custom arithmetic functions.

Workflow:

- Drag the **Multiplier block** into your Simulink design.
- Configure the inputs, precision, and pipelining settings.
- Integrate it with other blocks to simulate and deploy hardware-accelerated designs.

The **Multiplier block** ensures fast and efficient multiplication operations, critical for high-performance FPGA-based applications.

➔ Add/Sub



The **Adder** block in the Vitis Model Composer toolbox is a hardware-optimized block used for efficient addition operations in FPGA or SoC designs.

Key Features:

1. **High-Speed Addition**:
   - Performs addition of multiple input signals with minimal delay.
   - Supports fixed-point, integer, and floating-point data formats.
2. **Resource-Efficient Design**:
   - Optimized for FPGA architectures to minimize logic resource usage.
3. **Configurable Parameters**:
   - Allows customization of input bit-width, precision, rounding, and saturation modes.
   - Supports signed and unsigned data types.
   - Configurable to handle multiple inputs.
4. **Pipelining Support**:
   - Provides options for pipelining to achieve higher clock speeds and throughput.
5. **Applications**:
   - Used in arithmetic operations, filters, accumulators, and signal processing tasks.

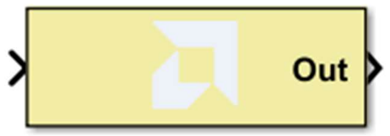
Workflow:

- Drag and drop the **Adder block** into your Simulink model.
- Configure input data types, bit-widths, and arithmetic settings.

- Integrate it with other blocks to perform addition operations as part of your FPGA design.

The **Adder block** ensures efficient and accurate addition operations, enabling seamless integration into hardware-accelerated designs.

➔ Gateway out



Gateway Out

The Gateway Out block in the Vitis Model Composer toolbox is used to define the output interface of your FPGA or hardware design. It serves as the endpoint for signals exiting the hardware design back into the Simulink simulation environment.

Key Features:

1. Signal Output:
   - Transfers hardware-processed signals from the FPGA design back into the Simulink model.

2. Data Type Conversion:
   - Converts hardware-compatible formats (fixed-point, bit-accurate representations) into Simulink-supported data types for visualization or further processing.

3. Boundary Definition:
   - Marks the output boundary between hardware design and Simulink simulation.

4. Applications:
   - Used in FPGA co-simulation workflows to observe results.
   - Essential for validating hardware outputs against Simulink models.

Workflow:
- Place the *Gateway Out* block at the output of your FPGA design in Simulink.
- Ensure the data type matches the requirements of your Simulink model.
- Use it to visualize and verify hardware outputs in your simulation environment.

The Gateway Out block ensures seamless integration between your FPGA design and Simulink, enabling effective testing and validation.

3.PROJECT WORK

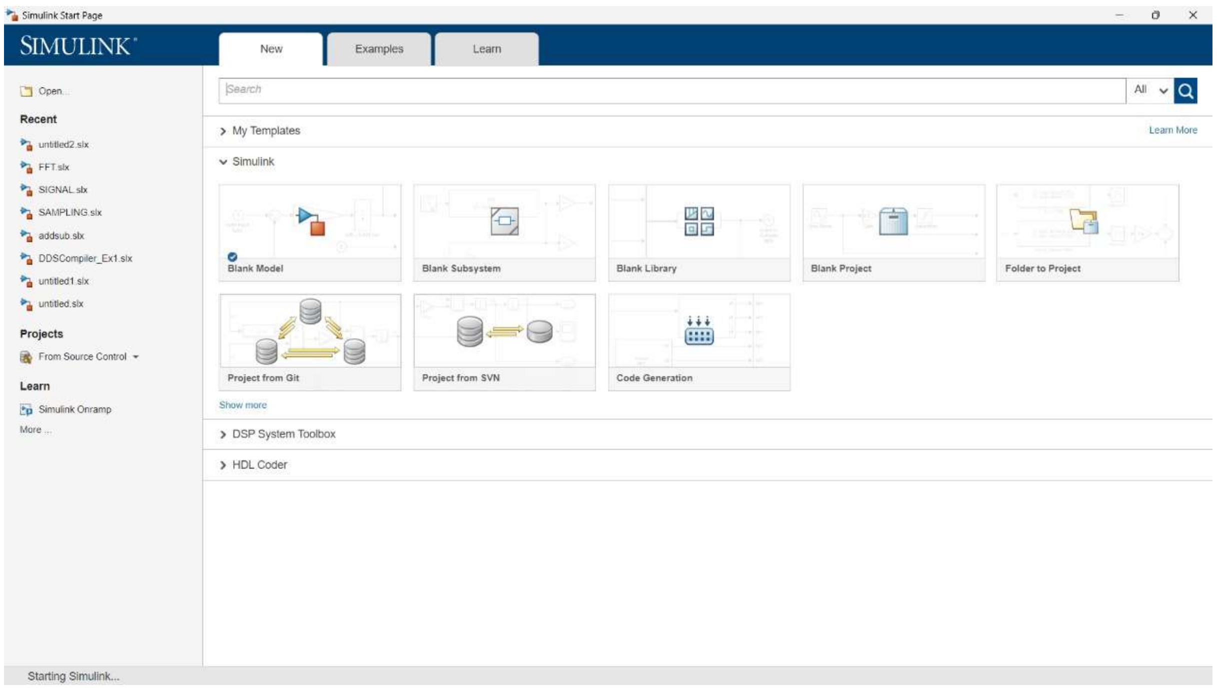
Aim:- Implementation Of Hardware Simulation for Spectrum Analysis Using Vitis Model Composer

Software Required:- Vitis Model Composer, Simulink

Tools Required:- Xilinx Tools And Simulink Tools

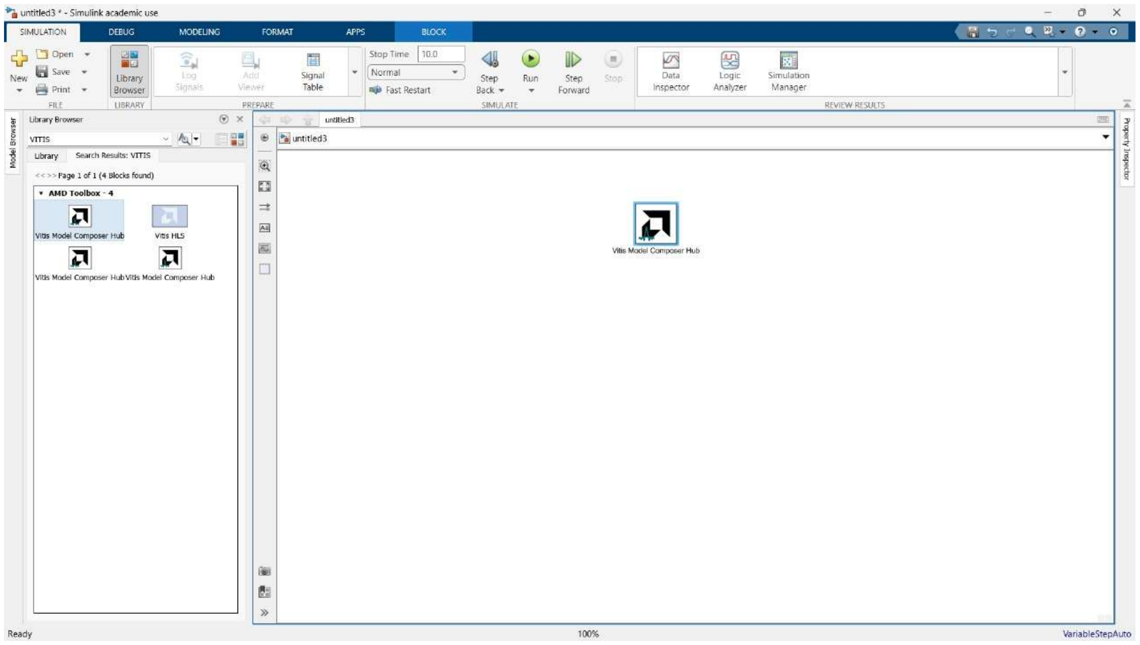
Procedure: -

- **Step 1:-** Create your file on the Desktop
  This makes to import the hardware to Xilinx software Development kit.
- Open the Vivado Model Composer
- Click on Simulink

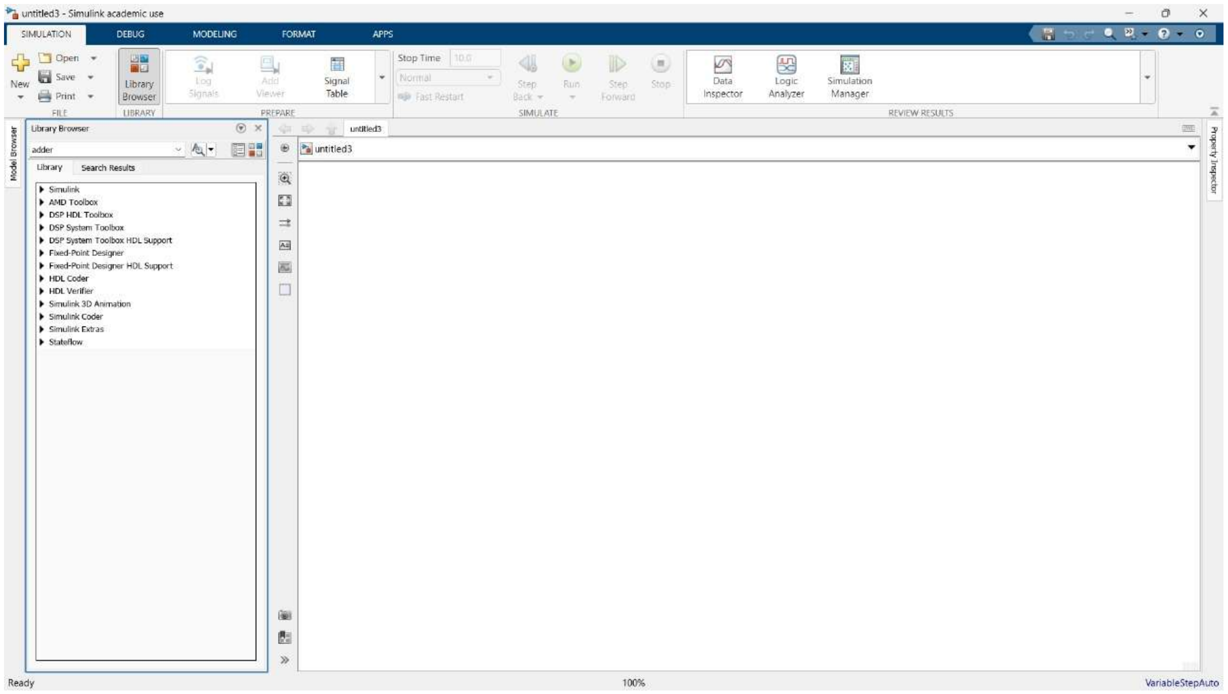


- **Step 2:-**
- Click on the Blank Model

- Open the Library Browser ->Next
- In the library browser search box type VITIS Model Composer Hub and click on the toolbox and drag into your Simulink model.

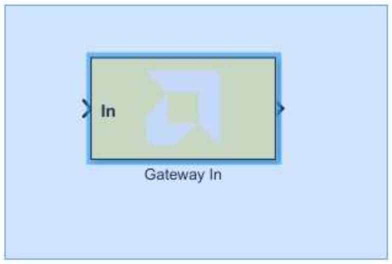


- The **Vitis Model Composer Hub Toolbox** in Simulink enables you to design, simulate, and generate code for FPGA and SoC designs. It provides blocks to connect your Simulink model to Xilinx hardware by:
- **Integrating Hardware and Software**: Allows co-simulation between Simulink and Xilinx platforms.

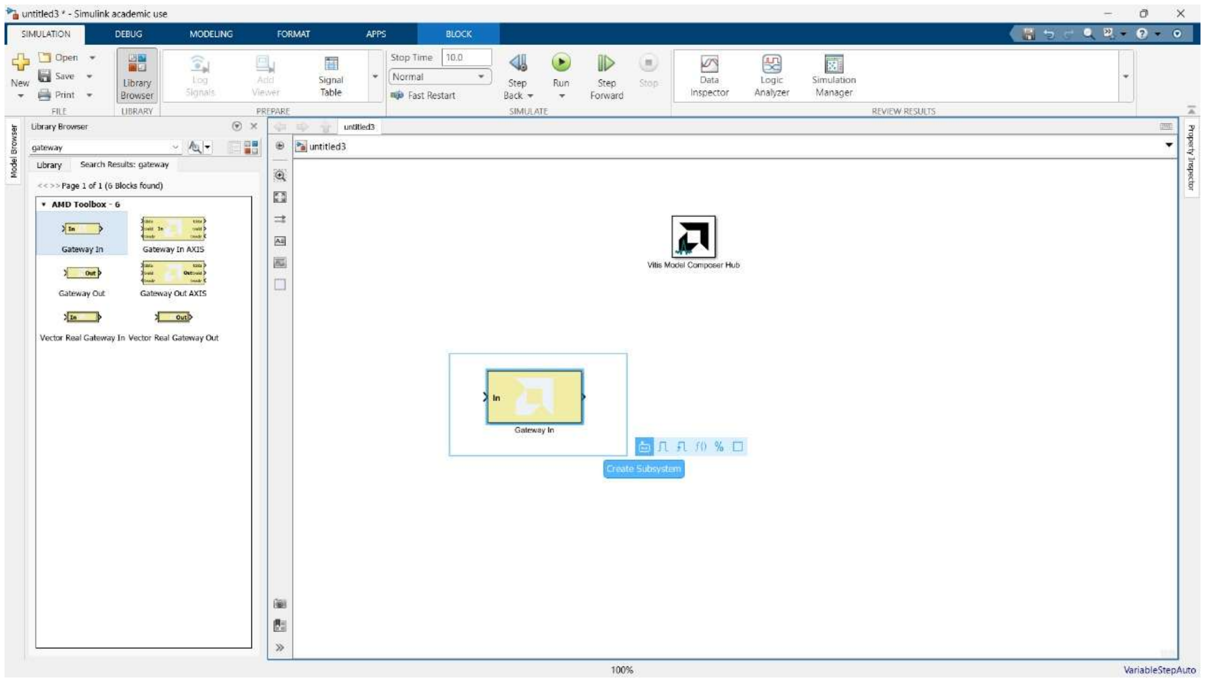


- **Code Generation**: Generates optimized HDL code and integrates with Vitis for deployment.
- **Hardware-in-the-Loop (HIL)**: Enables testing designs directly on FPGA hardware.
- It acts as a bridge between MATLAB/Simulink models and Xilinx's Vitis toolchain for high-performance embedded design.

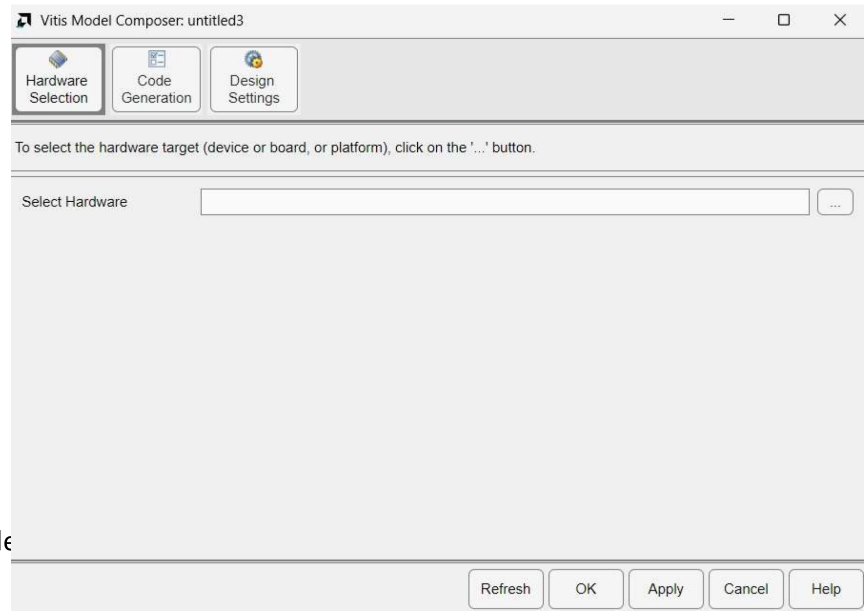
- **Step 3:-**
- Drag the Gateway In toolbox from your Library Browser



- **Step 4:-**
- Create a subsystem on the gateway in toolbox by right clicking on the mouse and drag on to the gateway in toolbox on your Simulink model and select the create subsystem option.



- **Step 5:-**
- Click on the VITIS Model Composer Hub toolbox on your Simulink model

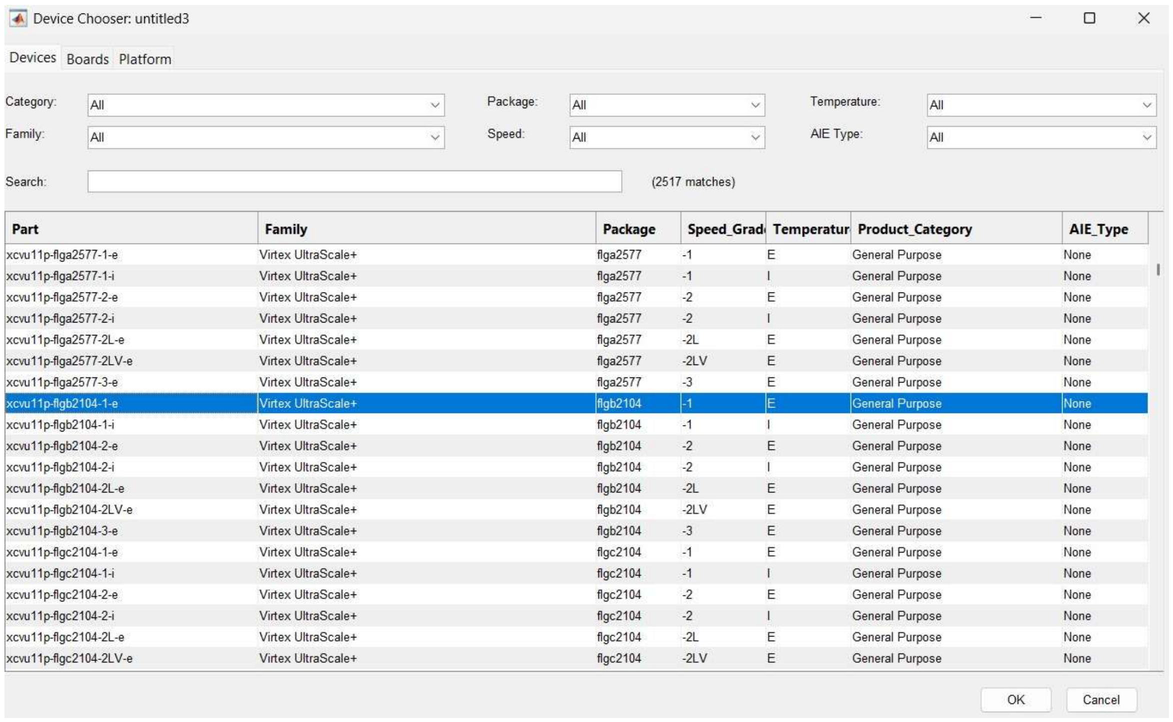


Vitis Model Composer: untitled3

To select the hardware target (device or board, or platform), click on the '...' button.

Select Hardware

- sele



Device Chooser: untitled3

Category: All

Family: All

Package: All

Speed: All

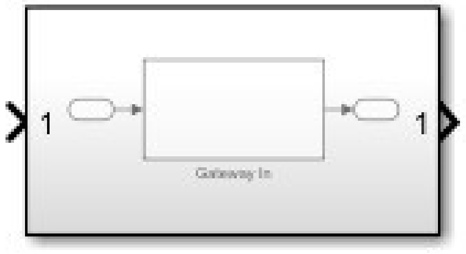
Temperature: All

AIE Type: All

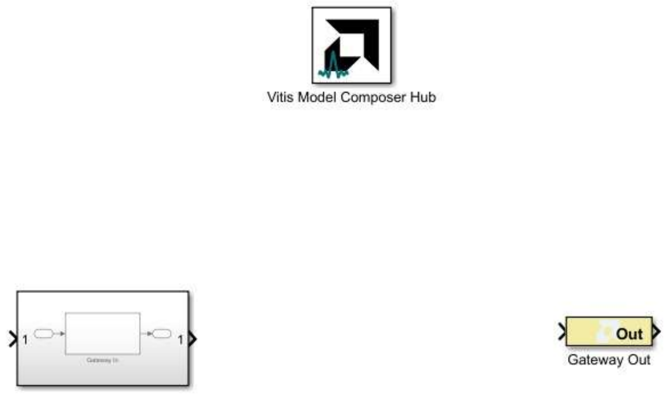
Search: (2517 matches)

Part	Family	Package	Speed_Grad	Temperatur	Product_Category	AIE_Type
xcvu11p-flga2577-1-e	Virtex UltraScale+	flga2577	-1	E	General Purpose	None
xcvu11p-flga2577-1-i	Virtex UltraScale+	flga2577	-1	I	General Purpose	None
xcvu11p-flga2577-2-e	Virtex UltraScale+	flga2577	-2	E	General Purpose	None
xcvu11p-flga2577-2-i	Virtex UltraScale+	flga2577	-2	I	General Purpose	None
xcvu11p-flga2577-2L-e	Virtex UltraScale+	flga2577	-2L	E	General Purpose	None
xcvu11p-flga2577-2LV-e	Virtex UltraScale+	flga2577	-2LV	E	General Purpose	None
xcvu11p-flga2577-3-e	Virtex UltraScale+	flga2577	-3	E	General Purpose	None
xcvu11p-flgb2104-1-e	Virtex UltraScale+	flgb2104	-1	E	General Purpose	None
xcvu11p-flgb2104-1-i	Virtex UltraScale+	flgb2104	-1	I	General Purpose	None
xcvu11p-flgb2104-2-e	Virtex UltraScale+	flgb2104	-2	E	General Purpose	None
xcvu11p-flgb2104-2-i	Virtex UltraScale+	flgb2104	-2	I	General Purpose	None
xcvu11p-flgb2104-2L-e	Virtex UltraScale+	flgb2104	-2L	E	General Purpose	None
xcvu11p-flgb2104-2LV-e	Virtex UltraScale+	flgb2104	-2LV	E	General Purpose	None
xcvu11p-flgb2104-3-e	Virtex UltraScale+	flgb2104	-3	E	General Purpose	None
xcvu11p-flgc2104-1-e	Virtex UltraScale+	flgc2104	-1	E	General Purpose	None
xcvu11p-flgc2104-1-i	Virtex UltraScale+	flgc2104	-1	I	General Purpose	None
xcvu11p-flgc2104-2-e	Virtex UltraScale+	flgc2104	-2	E	General Purpose	None
xcvu11p-flgc2104-2-i	Virtex UltraScale+	flgc2104	-2	I	General Purpose	None
xcvu11p-flgc2104-2L-e	Virtex UltraScale+	flgc2104	-2L	E	General Purpose	None
xcvu11p-flgc2104-2LV-e	Virtex UltraScale+	flgc2104	-2LV	E	General Purpose	None

- Subsystem of the gateway in toolbox looks like this.

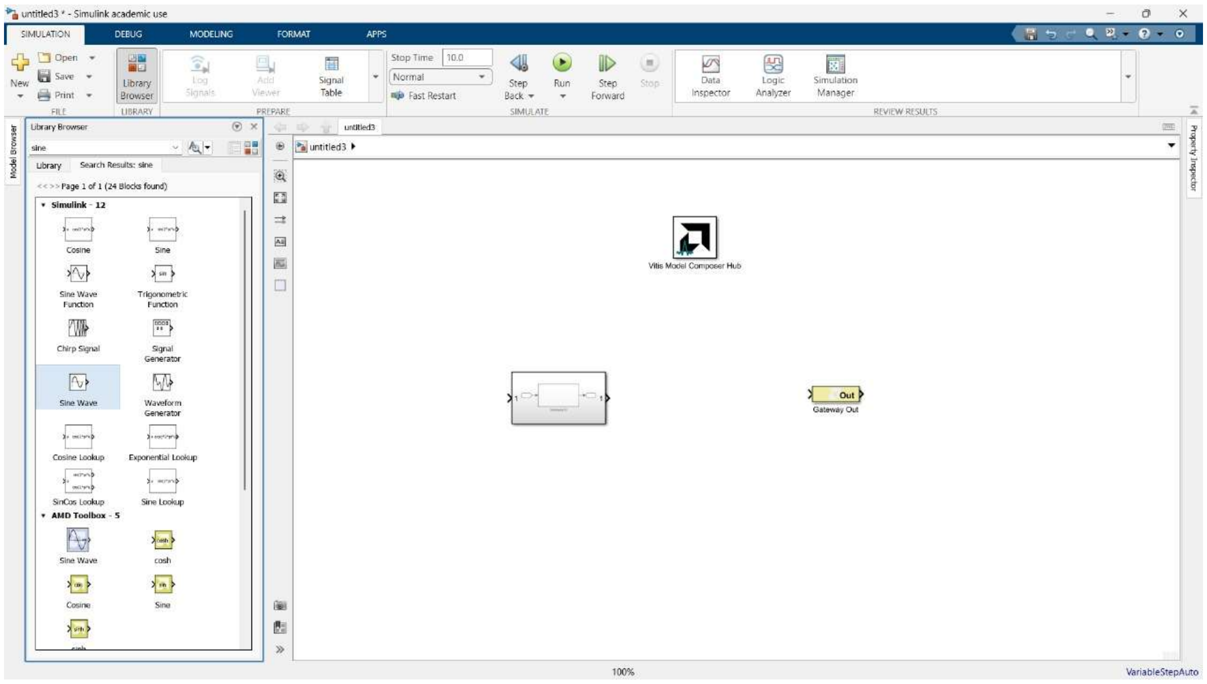


- Your Simulink model should look like this.

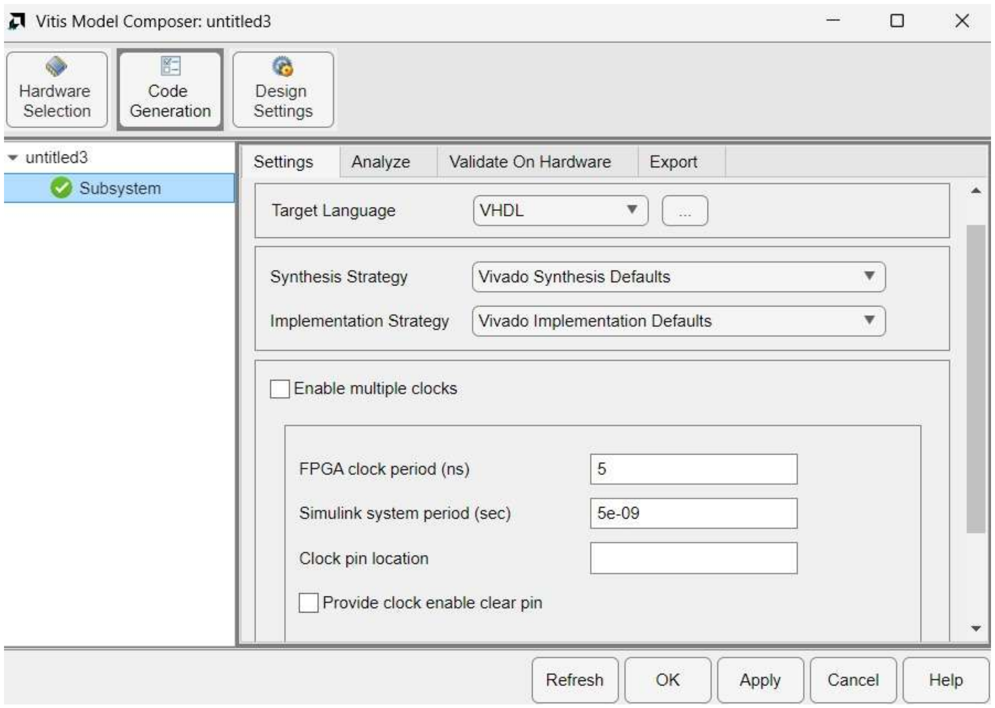


-

- **Step 6:-** Search for a sine wave in your library browser



- **Step 7:-** Double click on the VITIS Model Composer Hub and select the code generation and click on the subsystem input the parameters as follows.



Target Language: VHDL

Synthesis Strategy: Vivado Synthesis Defaults

Implementation Strategy: Vivado Implementation Defaults

☐ Enable multiple clocks

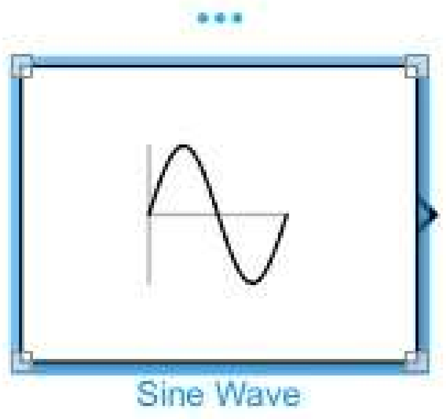
FPGA clock period (ns): 5

Simulink system period (sec): 5e-09

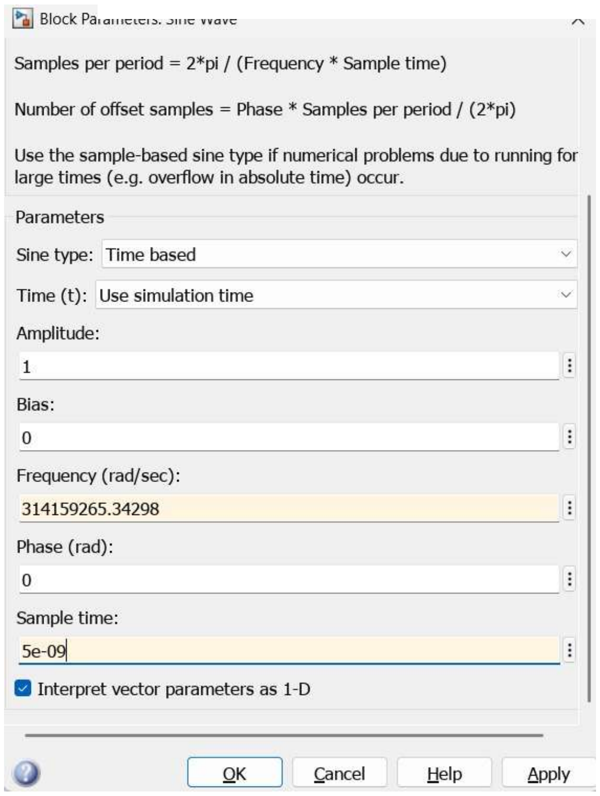
Clock pin location:

☐ Provide clock enable clear pin

- **Step 8:-** Search the Sine wave tool box from your library browser



- Drag that sine
- Select the par


Block Parameters: Sine Wave

Samples per period = 2*pi / (Frequency * Sample time)

Number of offset samples = Phase * Samples per period / (2*pi)

Use the sample-based sine type if numerical problems due to running for large times (e.g. overflow in absolute time) occur.

Parameters

Sine type: Time based

Time (t): Use simulation time

Amplitude: 1

Bias: 0

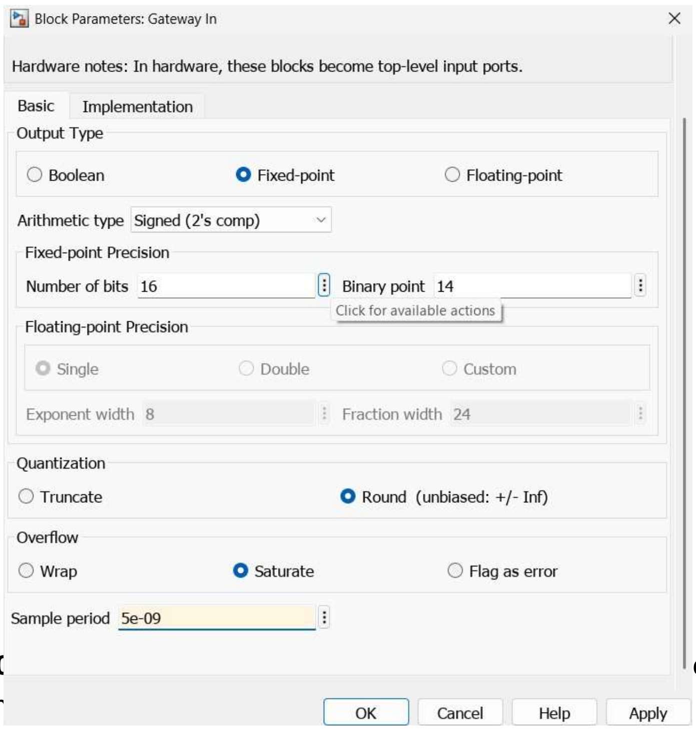
Frequency (rad/sec): 314159265.34298

Phase (rad): 0

Sample time: 5e-09

☑ Interpret vector parameters as 1-D

- **Step 9:-** Double click on the gateway in and input the parameters as follows.



Block Parameters: Gateway In

Hardware notes: In hardware, these blocks become top-level input ports.

Basic | Implementation

Output Type

( ) Boolean [✓ Fixed-point] ( ) Floating-point

Arithmetic type: Signed (2's comp)

Fixed-point Precision

Number of bits: 16 Binary point: 14

Click for available actions

Floating-point Precision

[✓ Single] ( ) Double ( ) Custom

Exponent width: 8 Fraction width: 24

Quantization

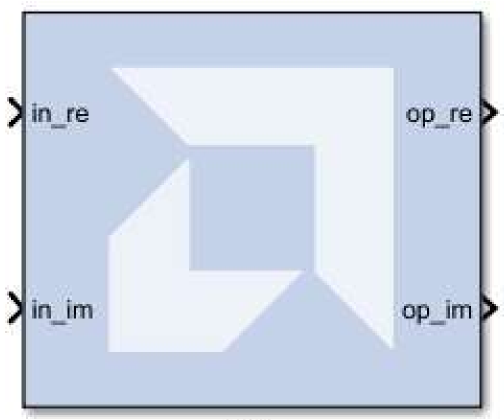
( ) Truncate [✓ Round (unbiased: +/- Inf)]

Overflow

( ) Wrap [✓ Saturate] ( ) Flag as error

Sample period: 5e-09

- **Step 1C** drag it onto your Simulir



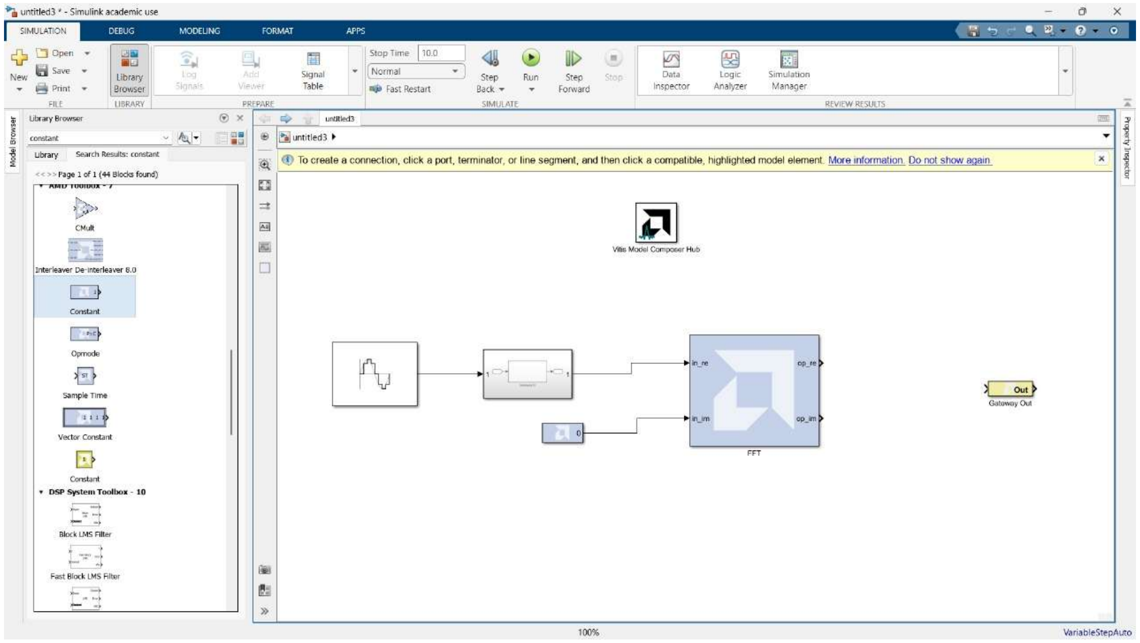
FFT

- **Step 11:- FFT toolbox** has 4 ports
  1. Two i/p ports and two o/p ports.
  2. The two i/p ports are one real port and other is imaginary port, same for the o/p ports.

  Connect the sine wave generator with the gateway in subsystem with the real i/p port of the fft toolbox.

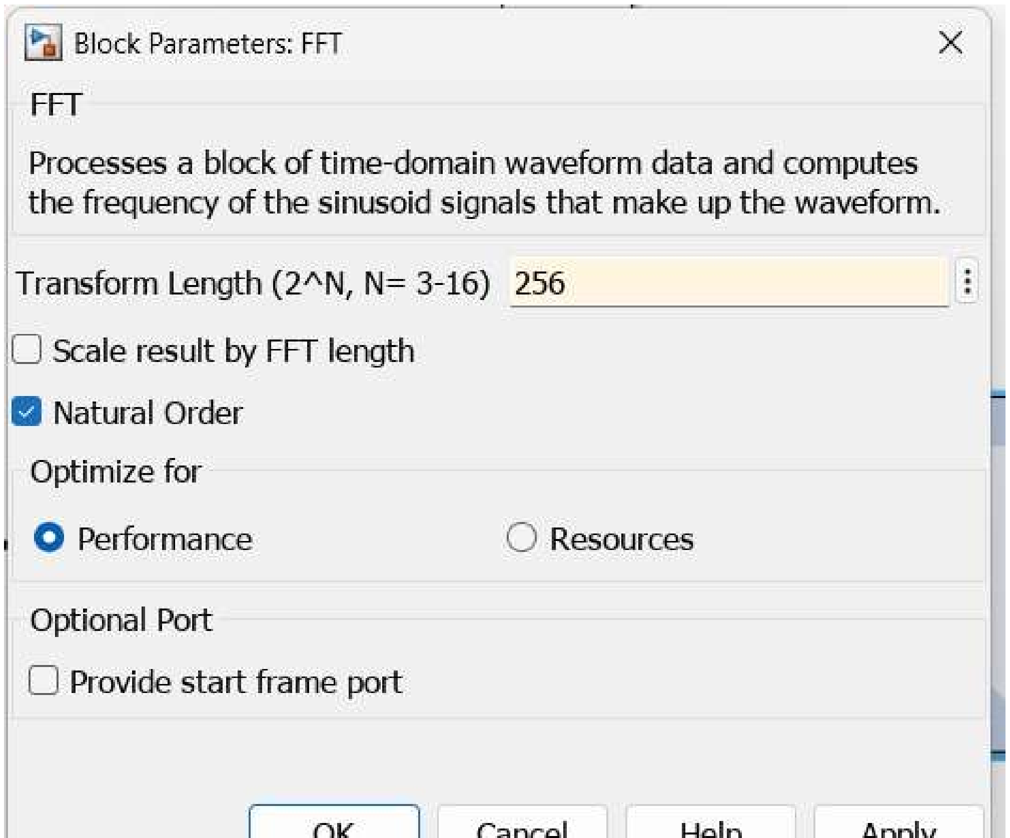
- **Step 12:-** Connect the i/p imaginary port of the fft toolbox with a constant toolbox from the library browser, and select the value of constant toolbox as 0.

  your connection should look like this.



To create a connection, click a port, terminator, or line segment, and then click a compatible, highlighted model element. More information Do not show again

- **Step 13:-** Double click on the FFT toolbox and give the transform length as 256, and select the Natural Order and select the Performance option, Click on apply and click on ok.



Block Parameters: FFT

FFT

Processes a block of time-domain waveform data and computes the frequency of the sinusoid signals that make up the waveform.

Transform Length (2^N, N= 3-16) 256

☐ Scale result by FFT length

☑ Natural Order

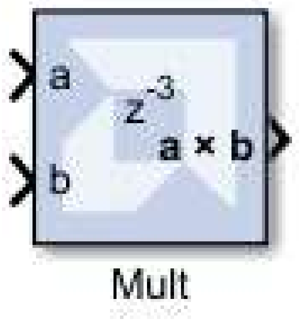
Optimize for

● Performance ○ Resources

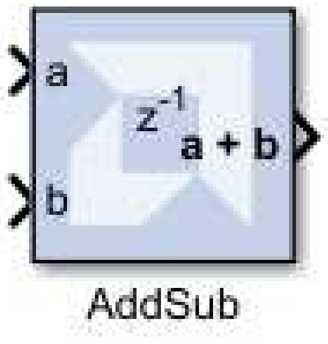
Optional Port

☐ Provide start frame port

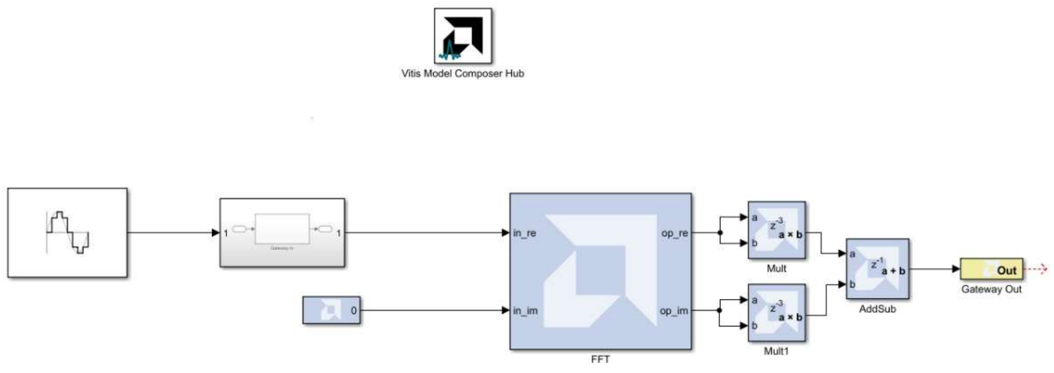
- **Step 14:-** Drag the multiplier toolbox from your library browser



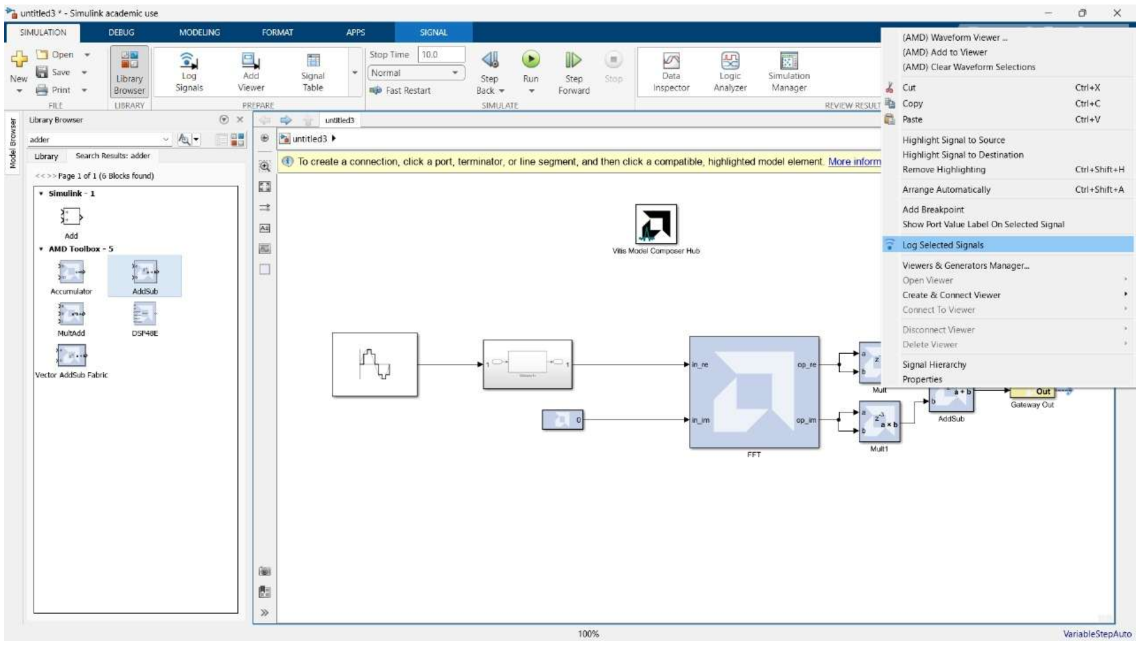
- **Step 15:-** Connect the real output from your fft toolbox to one multiplier toolbox and connect the imaginary output from your fft toolbox to another multiplier toolbox.
- **Step 16:**- Drag an adder toolbox from your library browser.



- **Step 17:-** Connect the AddSub toolbox to gateway out toolbox.
  Your overall connection should look like this.



- **Step 18:-** Right Click on the output of the gateway out toolbox then a dialogue box will appear on the screen then select the (Log Selected Signals)



Stop Time 10.0

Normal

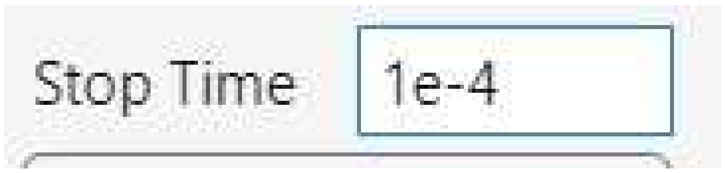
Fast Restart

adder

Page 1 of 1 (6 Blocks found)

To create a connection, click a port, terminator, or line segment, and then click a compatible, highlighted model element. More inform

- **Step 19:-** Adjust the stop time value to 100 micro seconds.

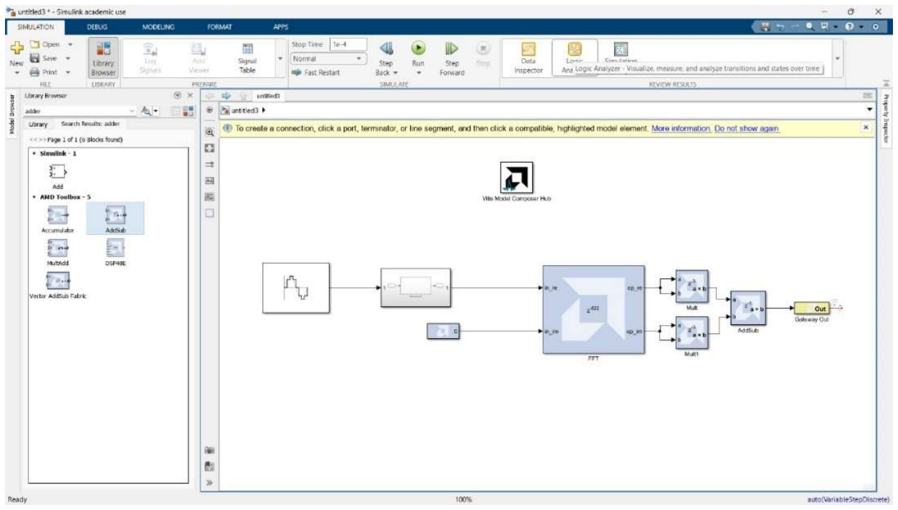


Stop Time 1e-4

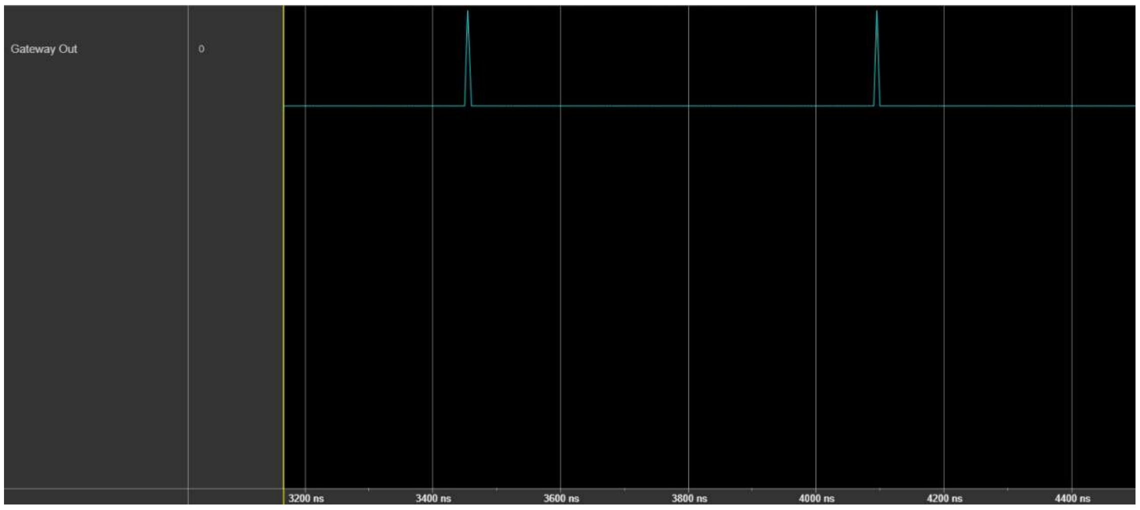
- **Step 20:-** Click on run

VERIFYING THE OUTPUT:-

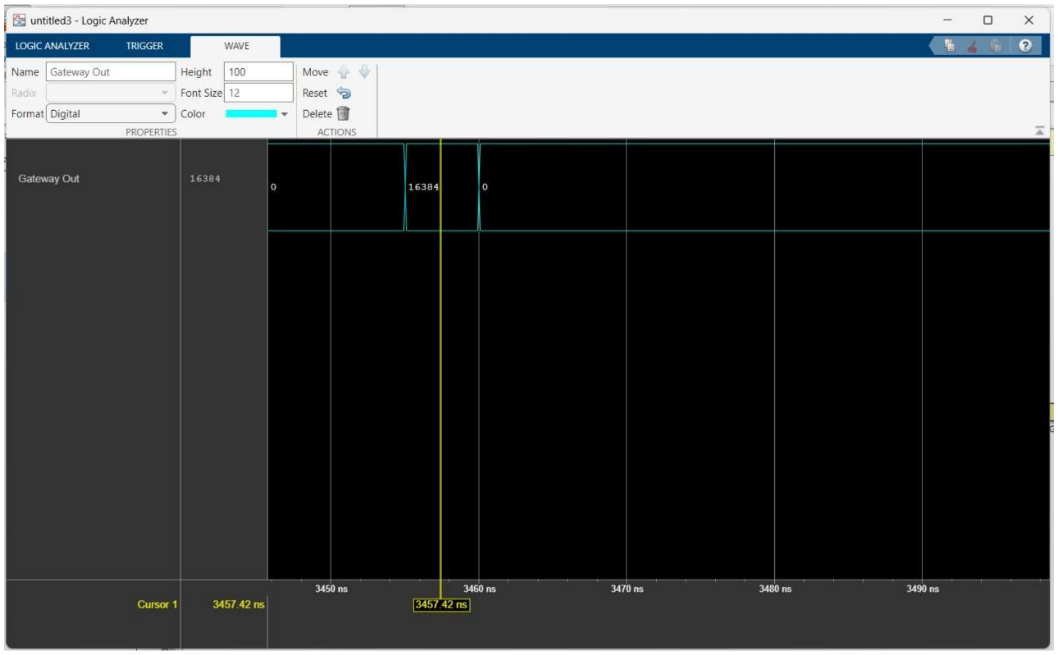
Click on the logic analyzer for the obtained spectrum waveform.


untitled3 * - Simulink academic use
SIMULATION
DEBUG
MODELING
FORMAT
APPS
Stop Time 1e-4
Normal
Fast Restart
Step Back
Run
Step Forward
Stop
Data Inspector
Logic Analyzer - Visualize, measure, and analyze transitions and states over time
Library Browser
adder
Search Results: adder
Page 1 of 1 (6 blocks found)
Simulink - 1
Add
AMD Toolbox - 5
Accumulator
AddSub
MultAdd
DSP48E
Vector AddSub Fabric
To create a connection, click a port, terminator, or line segment, and then click a compatible, highlighted model element. More information. Do not show again
Vitis Model Composer Hub
in_re
in_im
op_re
op_im
FFT
Mult
Mult1
AddSub
Out
Gateway Out
Ready
100%
auto(VariableStepDiscrete)

We will get the output spectrum as below.



In the Digital Form



Name: Gateway Out
Radix:
Format: Digital
Height: 100
Font Size: 12
Color:

Gateway Out 16384

0 16384 0

Cursor 1 3457.42 ns

After obtaining the desired output wave specturm we can clearly observe that there are peak value the value of that peak is 16384.

In the FFT toolbox used in our Simulink model, we can replace the simulated FFT blocks with the actual hardware used in practical applications, ensuring that the output from our Simulink model matches the real-world hardware performance.

In the Simulink model, clock speed and other parameters are set through the Vitis Model Composer Hub and other toolboxes. However, in practical applications, we need to manually adjust the clock speed and tune the frequency to the desired value for hardware execution.

The output obtained from the hardware will be the same as the output from the Simulink model. We can also modify the model by changing parameters and toolboxes as needed, but the output will always match exactly between the software model and the real-world hardware.

CONCLUSION

The project, **"Hardware Simulation for Spectrum Analysis Using Vitis Model Composer,"** successfully demonstrates the development and implementation of an efficient hardware-based solution for real-time spectrum analysis. By leveraging the capabilities of **Vitis Model Composer** for high-level modeling and seamless integration with FPGA hardware, the system addresses critical challenges faced by traditional software-based methods, such as high latency and limited computational efficiency. The incorporation of key signal processing techniques, including Fast Fourier Transform (FFT), noise filtering, and windowing, ensures accurate frequency analysis while optimizing performance for real-time applications.

The project highlights the potential of FPGA-based solutions to handle computationally intensive tasks by utilizing parallel processing and pipelining strategies. Performance evaluations validate the system's ability to achieve low latency, efficient resource utilization, and high accuracy, making it a robust alternative to conventional spectrum analysis methods.

Moreover, the project underscores the flexibility of Vitis Model Composer in bridging the gap between algorithmic design and hardware implementation. Its model-based approach simplifies the design process, reduces development time, and ensures compatibility with advanced hardware platforms.

While the system achieves significant advancements in speed and efficiency, some limitations, such as FPGA-specific resource constraints and the need for further optimization, open avenues for future work. Future extensions may include support for multi-channel spectrum analysis, implementation of advanced algorithms like wavelet transforms, and adaptation to other hardware platforms, such as ASICs.

In conclusion, this project contributes to the growing field of hardware-accelerated signal processing and demonstrates the practical feasibility of real-time spectrum analysis for applications in telecommunications, audio processing, industrial automation, and beyond. It sets a strong foundation for further research and development in high-performance hardware-based signal analysis systems.